

Universidad de Oriente

Sede “Julio Antonio Mella”

Facultad de Ingeniería en Telecomunicaciones Informática y Biomédica

**Trabajo de Diploma**

**En opción al título de Ingeniero en Informática**

Título: “Sistema de gestión para la creación de contratos inteligentes en Cuba”

Autor: Daniela de la Concepción González Domínguez

Tutor: Dr.C. Dionis López Ramos

2023

“Año 65 de la Revolución”

**Pensamiento**

**Dedicatoria**

**Agradecimientos**

**Resumen**

La realización de contratos comerciales y la verificación de las partes involucradas es una actividad importante para el comercio. En los momentos actuales de desarrollo tecnológico y científico, la creación de sistemas y la necesidad de su verificación en el tiempo acordado que sea inalterable por terceros es una problemática que aqueja a los campesinos a la hora de contratar su cosechas y se efectué el pago en tiempo. Para resolver esta necesidad han sido creados los contratos inteligentes que intentar automatizar, verificar y hacer cumplir el estos acuerdos y contratos de manera segura, transparentes e inmutable en una red blockchain.

La importancia de los contratos inteligentes radica en su uso del blockchain que permite su verificación y que sean inalterables por terceros. Al utilizar la cadena de bloques, estas gestiones se realizan de una manera transparente, libre de conflictos y descentralizada, es decir, sin intermediarios. A pesar de todos estos beneficios es difícil cambiar los procedimientos especificados en los contratos inteligentes, y corregir errores en el código puede ser muy costoso laborioso o imposible.

En esta investigación se propone el diseño e implementación de una plataforma para la creación de Contratos Inteligentes que permita resolver los contratos sin pagar a los campesinos cuando se contratan sus cosechas y estos los venden entidades estatales, ya que en la actualidad estas empresas incurren en impagos millonarios a estos campesinos creando graves problemas en la producción de alimentación. Empleando Truffle como entorno de desarrollo, Ethereum para la creación del Contrato Inteligente y Solidity como lenguaje de programación.

Palabras clave: Contratos Inteligentes, Blockchain, intercambio comercial

ABSTRACT

Title: "Platform for the creation of Smart Contracts with Blockchain technology"

Keywords: Smart Contracts, Blockchain, commercial exchange

Índice

**Capítulo 1 Marco Referencial**  **1**

Escribir el título del capítulo (nivel 2) 2

Escribir el título del capítulo (nivel 3) 3

**Escribir el título del capítulo (nivel 1)** **4**

Escribir el título del capítulo (nivel 2) 5

Escribir el título del capítulo (nivel 3) 6

1. **Introducción**

Un contrato comercial se refiere a un acuerdo legal entre dos o más partes mediante el cual se comprometen a realizar o a no realizar ciertas cosas. Los contratos pueden ser escritos o verbales y redactarse de manera formal o informal. La mayoría de las empresas celebran sus contratos por escrito para dejar en claro los términos del acuerdo, y, con frecuencia, buscan asesoramiento legal al celebrar contratos importantes. Los contratos comprenden todo tipo de aspectos comerciales como contrataciones, salarios, seguridad del empleado, locaciones, préstamos, y otros términos. [1]

Los contratos inteligentes han empezado a tomar fuerza en distintos sectores económicos. Todo indica que en un futuro cercano los contratos que realicemos serán inteligentes.  La plataforma más popular y de mayor prestigio es Ethereum, con un tráfico de 13 transacciones por segundo. Según Reuters, en 2021, comenzó a gestarse un nuevo mundo financiero descentralizado, conocido con el término DeFi, que permite a los usuarios intercambiar activos, pedir prestado y prestar sin involucrar a los bancos. [2]

En la actualidad existen herramientas pero son jurídicas, pero hay un problema enorme a la hora de realizar el pago de acuerdo a lo convenido en los contratos sin pagar a los campesinos de sus cosechas cuando son vendidas al estado, por lo que se hace necesario la implementación de soluciones como la que se propone en esta investigación. A partir de la situación problémica descrita anteriormente se identifica como **problema de la investigación:** Los contratos sin pagar a los campesinos cuando se contratan sus cosechas y son vendidas al estado es un problema hoy en día, por lo que se hace necesario implantar un sistema como el que propone esta investigación.

**Objeto de Estudio:** La gestión de la información referente a los contratos sin pagar de los campesinos cuando son contratadas sus cosechas.

**Campo de Acción:** Los sistemas de gestión para la creación de contratos inteligentes en Cuba.

**Objetivo General:** Desarrollar un sistema de gestiónpara la creación de contratos inteligentes en Cuba**.**

**Hipótesis:**

Si se implementará una plataforma de Contratos Inteligentes en aquellas empresas que tienen convenio con los campesinos garantizará el pago de su cosecha según lo convenido.

Teniendo en cuenta lo anteriormente expuesto, se realiza la presente investigación, la cual forma parte de los objetivos de trabajo del Proyecto de Investigación “Soluciones informática para la gestión de los procesos universitarios a partir de plataformas de gobierno electrónico.” del Departamento de Informática de la Universidad de Oriente.

**Objetivo Específicos:**

* Determinar las bases teóricas-conceptuales y metodológicas acerca de la gestión de los contratos inteligentes.
* Desarrollar un sistema informático, que gestione los contratos sin pagar a los campesinos cuando el contrata sus cosechas y garantice el pago en tiempo.

**Tareas de investigación:**

* Realizar un estudio del estado del arte de los sistemas de gestión de la creación de los contratos inteligentes.
* Entrevistar a los usuarios que interactuarán con el sistema para conocer la situación actual en cuanto a la gestión de los contratos sin pagar a los campesinos cuando son contratadas sus cosechas, sus necesidades y expectativas.
* Realizar un estudio del marco de trabajo, herramientas y metodología a utilizar para la elaboración de la aplicación web.
* Describir a través de la Arquitectura, Historias de Usuario, Diagrama de Clases y otros elementos, el diseño del sistema.
* Diseñar el Sistema de Gestión para garantizar el pago de los contratos en tiempo.
* Implementar el Sistema de Gestión.
* Realizar las pruebas pertinentes para comprobar la efectividad y confiabilidad del sistema.

**Métodos de Investigación empleados:**

Para realizar el trabajo fueron empleados los siguientes métodos de investigación:

* Método histórico-lógico: Se aplicó al realizar el análisis de la existencia de otros sistemas que den solución al problema en cuestión.
* Método análisis-síntesis: Se aplicó al realizar el análisis de todo el proceso llevado a cabo en el proyecto y sintetizar las ideas que fueron surgiendo; extrayendo los elementos comunes al objeto de estudio.
* Método inducción-deducción: Se aplicó como estrategia de razonamiento lógico, partiendo del análisis de premisas y principios generales del objeto de estudio, para llegar a conclusiones específicas.

**Estructura del Informe**

El presente Trabajo de Diploma queda estructurado por una introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

En el Capítulo 1 se abordan aspectos del marco teórico conceptual de la investigación, se explican con detalles el uso de las herramientas, lenguajes y las principales tecnologías utilizadas, así como la metodología que se empleó en el proceso de desarrollo del software.

El Capítulo 2 está orientado a describir los elementos fundamentales del diseño del sistema a desarrollar, tales como: requisitos funcionales y no funcionales, historias de usuarios, modelo de clases y descripción de la arquitectura del sistema.

En el Capítulo 3 se reflejan los resultados del diseño e implementación del sistema en términos de componentes (archivos de código fuente, secuencias de instrucciones y ejecutables). Durante la fase de prueba se realizan las descripciones de las pruebas efectuadas a las principales funcionalidades que brinda el sistema.

**Capítulo 1: Marco Referencial**

En este capítulo se explican los principales aspectos teóricos y tecnológicos de la investigación. Se realiza una breve reseña de las principales herramientas, tecnologías y metodología de software empleadas para garantizar la gestión y seguridad de los contratos.

