UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación de tecnología Blockchain a una cadena de distribución de productos

Presentado por Eduardo Rodríguez Soto
en la Universidad de Burgos — 17 de enero de 2020
Tutor académico: Ángel Arroyo Puente
Tutor empresarial en HP: Pablo Tejedor García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Ángel Arroyo Puente, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Eduardo Rodríguez Soto, con DNI: 71293020-N, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado: Aplicación de tecnología Blockchain a una cadena de distribución de productos.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 17 de enero de 2020

Vº. Bº. del Tutor académico:	Vº. Bº. del tutor empresarial :
D. Ángel Arroyo Puente	D. Pablo Tejedor García

Resumen

Este trabajo ha sido realizado con la Universidad de Burgos y la empresa *Hewlett-Packard HP*, en su sede ubicada en León.

La compañía HP se dedica a la fabricación y comercialización de *hardware* y *software*. Otro servicio destacable que ofrece HP es la asistencia a sus usuarios.

La empresa HP con sede en León tiene como objetivo el desarrollo de software para impresoras de largo formato. Los productos que se investigan y desarrollan en el centro están relacionados con el firmware de las máquinas así como con los programas que permiten el uso de las mismas a aplicaciones y sistemas operativos.

El rango de productos de largo formato va desde impresoras A3 a impresoras 3D pasando por máquinas de producción industrial y máquinas gráficas o diseño (usadas principalmente en estudios de arquitectura e ingeniería).

El principal propósito de este trabajo es el desarrollo de una página web con el fin de que un usuario concreto pueda interactuar con la tecnología blockchain y, así, poder llevar a cabo las diferentes opciones que nos ofrece la página web, como son:

- Crear de usuarios.
- Añadir un producto.
- Consultar todos los productos que tiene dicho cliente.

Este trabajo se ha realizado con la herramienta de trabajo VisualStudio Code, para la implementación de la página web, y con la herramienta Metamask , para la creación de los contratos.

Descriptores

Html, Php, tecnología blockchain, Metamask, Ganache, Remix

Abstract

This work has been carried out with the University of Burgos and the Hewlett-Packard HP company located in the city of León.

HP is engaged in the manufacture and commercialization of hardware and software. Another noteworthy service offered by HP is the assistance to its users.

HP León is dedicated to at developing software for long-format printers. The products that are researched and developed in the center are related to the firmware of the machines as well as to the programs that allow the use of them to applications and operating systems.

The range of long format products ranges from A3 printers to 3D printers to industrial production machines and graphics or design machines (mainly used in architectural and engineering studies).

The main purpose of this work is the development of a web page in order for a specific user to interact with blockchain technology and, thus, be able to carry out the different options offered by the web address, such as:

- Create users.
- Add a product.
- Consult all the products that the customer has.

This work has been done with the VisualStudio Code working tool, for the implementation of the web page, and with the Metamask tool, for the creation of contracts.

Keywords

Html, Php, tecnología blockchain, Metamask, Ganache, Remix

Índice general

Índice de figuras

Índice de tablas

Introducción

La empresa HP con sede en León tiene como objetivo el desarrollo de software para impresoras de largo formato. Los productos que se investigan y desarrollan en este centro están relacionados con el *firmware* de las máquinas así como con los programas que permiten el uso de las mismas a aplicaciones y sistemas operativos.

En los últimos tiempos, HP ha creado una nueva unidad de trabajo a la cual se la denomina grupo de innovación y consiste en explorar nuevas tecnologías. Una de las tecnologías a investigar es la tecnología Blockchain, de la que explicaremos en este documento.

Blockchain [1] surge en 1991 de la mano de *Stuart Haber* y *W. Scott Stornetta*, ambos científicos crearon la tecnología de las criptomonedas más conocida como cadena de bloques (esto permite a cada cliente de la red poder hacer una transferencia a otra persona sin tener que confiar entre sí). Para dar más seguridad, un año después se incorporan los árboles Merkle[1], lo que hace esta tecnología es que sea más eficaz y rugoso ya que permite que varios archivos o documentos se puedan juntar todos en un mismo bloque. En el año 2004, se introducen los sistemas RPoW (Prueba de trabajo Reutlizable), que resuelve el problema de doble gasto, en el 2008 se crea la red Bitcoin y en 2013 nace la red Ethereum.