* 1. **La plataforma para la creación automática de contratos inteligentes.**

Una plataforma para la creación automática de contratos inteligentes es un sistema informático que permite a los usuarios crear contratos inteligentes de manera automatizada, utilizando herramientas y plantillas predefinidas. Estas plataformas pueden incluir herramientas de automatización de procesos, integración con otras tecnologías y servicios, y una interfaz de usuario fácil de usar.

* 1. **Análisis del estado del arte de las plataformas existentes de contratos inteligentes.**

Para el desarrollo del sistema, se realizó el estudio del estado del arte de las plataformas, permitiendo realizar una búsqueda y análisis referente a las soluciones relacionadas con el objeto de investigación tanto las plataformas Blockchain de Contratos Inteligentes como las plataformas para la solución de Contratos Inteligentesy de esta manera obtener un mayor conocimiento de estos, permitiendo formular una lista de posibles soluciones que den respuestas a los objetivos planteados.

* + 1. **Plataformas Blockchain de Contratos Inteligentes**

**Ethereum** es una de las plataformas de computación distribuida más populares que hay en la actualidad. **Sirve para ejecutar contratos inteligentes peer to peer (P2P)**en Ethereum Virtual Machine (EVM), que es una máquina virtual descentralizada. Los **contratos inteligentes con Ethereum**están fundamentados en la teoría del Bitcoin, ya que, como esta tecnología, posee mineros e incluso cuenta con un Blockchain propio y la criptomoneda Ether. [5]

**Tezos** es una plataforma de blockchain descentralizada y multifuncional para alojar contratos inteligentes, aplicaciones y crear tokens. Su criptomoneda es el Tez, o Tezzie (XTZ). Esta moneda no está sujeta a la minería, por lo que los usuarios que tienen una cierta cantidad de Tez son recompensados con tokens cuando participan en el consenso de Prueba de Participación (PoS). El objetivo de Tezos es ampliar la funcionalidad de bitcoin mediante contratos inteligentes. Específicamente en crear una blockchain que facilite la adaptación y el cambio a través del consenso de la comunidad con una mínima interrupción de la red. Funciona con un algoritmo Prueba de Participación (PoS). Técnicamente, la red utiliza una de las variantes de PoS, llamada prueba de participación delegada (DPoS). En el lado negativo, el ancho de banda de la red es bajo en la plataforma Tezos, también podemos mencionar que la posibilidad de inflación es bastante alta porque la emisión de tokens es ilimitada. [6]

* + 1. **Plataformas para la solución de Contratos Inteligentes**

**Rootstock e**s una plataforma open source [7] o de código abierto que utiliza Solidity como principal lenguaje y la Prueba de Trabajo (PoW) de Bitcoin por medio de la minería fusionada. RootStock permitirá que los mineros puedan participar de un modo activo debido a que tiene como principal objetivo añadir al ecosistema de Bitcoin valor y funcionalidad a través de smart contracts seguros en la red. Este sistema de gestión se vale de la moneda y la liquidez que tiene el Bitcoin con lo mejor de Ethereum, que es su ecosistema para programar contratos inteligentes. Se puede emplear para hacer pagos instantáneos con unas cadenas laterales (sidechain). En resumen **Rootstock es una solución innovadora para la gestión de la cadena de suministro en la era de la tecnología Blockchain.**

**Counterparty e**s una **plataforma de contratos inteligentes** de libre acceso con la que se pueden crear aplicaciones peer-to-peer, utiliza el lenguaje de programación de los **contratos inteligentes Solidity**para crear diferentes clases de instrumentos financieros y acuerdos digitales. Este se trata de un sistema con el que las acciones del mundo real se pueden transformar en código, y funcionar automáticamente sin presentar fallos ni tener la necesidad de intermediarios. La **moneda de contratos inteligentes**de Counterparty es el XCP, con la que se programan y realizan operaciones con activos digitales. Esta plataforma innovadora que amplía las posibilidades de la tecnología blockchain para la creación y negociación de activos financieros personalizados de manera eficiente y segura. [5]

**EOS e**s una plataforma muy segura [8] con la que se pueden desplegar servicios y aplicaciones distribuidas (DApps). Para crear los smart contracts no usa una máquina virtual sino una API de programación basado en C/C++, que contribuye a que estos programas sean veloces y bastante flexibles, de modo tal que posee una documentación amplia para programar contratos inteligentes.  No obstante, EOS acusa una serie de desventajas que también han de tenerse en cuenta, como son: Sus rivales directos son Ethereum y Cardano, dos plataformas fuertes y sobre las que la comunidad tiene grandes expectativas, y la plataforma no está muy desarrollada.

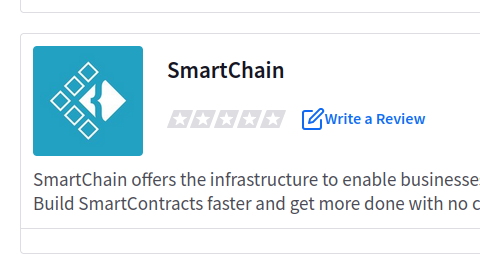
**Codius** en el caso de esta plataforma, cabe mencionar que trabaja diferente a la tecnología de los **contratos inteligentes de Ethereum** o Bitcoin, debido a que estos se ejecutan en computadoras independientes sin que exista una cadena de bloques que sea subyacente, tal como en el alojamiento tradicional. De esta forma, estos pueden interactuar con cualquier API o servicio, escalar de modo infinito y leer o escribir en cualquier clase de blockchain. Por esta razón, es una clase de contratos inteligentes que supera la interoperabilidad que es su desafío esencial.

**Dtecdeal** valiéndose de la tecnología Blockchain, así como el respaldo de abogados especializados y profesionales especialistas en Blockchain, esta plataforma ofrece soluciones contractuales mediante smart contracts sencillo y seguros. Estos crean contratos inteligentes para toda clase negocios, ayudando a **optimizar diferentes clases de actividades y darles a diversos procesos un valor agregado**, ya que con estos se podrá transformar de una forma notable el enfoque digital de las compañías que decidan implementarlos. [5]

* + 1. **Resultado del análisis de las plataformas**

Determinando a criterio del autor de esta investigación que la mejor para el desarrollo de la misma es la plataforma Ethereum, ya que es descentralizada y cuenta con un intérprete de programación más extenso, por lo que para **crear contratos inteligentes** se le puede añadir una lógica mucho más compleja que las otras plataformas que trabajan sobre la red de Ethereum antes mencionadas en la investigación.

**1.2.2 Análisis del estado del arte de la plataforma existente para la implementación de contratos empleando la tecnología blockchain**

SmartChain es la primera plataforma del mundo que proporciona una infraestructura que tiene como objetivo integrar fácilmente cualquier Blockchain en los negocios y habilite el uso de contratos inteligentes en la vida cotidiana sin ningún código ni dificultad. La estrategia de ganar-ganar está impulsada por una base sólida y sólida, por eso cree que todas las partes pueden beneficiarse de las características que SmartChain puede ofrecer. Es plataforma fácil de usar e infalible que está diseñada para un amplio público, tanto profesionales como noveles, convirtiéndose en la puerta de entrada hacia Blockchain. Facilitando la adopción generalizada de Blockchain a través de Implementación de SmartChain. La aplicación de tecnologías como SmartChain con todos sus beneficios le permite no contratar personal especialmente capacitado porque es bastante fácil llegar al fondo de las cosas y no ejerce presión sobre el negocio en general. Intenta hablar el mismo idioma con nuestros clientes. Brinda apoyo que es receptivo las 24 horas del día, los 7 días de la semana para asegurarse de que cada pregunta sea respondida y cada cliente esté satisfecho. Da acceso al seguimiento de transacciones en su cuenta para que esté totalmente informado. [9]

**1.2.2.1 Análisis de la plataforma**

La plataforma SmartChain ofrece una plataforma segura, escalable y de bajo costo para la creación de aplicaciones descentralizadas y contratos inteligentes. Esta tecnología de blockchain tiene el potencial de revolucionar la forma en que se realizan las transacciones y se ejecutan los contratos en línea, y ofrece una amplia gama de posibilidades para la innovación y la eficiencia en una variedad de industrias. En esta investigación se propone una solución orientada al mismo objetivo pero con variaciones desde el punto de vista que la interfaz de usuario sea aún más sencilla y fácil de usar para usuarios con el mínimo conocimiento de la tecnología con la que esta interactuando.

**1.3 Breve historia de la Blockchain y los Contratos Inteligentes**

La historia de la blockchain comienza en 2008, cuando se publicó un documento técnico bajo el seudónimo de Satoshi Nakamoto que describía una nueva forma de crear una moneda digital llamada Bitcoin. La idea principal detrás de Bitcoin era crear una moneda descentralizada que no dependiera de un banco central o de un gobierno para su emisión y validación. En su lugar, Bitcoin utilizó una tecnología de registro distribuido llamada blockchain para registrar todas las transacciones en una base de datos descentralizada y segura.

La blockchain de Bitcoin fue la primera implementación exitosa de una tecnología de registro distribuido y abrió la puerta a una amplia gama de aplicaciones en diferentes industrias. A medida que la popularidad de Bitcoin creció, también lo hizo el interés por la blockchain, y muchas empresas comenzaron a explorar formas de utilizar esta tecnología para mejorar sus procesos y servicios. [3]

En 2014, la plataforma Ethereum introdujo la idea de los contratos inteligentes, que son programas informáticos que se ejecutan en la blockchain y permiten la automatización de acuerdos entre partes. Ethereum utilizó una versión de la blockchain para permitir la creación y ejecución de contratos inteligentes, lo que permitió a los desarrolladores crear aplicaciones descentralizadas y servicios innovadores que aprovechaban esta tecnología.

Desde entonces, la blockchain y los contratos inteligentes han sido utilizados en una amplia gama de aplicaciones, desde la gestión de identidad y la votación electrónica hasta la creación de mercados descentralizados y la automatización de procesos empresariales. La blockchain también ha sido utilizada para mejorar la seguridad y la transparencia en los procesos de supply chain, la gestión de activos y la atención médica, entre otros. [4]

En resumen, la blockchain y los contratos inteligentes han evolucionado rápidamente desde su creación en 2008 y han revolucionado la forma en que se realizan las transacciones y se ejecutan los contratos en línea. Han creado nuevas oportunidades para la innovación y la eficiencia en una amplia gama de industrias, y han abierto la puerta a nuevas formas de colaboración y coordinación entre empresas y personas en todo el mundo.

**1.4 Gestión**

El término gestión es utilizado para referirse al conjunto de acciones, o diligencias que permiten la realización de cualquier actividad o deseo. Gestión, se refiere a todos aquellos trámites que se realizan con la finalidad de resolver una situación o materializar un proyecto. En el entorno empresarial o comercial, la gestión es asociada con la administración de un negocio [17].

**1.5 Sistema de Gestión**

Un sistema de gestión es un conjunto de reglas y principios relacionados entre sí de forma ordenada, que contribuyen a la gestión de procesos generales o específicos de una organización. Permite establecer una política, unos objetivos y alcanzar dichos objetivos [18].

* 1. **Herramientas, Lenguajes de Programación, Tecnologías.**

Las herramientas son objetos elaborados a fin de facilitar la realización de una tarea, estos objetos fueron diseñados para facilitar la realización de cualquier trabajo. Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación son lenguajes formales que pueden usarse para crear programas que pongan en práctica algoritmos específicos los cuales controlan el comportamiento físico y lógico de una computadora. La tecnología es la ciencia aplicada a la resolución de problemas concretos. Constituye un conjunto de conocimientos científicamente ordenados, que permiten diseñar y crear bienes o servicios que facilitan la adaptación al medio ambiente y la satisfacción de las necesidades humanas. Un entorno de desarrollo es un conjunto de herramientas, tecnologías y configuraciones que se utilizan para escribir, probar y depurar software.

**NextJS** es un framework JavaScript ligero y de código abierto creado sobre React, que permite desarrollar aplicaciones y sitios web muy rápidos y fáciles de usar. Aunque tiene una curva de aprendizaje, incluso los desarrolladores nuevos de front-end pueden aprenderlo rápidamente. Se basa en Babel y Node [10]. Se utilizó para el front-end del contrato inteligente.

**Truffle** es un marco de trabajo para el desarrollo de aplicaciones descentralizadas (dApps) y contratos inteligentes en la cadena de bloques. Es una herramienta popular en la comunidad de Ethereum y se utiliza para simplificar el proceso de desarrollo de dApps y contratos inteligentes en la cadena de bloques. Truffle proporciona una serie de herramientas y marcos de trabajos para ayudar a los desarrolladores a escribir, probar, implementar y administrar contratos inteligentes y dApps. [19]

**Visual Studio Code (VS Code)** es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación [11]. Se utilizó como editor de código del contrato inteligente.

**Solidity es un lenguaje de programación**, pero no está diseñado para crear programas normales, sino que es un lenguaje **específicamente creado para programar contratos inteligentes.** Su sintaxtis está basada en ECMAScript, y similar a otros lenguajes como JavaScript y C, pero con la diferencia de implementar un tipado fuerte a la hora de declarar el tipo de variables y argumentos. Esto es así para garantizar el rigor del contrato. Este lenguaje es capaz de compilar los contratos que se crean en el código de la red de Ethereum, i vincularlos a ella. Solidity fue creado en 2014 por diferentes colaboradores del Proyecto Ethereum. Concretamente, es un lenguaje **creado para ejecutarse en las Ethereum Virtual Machines**(EVM) que funcionan sobre la blockchain de Ethereum [12].

**TypeScript (TS) es un lenguaje de programación construido a un nivel superior de JavaScript (JS).** Esto quiere decir que TypeScript dota al lenguaje de varias características adicionales que hacen que podamos escribir código con menos errores, más sencillo, coherente y fácil de probar, en definitiva, más limpio y sólido. Fue creado por **Microsoft** en 2012 y, desde entonces, su adopción no ha hecho más que crecer. Especialmente, desde que **Google** decidió adoptarlo como lenguaje por defecto para [desarrollar conAngular](mhtml:file://C:\Users\Daniela Gonzalez D\Desktop\TypeScript_ qué es, diferencias con JavaScript y por qué aprenderlo.mhtml!https://profile.es/blog/angular-templates-las-directivas-ng-template-ng-container-y-ngtemplateoutlet/amp/)**.** Aunque, hoy en día, podemos desarrollar con TypeScript en cualquiera de los **frameworks o librerías** más punteras, como son [React](mhtml:file://C:\Users\Daniela Gonzalez D\Desktop\TypeScript_ qué es, diferencias con JavaScript y por qué aprenderlo.mhtml!https://profile.es/blog/librerias-react/amp/)para el frontend o **Node** para el backend. [13]