Uno de los objetivos del presente proyecto es la creación de una página web, esta página la realizaremos en un servidor local llamado XAMPP, primeramente tendremos que instalar XAMPP (se explicará en el apartado de los anexos) esto será nuestra base de datos, con la cual guardaremos a los

[1] Árbol de *Merkle*: es una estructura de árbol, en el que cada nodo que no es una hoja le corresponde una etiqueta hash de la concatenación de las etiquetas de los nodos hijo.

usuarios que se registren desde la página, desde la página tendremos opción de modificar y borrar los usuarios.

Cuando el usuario se haya registrado, tendremos la posibilidad de realizar los smart contract, es un contrato que es capaz de ejecutarse y hacerse cumplir por sí mismo, de manera autónoma y automática, sin intermediarios ni mediadores, que conectará a un sistema de red privada llamada Rospten, y le pasaremos diferentes parámetros, una vez creado no se podrán ni modificar ni destruir.

La realización de todas las pruebas ha ido cambiando desde la creación de diferentes *smart contract* tanto en la red local (Ganache), en cuentas privadas (Ropsten) e incluso en la red principal de Ethereum. La realización de los *smart contract* es mediante metamask, web3, node.js, solidity y más aplicaciones que explicaremos más detalladamente en los siguientes puntos.

Estructura de la memoria

Esta memoria constará de los contenidos explicados a continuación:

1) **Resumen**: breve introducción de los objetivos y descripción de la empresa con la cual realizo dicho trabajo. Este apartado estará tanto en castellano como en inglés.

2) **Objetivos del proyecto**: se detallarán los objetivos qeu se quieren conseguir al realizar este proyecto.

3) **Conceptos teóricos**: realizamos un pequeño resumen de los conceptos que hemos aprendido al realizar dicho proyecto.

4) **Técnicas y herramientas**: descripción de los programas utilizados para la elaboración y seguimiento del proyecto.

5) **Aspectos relevantes del desarrollo del proyecto**: aclaración de las herramientas utilizadas y argumentación final de nuestro proyecto.

6) **Trabajos relacionados**: consiste en hablar de los trabajos similares a nuestro trabajo, tanto realizados por la universidad de Burgos o ajenos a ella.

7) **Conclusiones y líneas de trabajo futuras**: valoración de todo lo conseguido y posibles mejoras que se podrán mejorar en futuros trabajos relacionados con este proyecto.

8) **Bibliografía**: información encontrada en Internet, a la que se hace referencia en cada apartado usado en la memoria del proyecto.

Estructura de los anexos

Los anexos se estructuran de la siguiente manera:

1) **Plan de proyecto de software**: estudio de la viabilidad tanto legal como económica del proyecto, también comentaremos los objetivos propuestos en cada reunión.

2) **Especificación de usuarios**: en este apartado explicaremos los casos de uso y requisitos funcionales de la herramienta creada.

3) **Especificación de diseño**: se detallaran los diseños software utilizados en este proyecto.

4) **Documentación técnica de programación**: explicación de la instalación del programa y guia del código fuente mas relevante.

5) **Manual de usuario**: guía para el manejo de la aplicación para cualquier usuario, requisitos e instalación.

Contenido entregable en el CD

Los documentos que contiene el CD serán:

1. Documentación de la memoria: versión en *pdf* de la memoria final.
2. Documentación de los anexos: versión en *pdf* de la anexos finales.
3. Código del programa: version final del código ejecutable para la puesta en marcha del trabajo.
4. Vídeo explicativo de las funcionalidades de la página web.

Objetivos del proyecto

El objetivo de este trabajo es la creación de una página web y la investigación es indagar sobre las posibilidades de la tecnología blockchain y su posible aplicación en las unidades de negocio de HP.

Los trabajos de observatorio son gestionados por una nueva unidad de trabajo de HP León denominada "grupo de innovación" cuya misión es explorar nuevas tecnologías que permitan mejorar los procesos de desarrollo o ampliar los campos de negocio de la compañía.