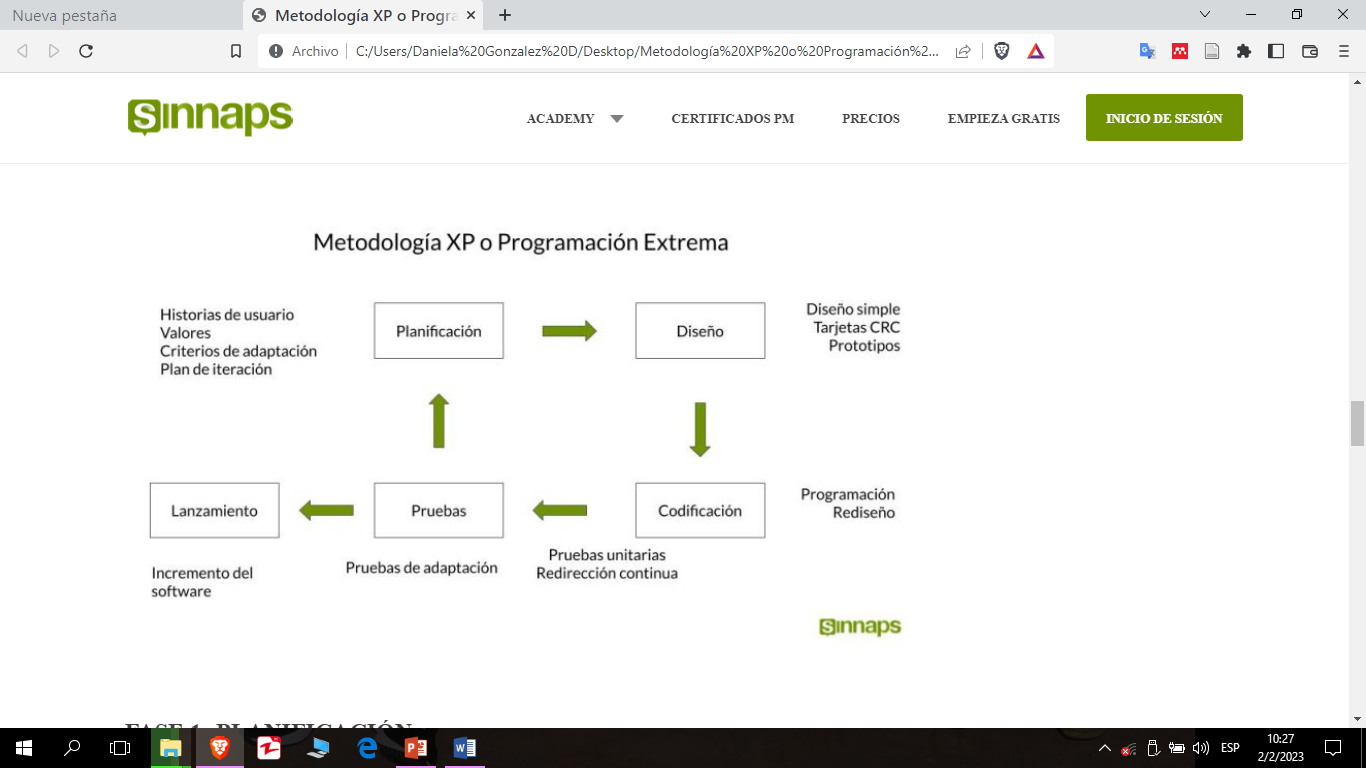
**Git** es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones. [14]

**1.7 Metodología de desarrollo de software**

**Programación Extrema (XP)**

La metodología XP es un conjunto de técnicas que dan agilidad y flexibilidad en la gestión de proyectos. También es conocida como Programación Extrema (Extreme Programming) y se centra crear un producto según los requisitos exactos del cliente. De ahí, que le involucre al máximo durante el método de gestión del desarrollo del producto.

La primera vez que oímos este tipo de metodología fue a través del libro *Extreme Programming Explained: Embrace Change* (1999), escrito por el ingeniero de software Kent Beck. El uso de esta metodología supone, para muchos teóricos, una aproximación a la calidad óptima del producto. Pues durante el ciclo de vida del software, ocurren cambios naturales. Es más, cuanto más cambios, puede que más cerca estemos del mejor resultado que espera nuestro cliente. Por eso, este cambio constante en el proyecto se llega a considerar como favorable. Y si podemos aplicar una manera dinámica de gestionarlos. Esta forma es conocida como metodología XP.



## Imagen 1: Metodología XP: Fases

### Fase 1: Planificación

Según la identificación de las historias de usuario, se priorizan y se descomponen en mini-versiones. La planificación se va a ir revisando. Cada dos semanas aproximadamente de iteración, se debe obtener un software útil, funcional, listo para probar y lanzar.

### Fase 2: Diseño

En este paso se intentará trabajar con un código sencillo, haciendo lo mínimo imprescindible para que funcione. Se obtendrá el prototipo. Además, para el diseño del software orientado a objetos, se crearán tarjetas CRC (Clase-Responsabilidad-Colaboración).

### Fase 3: Codificación «de Todos»

La programación aquí se hace «a dos manos», en parejas en frente del mismo ordenador. Incluso, a veces se intercambian las parejas. De esta forma, nos aseguramos que se realice un código más universal, con el que cualquier otro programador podría trabajar y entender.  Y es que deber parecer que ha sido realizado por una única persona. Así se conseguirá una programación organizada y planificada.

### Fase 4: Pruebas

Se deben realizar pruebas automáticas continuamente. Al tratarse normalmente de proyectos a corto plazo, este testeo automatizado y constante es clave. Además, el propio cliente puede hacer pruebas, proponer nuevas pruebas e ir validando las mini-versiones.

### Fase 5: Lanzamiento

Si hemos llegado a este punto, significa que hemos probado todas las historias de usuario o mini-versiones con éxito, ajustándonos a los requerimientos del cliente. Tenemos un software útil y podemos incorporarlo en el producto. [15]

**Conclusiones del capítulo**

Se concretaron los elementos teóricos-prácticos que ayudaron a conformar el modelo conceptual del proyecto, los distintos factores y elementos que fueron objeto de estudio para la realización del mismo, así como la metodología de desarrollo de software. Se analizaron las herramientas disponibles para ejecutar dichas tareas y se determinaron cuales se adaptan mejor para la implementación del sistema.

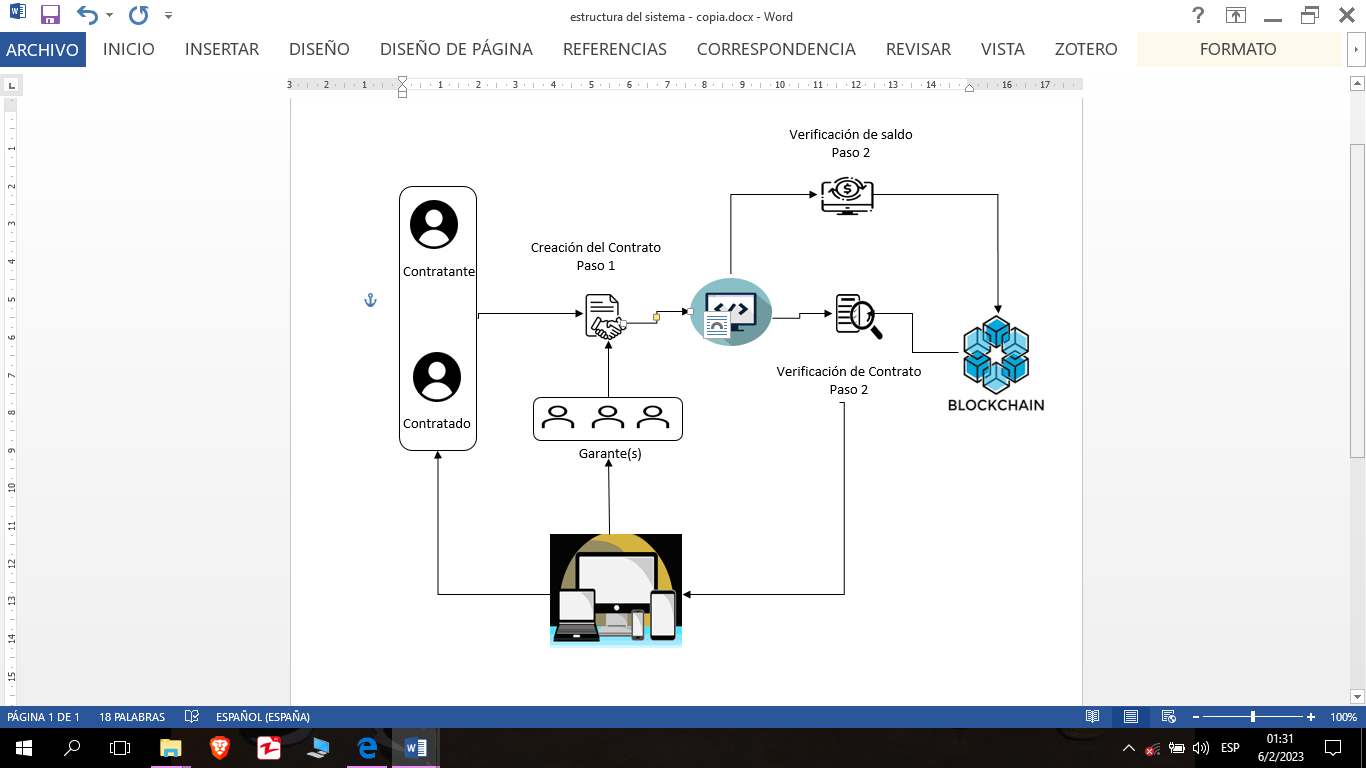
# 

# **Capítulo 2. Planificación y Diseño**

Este capítulo está orientado a describir los elementos fundamentales del diseño del sistema a desarrollar, tales como: la propuesta del sistema y sus usuarios, roles y responsabilidades identificados. Se declaran las funcionalidades del sistema, así como las historias técnicas y las historias de usuario de las funcionalidades más importantes que se implementan en el software. También se aborda a profundidad todo lo relacionado con los sistemas de recomendación y los servicios web, así como los elementos a tener en cuenta para su implementación.

**2.1 Propuesta del Sistema**

Con el objetivo de dar solución al problema planteado anteriormente se propone desarrollar de un Contrato Inteligente empleando la cadena de bloques de Ethereum y una aplicación web que permita a partir de la entrada de datos generar, almacenar e interactuar con los contratos.



**Figura 2.1.** **Estructura del Sistema**

**2.2** **Roles y responsabilidades**

Para llevar a cabo las diferentes tareas durante el desarrollo del software, es necesario definir los roles y las responsabilidades que le corresponde a cada miembro del equipo de trabajo. La tabla 2.1 muestra los roles y las responsabilidades adquiridas durante todo el proceso.

**Tabla 2.1:** Roles y responsabilidades

|  |  |  |  |
| --- | --- | --- | --- |
| **Roles** | **Nombre y Apellidos** | **Estatus** | **Descripción** |
| **Jefe de proyecto** | Dionis López Ramos | Doctor en Ciencias | Es el encargado de dirigir todo el proceso vinculado al desarrollo e implementación del sistema. |
| **Cliente** | ANAP y el Banco BPA | Entidad | Determina qué construir y brinda la información para la confección de los requisitos del negocio. |
| **Programador** | Daniela de la Concepción González Domínguez | Estudiante | Responsable de las decisiones técnicas, construcción del sistema, diseño y realización de las pruebas. |
| **Analista** | Daniela de la Concepción González Domínguez | Estudiante | Reúne toda la información referente al problema y especifica sus requisitos en cada fase de desarrollo. |
| **Diseñador** | Daniela de la Concepción González Domínguez | Estudiante | Se encarga de todo el diseño del software en correspondencia con los requisitos del cliente. |
| **Arquitecto** | Daniela de la Concepción González Domínguez | Estudiante | Define las partes integrantes del sistema |
| **Probador** | Daniela de la Concepción González Domínguez | Estudiante | Ejecuta las pruebas y difunde los resultados de las mismas. |

**2.2 Usuarios del Sistema**

Los usuarios son todas aquellas personas que interactúan de alguna forma con esta herramienta o que desempeñan algún rol específico. A continuación, se describen las actividades que realiza cada rol de usuario.

**Tabla 2.2 Usuarios del Sistema**

|  |  |
| --- | --- |
| **Usuario** | **Descripción** |
| Blockchain | Acceso a funciones internas del sistema, base de datos y procesos en general del sistema. |
| Cliente | Acceso a realizar las operaciones comunes para lo que fue diseñada la herramienta, dichas operaciones se realizan desde la interfaz gráfica de la misma. |

## **2.3 Planificación del desarrollo del Sistema**

La planificación es la etapa inicial de todo proyecto. Es aquí donde se comienza a interactuar con el cliente para descubrir los requerimientos del sistema y realizar los ajustes a la metodología según las características del software. Se definen los requisitos funcionales y no funcionales.

### **2.3.1. Funcionalidades**

**F1:** Crear Contrato: Le permitirá a los usuarios crear el contrato con la información requerida por el sistema.

Entrada: Datos

Salida: Formulario

**F2:** Verificación del estado de cuenta: el sistema verificará si la parte contratante dispone en su billetera de la suma necesaria para realizar el contrato.

Entrada: Importe del contrato

Salida: Verificación sí o no.

**F3:** Carga y guardado de datos: Permitirá guardar los datos en la Blockchain de Ethereum y la consulta de los contratos guardados

Entrada: Datos y resultados.

Salida: Guarda los datos con posibilidad de volver a ser usados.

**F4:** Transferencia de saldo a cuenta demo: Después de ser verificada la liquidez necesaria para establecer el contrato en la billetera del contratante, se transferirá la suma equivalente al importe del contrato a una cuenta demo donde estará hasta la culminación de este.

**F5:** Carga del contrato: Al cumplirse el término de tiempo establecido en el contrato se le notificara a los usuarios (contratante, contratado y el garante(s)) de la culminación del mismo.

Entrada: Identificador del contrato y la fecha

Salida: Correo electrónico y estado de contrato

**F6:** Validación de la entrega de mercancía: Le permitirá a los usuarios verificar si se cumplieron los términos del contrato.

Entrada: Identificador del contrato, estado del contrato, usuarios y garantes del contrato.

Salida: Formulario para la validación de la entrega de la mercancía.

**F7:** Pago del contrato: Si se verifica la entrega de la mercancía se valida de acuerdo a los términos del contrato, se transferirá automáticamente el dinero de la cuenta demo a la cuenta de la parte contratada.

Entrada: Importe del contrato

Salida: Confirmación de transferencia

**F8:** Cancelación de Contrato: Si la verificación de la entrega de la mercancía no se valida de acuerdo a los términos establecidos en el contrato se eliminara el contrato y automáticamente se transferirá el dinero de vuelta a la billetera del contratante.

Entrada: Importe del contrato

Salida: Confirmación de cancelación

**2.3.2. Características para las funcionalidades**

**Requisitos de Software**

* Manjaro Linux Gnome 21.0.3 o Windows 10
* NodeJS

**Requisitos de Hardware**

Los requerimientos mínimos de hardware para correr la aplicación:

* Procesador Intel Corei7 a 4.0 GHz
* 8.0 GB de memoria RAM
* Buena conexión a Internet

### **2.3.3. Historias técnicas**

Las funcionalidades son complementadas por las historias técnicas que se enfocan en el diseño o la implementación. Las historias técnicas constituyen las propiedades o cualidades que el producto debe tener.

**Usabilidad**

La aplicación web debe brindar al usuario una clara navegabilidad de las funcionalidades del sistema.

**Confiabilidad**

La aplicación web debe ser capaz de funcionar correctamente cada vez que se utilice, las funciones para las que fue diseñado deben arrojar los resultados que el usuario espera, dichos resultados deben ser íntegros.

**Seguridad**

El espacio de trabajo de cada usuario será protegido mediante autenticación desde la billetera de criptomoneda que utiliza el usuario para entrar y poder realizar acciones con la herramienta. De esta forma se garantiza la seguridad e integridad de la información del usuario.

### **2.3.4. Historias de Usuario**

Las Historias de Usuarios (HU) son la técnica utilizada en XP para especificar los requisitos de software, tanto funcionales como no funcionales y se dividen en tareas de ingeniería asignadas a los programadores para ser implementadas durante las iteraciones. Están escritas con el vocabulario del cliente, no con vocabulario técnico. El procedimiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, emplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas.