2.1. Objetivos Técnicos y Generales

1 Explorar las posibilidades de la tecnología blockchain.

2 Explorar las posibilidades y capacidades de la red *ethereum* y *solidity* para desarrollar smart contracts.

3 Ampliar conocimientos del alumno en la tecnología blockchain.

4 Creación de la página web, con el fin de conseguir la conexión entre la plataforma Ethereum y nuestra página mediante el uso de Metamask.

5 Familiarizar al alumno con las metodologías ágiles como *SCRUM*.

6 Permitir al alumno conocer los métodos de trabajo de la compañía HP.

7 Familiarizar al alumno con las metodologías ágiles como *SCRUM*.

Conceptos teóricos *Scrum*

Después de analizar diferentes fuentes bibliográfica, se ha decidido tomar como referencia el libro *[Scrum y xp desde las trincheras]*"[Henrik Kniberg, 2018]" [2] como principal fuente de información.

3.1. Introducción

Scrum es una herramienta ágil de trabajo creada por Ken Schwaber, quien defendió que Scrum no era una metodología de trabajo, sino que más bien era un marco de trabajo. Es decir, este sistema explicaba los pasos que se deben seguir para realizar un trabajo con éxito. El autor esclareció que Scrum era un vía a la que adaptarse según la situación concreta en la que se fuera a trabajar, puesto que cada proyecto puede contar con particularidades. De todas formas, el objetivo del libro es desarrollar una forma de trabajo más eficiente y productiva.

En los siguientes apartados pretendemos, analizar el libro, todo ellos desde la visión de la persona que escribió el libro, el sueco Henrik Kniberg.

3.2. Cómo hacemos las pilas de producto

La lista de producto es la parte más importante del *Scrum*. Aquí comienza a funcionar todo el proyecto, en esta lista se incluyen los principales objetivos de los requisitos que se quieren conseguir al final del proyecto, a esto lo llama elementos de la pila o historias.

Elementos de la pila:

- ID: es el identificador único, contador de los elementos o historias.

- Nombre: breve explicación del problema a tratar será clara y concisa.
- Importancia: ratio de consideración que el dueño dé a cada elemento.
- Estimación inicial: valoración del equipo de cuánto trabajo es necesario para cada tarea.
- Como probarlo: realización de pruebas o *demos* en el *sprint* final, ej. realización de pruebas o test.
- Notas: información que pueda considerarse aclaratoria, o que haga

Estos seis campos mencionados son aquellos que más importancia se les da a la hora de realizar los *sprint*. Para la realización de estos campos en los diferentes trabajos académicos es importante realizar los trabajos en tablas Excel, las cuales podrían ver simultáneamente tanto el dueño como los múltiples usuarios.

De la misma manera, el autor menciona que se pueden usar unos elementos adicionales que pueden mejorar la accesibilidad por parte del dueño y así controlarlo mejor las propiedades de cada apartado. Estos campos son: **la categoría** explicación de la historia básica, **componentes** realización de tickets para comprobar si los procesos se cumplen, **solicitante** indicar los avances del producto y **bug tracking id** seguimiento de errores reportados.

3.3. Cómo nos preparamos y planificamos un *sprint*

Llegados a este punto las preparaciones de los *sprint* se aproximan y lo más importante es tener la pila del producto correctamente lista antes de la planificación del *sprint*. Esto quiere decir que es necesario intentar conseguir unos objetivos antes de empezar la planificación:

- Deberá existir la pila del producto.
- Habrá una pila de producto y un dueño de producto.
- Tendrá los ratios de importancia bien definidos (independientemente de que puedan coincidir algunos de los ratios para las tareas menos importantes).
- El dueño del producto es el que dará importancia de cada cosa.

Una vez concluida la preparación, es momento de desarrollar la planificación, que consiste en realizar una reunión, la que se convertirá en la más decisiva del *scrum*. Si la reunión fuera mal ejecutada podría arruinar por completo el *sprint*, el objetivo de esta planificación de *sprint* es dar al equipo la información necesaria para poder trabajar durante las siguientes semanas.

Los objetivos de la planificación del *sprint* son: establecer la meta, elegir a los miembros con la dedicación que tendrán y realizar la pila del *sprint* (incluyendo las historias). También es adecuado elegir el lugar y la fecha para la reuniones de las celebraciones de los *sprint* diario.

El dueño del producto, por regla, no suele asistir a las reuniones y no trabaja con el equipo, aunque, es recomendable que asista ya que pueden darse tres variables interdependientes, como muestra la figura 3.1.

El alcance y la importancia son fijados por el dueño de producto. La estimación la proporciona el equipo de trabajo.

Cuando se planifica el *sprint*, estas variables sufren un desajuste continuo con cada dialogo que se produce cara a cara entre el equipo y el dueño del producto.

Figura 3.1: Triangulo de variables

En el triángulo antes expuesto faltaría un último elemento que sería la calidad. Esta se subdivide en dos categorías: calidad externa e interna, la primera la perciben los usuarios del sistema y se trata de una interfaz lenta y poco intuitiva, por otro lado, dentro de la calidad interna se encuentran los aspectos no visible al usuario, pero con gran efecto en la mantenibilidad del sistema, como consistencia, refactorización. La segunda es algo alcanzable y la calidad interna es algo que no puede ser discutido.

En las reuniones, el dueño de producto explica cuáles son los objetivos del *sprint* y las historias más importantes a conseguir. El equipo las repasa y las asigna un tiempo estimado de ejecución.

Si el dueño no se presentara a las reuniones, se propondría a alguien del grupo para que desempeñara las funciones de delegado durante las reuniones. Si no surgiera un sustituto, se intentaría asignar un nuevo dueño del proyecto y, por último, se pospondrían los lanzamientos del *sprint* hasta que el dueño esté disponible y pueda asistir a las reuniones.

Reuniones de planificación de *Sprint* que duran

Los *scrum* tienen una duración determinada (*time-boxed*). Cuando una reunión de planificación de *sprint* está llegando al final y aún no están todos los objetivos cumplidos, ¿qué se aconseja hacer? se recomienda la opción de dar la reunión por finalizada. La reunión perdería parte de su utilidad y con esto conseguimos que en la siguiente reunión los *sprint* sean más eficientes y se ofrezca menos resistencia cuando se proponga una duración excesiva.

Se recomienda que las reuniones sean lo más ágiles posible, aunque los sprint largos pueden resultar beneficiosos ya que prestan más tiempo a la detección de errores. Aproximadamente tienen una duración de tres semanas.

¿Qué historias incluir en un *Sprint*?

Al principio de las reuniones se intentan poner unos objetivos en común (historias) con la importancia de cada una de ellas; estas se pondrán en orden de mayor a menor. El grupo de trabajo intentará agrupar el número de historias convenientes según ellos estimen que serán capaces de realizar en la duración de un*sprint*.

Tendremos la lista de las tareas por realizar en la pila del producto y serán incluidas en la pila del *Sprint*, como se puede ver en la siguiente fotografía.

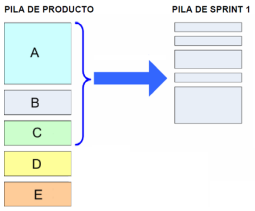

Figura 3.2: Pasos de un *sprint*

En el supuesto de que se quieren meter las tareas A, B y C, pero el dueño del producto quiera introducir la tarea D, una de las posibles combinaciones a realizar es: reorganizar la lista y dejar alguna de las tareas que tenemos fuera. Por ejemplo, nos quedaría A, B y D y se dejaría la tarea C fuera de la pila.

También se podría realizar el reparto de otras formas.

El equipo decide qué historias incluir mediante dos cálculos:

1. Como se dice tradicionalmente,"a ojo de buen cubero": suele darse para equipos pequeños y con *sprint* cortos, se calculan suponiendo la dificultad de cada uno de ellos.

2. Cálculos de velocidad:

   a) Decir la velocidad estimada: medida de cantidad de trabajo realizado donde se evalúa en función de la estimación inicial

   b) Calculando las historias que se pueden añadir sin sobrepasar las velocidades: la velocidad es una medida de cantidad de trabajo realizado, en la cual cada elemento se valora por la estimación que se hizo inicialmente. Una forma de realizar los cálculos de los recursos será mediante fórmulas, las cuáles son:

      Días – hombre disponible : Sumaremos los días que emplearan todos los trabajadores del proyecto (días/hombres disponibles).