La prioridad de las Historias de Usuario se clasificó de la siguiente manera:

* Alto: requerimientos que no pueden faltar.
* Medio: requerimientos que pueden ser implementados en otra iteración.
* Bajo: requerimientos adicionales.
* HU.1 Crear el Contrato
* HU.2 Verificación del estado de cuenta
* HU.3 Carga y guardado de datos
* HU.4 Transferencia de saldo a cuenta demo
* HU.5 Carga del contrato
* HU.6 Validación de la entrega de mercancía
* HU.7 Pago del contrato
* HU.8 Cancelación de Contrato

**Historias de Usuario**

**Tabla 2.3 Historia de Usuario: iniciar sesión**

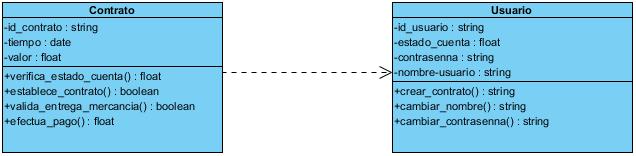
|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:1** | **Nombre de HU: Iniciar sesión** |
| **Usuario:** Cualquier usuario | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Bajo |
| **Descripción:** Donde el usuario inicia sesión para acceder a la plataforma | |
|  | |

**Tabla 2.4 Historia de Usuario: Crear el Contrato**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:2** | **Nombre de HU:** Crear el Contrato |
| **Usuario:** Cualquier usuario | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Bajo |
| **Descripción:** Donde el usuario completa el formulario con los datos requeridos. | |
|  | |

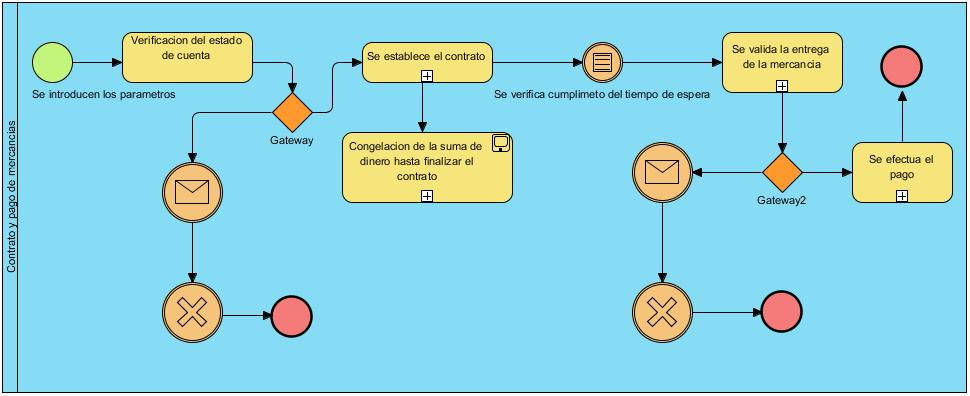
**2.4 Diagrama de Clases del Diseño**

Un Diagrama de Clases del Diseño no es más que la representación a través de sus respectivos estereotipos de las Clases del Diseño, el papel que juegan y las relaciones existentes entre ellas. Permite comprender cómo interactúan las diferentes clases del diseño dependiendo de las funcionalidades a cumplir [20]. En la figura 2.2 están representadas las clases que se identificaron durante el proceso de análisis y diseño del sistema, con sus correspondientes atributos y las relaciones que existen entre ellas.

**Figura 2.2:** Diagrama de clases del Diseño

**2.5 Diagrama de flujo para la verificación del cumplimiento del contrato**

Un diagrama de flujo es una forma visual la cual nos permite representar un algoritmo o proceso. En un diagrama de flujo se usan diferentes elementos y conexiones la cuales permiten representar el algoritmo de una manera visual, estructurada y organizada. [21] En la figura 2.3 se explica el proceso del contrato que comienza con la introducción de los parámetros (pago y fecha de culminación), se verifica el estado de cuenta del contratante en caso de no tener el capital necesario se le envía un mensaje notificando y se termina el proceso en caso que si se disponga del dinero que se establece en el contrato (se congela la suma de dinero de la cuenta hasta el final del contrato para garantizar su existencia a la hora de hacer el pago), al cumplirse el tiempo acordado se valida la entrega de la mercancía en caso de no entregarse se notifica a las partes del contrato y se cancela. En caso de que se entregue se realiza el pago de forma automática y se culmina el contrato.

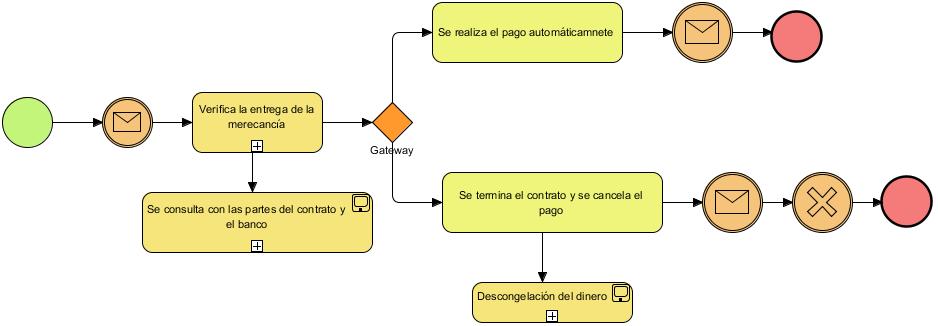


**Figura 2.3 Diagrama de flujo para la verificación del cumplimiento del contrato**

Fuente: Business Process Model and Notation (BPMN)

**2.6 Diseño del protocolo o algoritmo de consenso para reconocer el cumplimiento de las condiciones.**

El diseño de un protocolo de consenso puede ser un proceso complejo que involucra consideraciones sobre seguridad escalabilidad, eficiencia y equidad. En la figura 2.4 se describe el cumplimiento de las condiciones para realizar el contrato primeramente se manda un mensaje verificando la entrega de la mercancía y se consulta con las partes del contrato y el banco. Si es entregada la mercancía se realiza el pago automáticamente, sino se termina el contrato y se cancela el pago y se descongela el dinero.



**Figura 2.4** **Diseño del protocolo o algoritmo de consenso para reconocer el cumplimiento de las condiciones.**

Fuente: Business Process Model and Notation (BPMN)

**Conclusiones del capítulo**

En el presente capítulo se mostraron aspectos relacionados con las fases de planificación y diseño del desarrollo del software. Quedaron expuestas las funcionalidades a implementar definidas por los artefactos que propone la metodología XP, se mencionan los requisitos tanto funcionales como los no funcionales, las historias de usuarios y las historias técnicas.

# **Capítulo 3. Implementación y Pruebas del Sistema**

En este capítulo se presentarán algunas de las funcionalidades del sistema diseñado y detalles de su implementación, explicando las herramientas de software utilizadas y aspectos relevantes que han tenido lugar en su elaboración. También se realizan las pruebas pertinentes a algunas de las funcionalidades del sistema, evaluando de esta manera el software y su correcto funcionamiento.

## **Estructura del proyecto**

Antes de empezar a programar la aplicación, fue necesario adaptar todas las funcionalidades y esquemas a la filosofía de trabajo de la Suite Truffle, lo cual aseguró el desarrollo e implementación del sistema lo más eficientemente. Truffle es un marco de trabajo para el desarrollo de aplicaciones descentralizadas y contratos inteligentes en la cadena de bloques. A continuación, se describen algunos de los elementos principales de Truffle:

* **Truffle Suite** es un conjunto de herramientas que incluye Truffle, Ganache, Drizzle y otros componentes. Ganache es un simulador de red de Ethereum que permite a los desarrolladores probar sus contratos inteligentes en un entorno simulado, mientras que Drizzle es una biblioteca para la gestión del estado de la aplicación.
* **Truffle Framework** es un marco de trabajo para el desarrollo de contratos inteligentes en la cadena de bloques. Incluye una sintaxis de lenguaje de programación, una biblioteca de contratos inteligentes y una serie de herramientas de línea de comandos para ayudar en el desarrollo, prueba y despliegue de los contratos inteligentes.
* **Truffle Boxes** son proyectos de plantilla pre-configurados que incluyen una estructura de proyecto básica, bibliotecas y herramientas preinstaladas para acelerar el proceso de desarrollo.
* **Truffle Contracts** es una biblioteca de contratos inteligentes que permite a los desarrolladores definir la estructura y la lógica de sus contratos inteligentes en un lenguaje de programación familiar como Solidity.