      1) Velocidad estimada: es igual a la disponibilidad de hombres y días junto al factor de dedicación, que depende de la concentración del equipo. Para calcular un buen factor estudiaremos cómo fueron los anteriores sprint.
      2) Factor de dedicación del ultimo *sprint*: es igual a la velocidad real repartida entre los días y hombres disponibles.

   c) Uso de tarjetas: con este método se logra que todos los integrantes del grupo sean participes y así dar el tiempo de realización de cada actividad.

   d) Una vez que el equipo tiene definidas todas las tareas, estas se pasarán a una hoja de cálculo para ver el avance de todo ello.

3. Definición de terminado

   El primer caso que debe darse para que se considere como acabado es que el equipo este de acuerdo con esta decisión. Una historia se considera terminada cuando el encargado del *scrum* así lo determina.

4. Estimación de tiempos con *PLANNING POKER*

   En este tipo de estimación todos los miembros del equipo deben involucrarse para estimar cada historia ya que no sabemos quién implementará cada historia o si esa tarea la realizaran personas de diferentes áreas.

   Para ello existe el *planning Poker*, diseñado por Mike Cohn. Esto consiste en cada miembro del equipo cuenta con unas cartas y cada vez que sale una historia los trabajadores tendrán que levantar una carta que indica la estimación del tiempo que necesitaran para realizar ese trabajo. Si los tiempos propuestos son muy diferentes, aquellos que hayan seleccionado las cartas más dispares explicarán sus motivos. Tras ello, se procede a una segunda votación.

Hay dos cartas que son totalmente diferentes a números e indican: "?" se desconoce la cantidad de tiempo que puede llevar el desempeño de esta tarea, "0" esta historia ya está realizada o apenas lleva tiempo completarla y "Taza de café" se solicita un momento descanso.

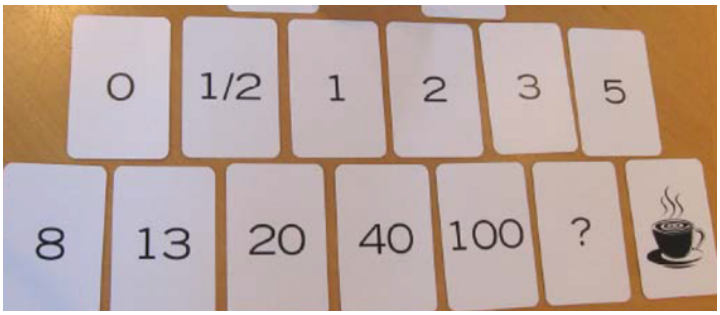

Figura 3.3: Planning poker

5. Diferencia entre historia y tarea

   La diferencia es simple, la primera es aquello que se le entrega al dueño del producto; la segunda, los archivos no entregables o aspectos que no preocupan al dueño del proyecto.

6. Definir sitio y hora para el *scrum* diario

   El primer *scrum* es el lanzamiento, cuando decide el personal por dónde empezará a trabajar. Existen dos formas de reuniones:

   Por la mañana, consiste en recordar lo que se realizo el día anterior para ponerlo en común y acordar la tarea a realizar hoy y por la tarde, consiste en recordar lo que hemos realizaste durante la jornada de trabajo y hablar sobre lo que realizaras el día de mañana.

7. Historias no funcionales

   Son elementos no funcionales, es decir, son los archivos o elementos no entregables o que no están relacionadas con ninguna historia específica y no tienen mucho valor para el dueño del producto.

   Por ejemplo: escribir una descripción del diseño, refactorizar los accesos a los datos o tener un seguimiento de los errores o programa encargado de ello.

8. Seguimiento de errores vs. pila del producto

Hay diferentes programas para el seguimiento de errores. En el libro recomiendan (*Jira*), ya que Excel es un formato bueno para la pila de producto, pero no es tan específico para los errores.

9. Finaliza la planificación del *sprint*

   La reunión llega a su fin cuando el dueño del producto y los miembros del equipo llegan a consenso. Una vez que están todos de de acuerdo, el siguiente paso es comenzar a trabajar con un *scrum* diariamente durante las siguientes semanas de trabajo.

3.4. Cómo comunicamos los *sprint*

La comunicación entre la plantilla de trabajo es crucial a la hora de informar sobre lo que está ocurriendo en cada jornada de trabajo. Para informar en todo momento de lo que está sucediendo, en el libro indican que puede ser beneficioso crear una página de información del *sprint*, donde se indiquen los objetivos, pasos a cumplir y fechas de entrega. El jefe del *scrum* podrá imprimir la hoja y dejarla en un lugar visible para que todos los miembros puedan saber dónde está el *sprint* actualmente.

3.5. Cómo hacemos las pilas de *sprint*

Una vez haya concluido la reunión de planificación del *sprint* y se haya informado a los trabajadores de ello, el jefe del proyecto (*scrum Máster*) debe crear una pila de *sprint*. Esto se hace antes de la reunión de planificación de *sprint*, pero antes del primer *scrum* diario.

El formato de la pila será una lista de tareas, como la imagen que se muestra a continuación:

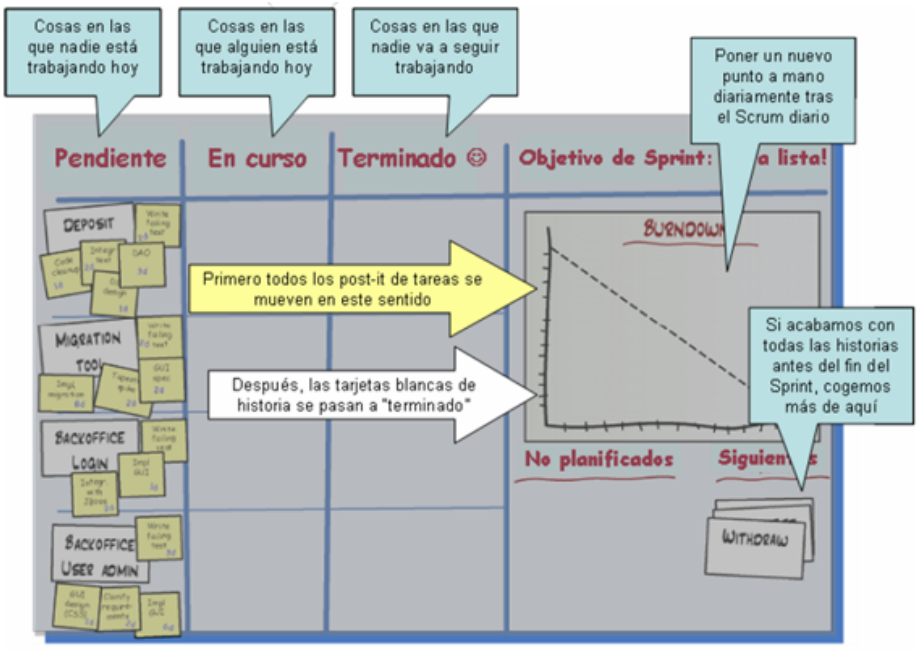

Figura 3.4: Tablón de tareas

En dicha tabla se podrán añadir todas las columnas que se deseen, por ejemplo: Test funcionando, proceso cancelado, etc.

El diagrama *burn-down* indicara cómo va el proyecto con respecto al tiempo empleado. (trabajo restante/ fecha estimada).

El gráfico *burn-down* puede indicar varios síntomas:

- Alarmante: cuando la línea está por encima de la línea de puntos, se considerará quitar elementos de la pila del *sprint*.
- Buen camino: cuando la línea está por debajo de la línea de puntos, se considerará añadir nuevas historias al *sprint*.

El *Scrum Máster* es el responsable de asegurarse de que el equipo actúa ante diferentes señales como que se acumulen las actividades en la tarea pendiente o el gráfico *burn-down* no siga una tendencia recta.

3.6. Cómo distribuimos la sala del equipo

Los miembros se reúnen en frente de una pizarra para que se puedan observar entre ellos mientras exponen ideas de cómo abordar los diferentes

errores o ideas que puedan surgir durante el devenir del proyecto.

También es importante que los miembros del equipo se sienten juntos o cercanos en el área de trabajo para fomentar la comunicación entre ellos.

Ventajas de trabajar juntos en la oficina resultados inmediatos, audibilidad posibilidad de hablar con cualquier miembro del equipo sin levantarse de la silla, visibilidad todos los trabajadores pueden ver lo que sucede y tener cercano el tablón de tareas y aislamiento mayor orden a la hora de reunir al equipo y no tener que levantarse y molestar a molestar al resto de compañeros de la oficina.