Truffle se integra con otras herramientas populares de la cadena de bloques, como **Metamask**, una extensión de navegador para interactuar con aplicaciones descentralizadas. [19]

La aplicación CSolution (Solución de Contratos) se desarrolla para la creación de contratos inteligentes resolviendo el problema de los contratos sin pagar a los campesinos cuando se contratan sus cosechas en Cuba. La misma está compuesta por un conjunto de directorios, dentro de los que se destacan:

* El directorio **“pages/”:** está constituido por las vistas de la aplicación.
* El directorio **“src/”:** está el código de los componentes de la aplicación (botones, estilos, etc.) y el archivo configuración para conectar los contratos con las vistas.
* El directorio **“contract/”:** están los contratos inteligentes con los cuales funcionan la aplicación.

## **Instalación de las herramientas utilizadas**

Manjaro Linux:

1. Instalación de NodeJS en el LTS versión 18.13.0 que es uno de los últimos, para esto uso nvm:

* En primer medida debemos de actualizar los repositorios de Manjaro para esto ejecutamos:
* sudo pacman –Syy
* Si el comando anterior no presenta errores te recomiendo saltar este paso siguiente:
* sudo pacman –Syyu
* Después de tener actualizado los repositorios de Manjaro se procede a la instalación nodejs para esto utilizamos el comando;
* sudo pacman -S nodejs npm
* Luego de esto se descargara e instalara node.js para comprobar que quedo bien instalado puedes comprobar el comando:
* npm –version

1. Instalación de Truffle Suit:

* En la terminal: npm install -g truffle
* truffle version

1. Instalación Ganache:

* Descargar: <https://github.com/trufflesuite/ganache-ui/releases>
* Ejecutarlo en la terminal.

1. Instalación React:

* sudo npm install -g create-react-app

1. Instalación opcional de GitHub Desktop: Ejecutamos Instalador, una vez instalado pudimos poner nuestra cuenta personal en GitHub y alojar nuestro proyecto allí para tenerlo salvado y controlar todos los cambios que se hicieron durante la implementación del mismo.

## **3.3 Implementación de Funcionalidades**

## **3.4 Análisis económico**

En la realización de un proyecto se hace necesaria la planificación y el control del esfuerzo, costo y tiempo que tomará llevarlo a cabo. Con la utilización de métodos de estimación de costos, se puede determinar una aproximación de los recursos necesarios, así como el total de tiempo que gastaría una persona o un equipo, en el desarrollo de un producto de software específico. Además, se determina la viabilidad económica, ambiental, técnica y de mercado. A continuación, se realiza un análisis de costos para el sistema.

### **3.4.1. Estimación de costo y tiempo**

Para determinar los costos de los sistemas desarrollados se usó el método de puntos en casos de uso [22]. Este es un método de estimación prometedor que se adapta bien al enfoque de caso de uso para la descripción de los requisitos. En sus bases yace el concepto de transacción de caso de uso, la unidad más pequeña de medición. Se realizó este análisis teniendo en cuenta las Historias de Usuario proporcionadas por la metodología XP.

El método de punto de casos de uso consta de cuatro etapas, en las que se desarrollan los siguientes cálculos:

**Ecuación 3.1**: Cálculo de los Puntos de Historias de Usuarios sin ajustar

**UUCP = UAW + UUCW**

Donde:

* **UUCP**: Puntos de Historias de Usuarios sin ajustar.
* **UAW**: Factor de Peso de los Actores sin ajustar.
* **UUCW**: Factor de Peso de Historias de Usuarios sin ajustar.

**Factor de Peso de los Actores sin ajustar (UAW)**

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. En la [tabla 3.1](#_Tabla_3.1_Factor) se presenta el Factor de Peso de los Actores sin ajustar.

#### 

#### Tabla 3.1 Factor de peso de los actores sin ajustar.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo** | **Descripción** | **Peso** | **Cant\*peso** |
| Simple | Otro sistema que interactúa mediante una interfaz de programación de aplicaciones. (API) | 1 | 1\*1 |
| Medio | Otro sistema que interactúa mediante un protocolo o una persona interactuando con una interfaz basada en texto. | 2 | 1\*2 |
| Complejo | Una persona que interactúa con el sistema mediante una interfaz gráfica. (GUI) | 3 | 1\*3 |
| Total |  |  | 1+2+3= **6** |

**Para calcular UUCW**

Este valor se calcula mediante un análisis de la cantidad de HU presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los HU se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo según muestra la [tabla 3.2.](#_Tabla_3.2_Peso)

#### Tabla 3.2 Peso de las Historias de Usuarios sin Ajustar

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Descripción | Peso | Cant \* peso |
| Simple | La HU contiene de 1 a 3 transacciones. | 5 | 7\*5 |
| Medio | La HU contiene de 4 a 7 transacciones. | 10 | 1\*10 |
| Complejo | La HU contiene más de 8 transacciones. | 15 | 0\*15 |
| Total |  |  | 35+10+0=**45** |

**Cálculo de los Puntos de Historias de Usuarios ajustadas.**

Una vez que se tienen los Puntos de Historias de Usuarios, se debe ajustar este valor como se muestra en la ecuación 3.2.

**Ecuación 3.2**: Cálculo de los Puntos de Historias de Usuarios ajustadas

**UCP = UUCP \* TCF \* EF**

Donde:

* **UCP**: Puntos de Historias de Usuarios ajustados.
* **UUCP**: Puntos de Historias de Usuarios sin ajustar.
* **TCF**: Factor de complejidad técnica.
* **EF**: Factor de ambiente.

**Factor de complejidad técnica** (**TCF**).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante o nulo y 5 un aporte muy importante. En la siguiente [tabla 3.3](#_Tabla_3.3_Factor) se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores:

#### Tabla 3.3 Factor de complejidad técnica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Factor | Descripción | **Peso** | **Valor** | **(Peso-i \* Valor-i)** |
| T1 | Sistema distribuido | 2 | 1 | 2 |
| T2 | Rendimiento o tiempo de respuesta | 1 | 5 | 5 |
| T3 | Eficiencia del usuario final | 1 | 2 | 2 |
| T4 | Procesamiento interno complejo | 1 | 5 | 5 |
| T5 | El código debe ser reutilizable | 1 | 5 | 5 |
| T6 | Facilidad de instalación | 0.5 | 2 | 1 |
| T7 | Facilidad de uso | 0.5 | 5 | 2.5 |
| T8 | Portabilidad | 2 | 5 | 10 |
| T9 | Facilidad de cambio | 1 | 2 | 2 |
| T10 | Concurrencia | 1 | 4 | 4 |
| T11 | Incluye objetivos especiales de seguridad | 1 | 5 | 5 |
| T12 | Acceso directo a terceras partes | 1 | 2 | 2 |
| T13 | Se requieren facilidades especiales de entrenamiento a los usuarios | 1 | 0 | 0 |
| Total |  |  |  | **45.5** |

**Para Calcular TCF:** Factor de complejidad técnica se muestra la ecuación 3.3 cálculo del factor complejidad técnica.

**Ecuación 3.3:** Cálculo del Factor de complejidad técnica

**TCF = 0.6 + 0.01 \* Σ (Pesoi\* Valori)**

**TCF = 0.6 + 0.01 \*45.5**

**TCF = 1.05**

**Factor Ambiente (EF).**

El factor de ambiente está relacionado con las habilidades y entrenamiento del grupo de desarrollo. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

En la siguiente [tabla 3.4](#_Tabla_3.4_Factor) se muestra factor de ambiente con su significado y el peso de cada uno de estos factores:

#### Tabla 3.4 Factor de ambiente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor** | **Descripción** | **Peso** | **Valor** | **(Peso-i \* Valor-i)** |
| E1 | Familiaridad con el modelo de proyecto utilizado | 1.5 | 3.5 | 5.25 |
| E2 | Experiencia en la aplicación | 0.5 | 1 | 0.5 |
| E3 | Experiencia en orientación a objetos | 1 | 3 | 3 |
| E4 | Capacidad del analista líder | 0.5 | 3.5 | 1.75 |
| E5 | Motivación | 1 | 4.5 | 4.5 |
| E6 | Estabilidad de los requerimientos | 2 | 4 | 8 |
| E7 | Personal Part-Time | -1 | 0 | 0 |
| E8 | Dificultad del lenguaje de programación | -1 | 4 | -4 |
| **Total** |  |  |  | **19** |

|  |  |
| --- | --- |
|  |  |

**Para Calcular EF**: Factor de ambiente se muestra la ecuación 3.4 cálculo del factor ambiente.

**Ecuación 3.4**: Cálculo del Factor de ambiente

**EF** = 1.4 - 0.03 \* Σ (Peso**i** \* Valor**i**)

**EF** = 1.4 - 0.03 \* **19**

**EF** = **0.83**

**Luego: UCP=UUCP \* TCF \* EF**

**UCP = 45 \* 1.05 \* 0.83**

**UCP = 39.21**

**Estimación de esfuerzo a través de los Puntos de Historias de Usuarios**.

**Ecuación 3.5**: Esfuerzo estimado en horas hombres.

E = UCP \* CF

Donde:

**E:** Esfuerzo estimado en horas hombres.

**UCP**: Punto de historias de usuarios ajustadas.

**CF**: Factor de conversión.

Para obtener el factor de conversión (CF) se cuentan cuántos valores de los que afectan el factor ambiente (E1 a E6) están por debajo de la media (**<3**), y los que están por encima (**>3**) para los restantes (E7 a E8). Si el total (nos da **0**) es 2 o menos se utiliza el factor de conversión 20 Horas- Hombre / Punto de historias de usuarios. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de historias de usuarios. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso es demasiado alto. En este caso:

CF= 28 Horas-hombre / Puntos de historias de usuarios.

Luego

**E = 39.21 \* 28 horas-hombre**

**E = 1097.88 horas-hombre**

En la siguiente [tabla 3.5](#_Tabla_3.5_Distribución) se muestra distribución del esfuerzo por etapas.

#### Tabla 3.5 Distribución del esfuerzo por etapas

|  |  |  |
| --- | --- | --- |
| **Actividad** | **% esfuerzo** | **Valor esfuerzo** |
| Planificación | 10 | 109.788 |
| Diseño | 20 | 219.576 |
| Codificación | 40 | 439.152 |
| Prueba | 15 | 164.682 |
| Sobrecarga | 15 | 164.682 |
| Total | 100 | **1097.88** |

Una vez estimado el tiempo de desarrollo del proyecto y conociendo la cantidad de desarrolladores y el pago que recibe cada uno de estos se puede llevar a cabo una estimación del costo total del proyecto referidos a los recursos humanos; existen otros costos como por ejemplo del equipamiento que se suman al anterior.

**K**: Coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0).

**THP:** Tarifa Horaria Promedio. El salario promedio mensual de los trabajadores en este caso es de $12 000 CUP dividido entre 176h.

**176 horas** (horas de trabajo para 1 mes, esto se toma a razón de 24 días, ya que no se cuentan los fines de semana ni sábados cortos).

**Tiempo**= 1097.88 horas/176 ≈ equivalente a 7 meses, este es el tiempo que tomaría desarrollar el proyecto empleando una sola persona.

7 meses a razón de 24 días laborables por mes, representan 168 días. En nuestro caso como empleamos 2 trabajadores para el desarrollo del proyecto, el tiempo para su culminación quedará reducido a 84 días aproximadamente.

**Entonces el costo total del proyecto:**

**C = E (Total) \* K \* THP**

**C** = 1097.88 \* 2.0 \* 12 000/176 = **$** **149 709.917**

Para el salario a los trabajadores se investigó como se mueve en diferentes organizaciones o sucursales en Santiago de Cuba de diferente sector ya sea estatal o privado, dada las nuevas regulaciones y tasas de cambio. Los datos se obtuvieron a través de trabajadores de las entidades y por anuncios laborales.

Tabla 3.6 Situación actual de pago.

|  |  |
| --- | --- |
| **Organización (Sector)** | **Pago/mensual (Moneda Nacional)** |
| XETID (Estatal) | 10 000 |
| DATYS (Estatal) | 9 600 |
| DESOFT (Estatal) | 10 000 |
| MYPIMES (Privado) | 20 000 |
| Freelancer o persona autónoma (Privado) | 200MLC a 210 CUP (tasa oficial actual) son 42 000 |

## **3.4 Pruebas al sistema**

**Referencias Bibliográficas**

1. Cornel Law School. 2023 Contratos Comerciales <https://www.law.cornell.edu/wex/es/contratos_comerciales>
2. Sordo, Ana Isabel. Febrero, 2021. Metodología Scrum. [http://blog.hubspot.es](http://blog.hubspot.es/) Citado [11-05-2022].
3. Martin, Pedro. 2021. Blockchain ¿Por qué y cómo surge? [https://visualeo.com/blockchian-por-que-y-como-surge/#](https://visualeo.com/blockchian-por-que-y-como-surge/)citado[02-05- 2022]
4. Universidad de Alcalá. Junio 2017. Historia de los Smart Contract. Citado[02-05-2022]
5. 2022. Sistema de gestión de *Smart Contract.* <https://contratosinteligentes.net/> citado [5-10-2022]
6. Bit4you. 2022. <https://www.bit4you.io/> citado [23-10-2022]
7. De la Fuente, Judith. 6-6-2021. Que es open source.
8. <https://eossolution.io/> citado[11-1-2023]
9. Colectivo de autores. Nov-2019. SmartChain.io Easy Blockchain for your business. <http://www.smartchain.io/sites>
10. 2021. Next.js\_ ¿qué es y por qué debes usarlo? <https://www.aplyca.com/blog/nextjs-el-futuro-web-que-es-nextjs> citado [1-2-2023]
11. OpenWebinars. 2023. Qué es Visual Studio Code y qué ventajas ofrece. <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece> citado [1-2-2023]
12. Fernandez, Yubal. Abril, 2022. Solidity: que es y para qué sirve este lenguaje de programación. <http://www.xataka.com/basics/solidity-que-sirve-este-lenguaje-programacion/amp> citado [1-2-2023]
13. Chacón, José Luis. 2022. TypeScript: qué es, diferencias con JavaScript y por qué aprenderlo. <https://profile.es/blog/que-es-typescript-vs-javascript/>
14. Microsoft. Septiembre, 2022. Que es Git. <https://learn.microsoft.com/es-es/devops/develop/git/what-is-git> citado [9-04-2023].
15. Sinnaps. 2020. Metodología XP o Programación Extrema <https://www.sinnaps.com/> citado [2-2-2023].
16. Puntos de caso de uso. (2021). En *Wikipedia, la enciclopedia libre*. <https://es.wikipedia.org/w/index.php?title=Puntos_de_caso_de_uso&oldid=133055102> citado [17-04-2023].
17. Gestión, Wikipedia, la enciclopedia libre. mar. 02, 2021. [En línea]. Disponible en: [https://es.wikipedia.org/w/index.php?title=Gesti%C3%B3n&oldid=133648874\](https://es.wikipedia.org/w/index.php?title=Gestión&oldid=133648874\) citado [4-05-2023]
18. Sistemas de Gestión Normalizados | THINK&SELL. https://thinkandsell.com/servicios/consultoria/software-y-sistemas/sistemas-de-gestion-normalizados/ citado [4-05-2023]
19. ConsenSys Software. 2022. <http://www.trufllesuite.com/docs/truffle/over/view> citado [6-05-2023]
20. G. Booch, J. Rumbaugh, I. Jacobson, J. J. G. Molina, y J. S. Martínez. 2006. El lenguaje unificado de modelado: guía del usuario. Addison-Wesley. Citado [6-05-2023]
21. Epitech Spain. 2021. Diagrama de flujo en programación ¿Qué es y para que se usa? <http://www.epitech-it.es/diagrama-de-flujo-programación/> Citado [7-05-2023]
22. Puntos de caso de uso. (2021). En *Wikipedia, la enciclopedia libre*. <https://es.wikipedia.org/w/index.php?title=Puntos_de_caso_de_uso&oldid=133055102> 21-06-2023