Si el equipo está distribuido, se intentará que este lo más cercano posible usando técnicas como videoconferencia, *webcams*, maquinas remotas de escritorio, etc.

Sería recomendable que el dueño del producto esté cerca para consultar dudas y el tablón de tareas, pero no lo suficientemente cerca de estar al lado del equipo, ya que puede que no sea capaz de controlarse y pueda interrumpir en cualquier detalle y provocar que el equipo no fluya adecuadamente.

3.7. Cómo hacemos los *scrums* diarios

Deben ser realizados todos los días, a la misma hora y en el mismo lugar. Se suelen realizar de pie para intentar que no sobrepasen los 15 minutos de duración. Una vez realizado el *scrum*, actualizamos el tablón de tareas de acuerdo con las indicaciones de los trabajadores sobre lo que realizaron el día anterior y los planes que tienen para ese día. Lo puede realizar el *Scrum Máster* y así, mientras los trabajadores lo indican, él va modificando los *post-it* en el tablón, para incentivar que la gente no llegue tarde, en el libro recomienda el uso de pequeñas multas para que no haya informales en el grupo de trabajo.

3.8. Cómo hacemos la *demo* de *sprint*

Tener que hacer una demostración con cada final del *sprint* obliga al equipo a poder demostrar el trabajo realizado durante ese proceso, esto es favorable ya que consigue un *feedback* entre las partes interesadas en el proyecto. Con esto se intenta conseguir un reconocimiento a los trabajadores y las personas que no son parte del proyecto sea capaz de entender lo que realiza el equipo si las demostraciones son claras y descriptivas, estaremos consiguiendo un ahorro de trabajo para futuro trabajo.

3.9. Cómo hacemos las retrospectivas de *sprint*

Las retrospectivas son importantes, en vista de que con ello conseguimos y nos aseguramos de que son realizadas por el equipo. Permite, además, mejorar ideas o alguna cosa que el grupo vio mal en el anterior *sprint*, con el objetivo de que no se vuelva a repetir de nuevo en los posteriores *sprint*.

El autor indica un ejemplo para realizar las retrospectivas:

- Reservar de una a tres horas, dependiendo la discusión.
- Los representantes: el dueño del producto y el equipo.
- Intentar que sea una reunión cómoda y sin gente interrumpiendo.
- Designar un secretario.
- Intentar que participe todo el mundo, opiniones, errores e ideas sobre el próximo *sprint*.
- Observar los tiempos reales con los que teníamos estimados.

Se puede escribir sobre una pizarra las retrospectivas con tres columnas: **buena**, si volviéramos a realizar esta tarea, realizaríamos los mismos pasos, **mejorable**, ¿qué cambiaríamos si lo volveríamos a realizar las mismas personas? y **mejoras**, ideas concretas sobre qué mejorar en el futuro.

Se podría decir que *bien* y *mejorable* son miradas al pasado y *mejoras*, miradas al futuro.

3.10. Descansos entre *sprint*

Como indica el libro, no se puede estar siempre esprintando ya que acabarías derrotado y al final estarías haciendo *footing*.

Los *sprint* son muy intensos y por ello hay que dejar espacio entre ellos. Con esto se consigue liberar la cabeza y, *a posteriori*, aportar mejores ideas.

Es conveniente que la retrospectiva y la siguiente reunión no tengan lugar en el mismo día y es incluso aconsejable realizarla con el fin de semana de por medio.

Un de las mejores ideas que propone es tener un día de laboratorio (que los trabajadores puedan tener momentos largos de libertad) para poder realizar otras cosas relacionadas con el trabajo, pero no con ese trabajo, esta idea está inspirada en Google.

Lo más importante es que entre *sprint* y *sprint* haya horas de desconexión y te permitan dormir para relajarte.

3.11. Planificación de entregas y los contratos de precio fijo

En algunas ocasiones, cuando el contrato es de precio cerrado, necesitamos planificar los *sprint* con antelación. Por ello el propietario necesita las estimaciones aproximadas de las tareas que se incluirán en el contrato, estas tareas las deberá haber realizado con antelación el equipo de trabajo.

Definir los umbrales de aceptación y estimar elementos importantes

Los umbrales son una clasificación según la importancia de la pila de producto de acuerdo con los términos que aparezcan en el contrato. Conforme con dónde esté el umbral de aceptación, meteremos esto en las diferentes versiones de nuestro proyecto.

Para estimar los elementos más importantes, se deben incluir todas las historias que se añaden en el contrato. Las estimaciones las tendrá que realizar el equipo y no tendrá que dedicarse demasiado tiempo para ello, son estimaciones no compromisos.

Figura 3.5: Estimación de los elementos

Después de realizar cada *sprint* comprobaremos la velocidad real del *sprint* y, en caso de que sea muy diferente a la que se estimó en un principio, revisaremos la velocidad que dijimos en el resto del *sprint* y tendremos que actualizar el plan de entregas. Quizás la modificación de las entregas no guste al cliente, pero lo que se intenta es cumplir el objetivo y ser honestos con él.

3.12. Cómo combinamos *Scrum* con XP

Hay que indicar que tanto *scrum* como XP (Programación eXtrema) se combinan de forma perfecta, dado que *scrum* se centra en prácticas de organización y gestión, mientras que XP se centra en prácticas de programación. Ya que tratan áreas diferentes pueden complementarse muy bien entre ellas.

Los beneficios de la programación por parejas son:

Mejorar la calidad del código, la concentración, la distribución del conocimiento entre el equipo, poder cambiar de parejas para aportar nuevas ideas, poder estandarizar el código(esto puede ser útil si se cambia de parejas).

El desarrollo guiado por pruebas TDD es un test automático, en el cual tú escribes el código necesario para pasar ese test y a continuación refactorizas el cogido para su legibilidad y eliminar posibles duplicaciones.

TDD es duro, puede llevar tiempo comprenderlo, pero es muy importante, tiene efecto positivo en el diseño del sistema además tiene pruebas de integración tipo “caja negra” y permite utilizarlo con otros programas como *Jetty*, Cobertura (métricas).

Una parte importante es el diseño incremental. Consiste en mantener el diseño simple desde el momento que empezamos el proyecto e ir mejorándolo continuamente, posteriormente se podrá limpiar el código y revisarlo. Primeramente, es necesario que funcione de forma correcta, las mejoras del diseño son efectos del realizar el TDD.

Es importante la integración continua, con esto evitamos el problema de que el proyecto funcione en todas las maquinas. Una vez compilado en un ordenador, realizamos la copia en otro servidor y este se encarga de comprobar el funcionamiento y correr con todas las pruebas. El entorno que recomiendan es el basado en *Maven* y *QuickBuild*.

3.13. Cómo hacemos pruebas

Posiblemente es la parte más dura del *scrum* ya que la realización de estas puede resultar la parte más variable entre las diferentes organizaciones.

Nada está acabado hasta que el equipo de pruebas lo da por finalizado. Si por un casual el equipo destinado a las pruebas no está trabajando en un momento dado, lo óptimo sería que vaya preparando las pruebas para así adelantar trabajo.

Figura 3.6: Ejecución del orden de las pruebas

Para minimizar la fase de pruebas se puede reducir la cantidad de tiempo que se dedica a este proceso. Esto se consigue de dos formas: maximizando la calidad del código desarrollado por el equipo o maximizando la eficiencia del trabajo manual de pruebas (*beta-tester* y dándoles mejores herramientas).

Para incrementar la calidad se puede incluir a un miembro del equipo de las pruebas en el equipo *scrum*, con esto podríamos cumplir los tiempos ya que los *sprint* tienen un tiempo limitado. De esta manera, se podría optimizar el tiempo y conseguir los objetivos.

Si todo fuera perfecto, veríamos innecesario el papel del equipo de pruebas ya que cada equipo *scrum* entregaría la versión directa para salir al mercado tras cada *sprint*. Pero esto no suele acontecer así ya que al finalizar el *sprint* uno y pasar al dos, seguramente nos encontremos con algún error en el código. Por esa razón los equipos de prueba son tan necesarios.

3.14. Cómo manejar múltiples equipos *Scrum*

Cuando tienes varios equipos trabajando en un mismo proyecto, la cosa se puede complicar y esto te genera una serie de preguntas: cuántos equipos crear y cómo distribuirlos.

Cuanta más gente en el mismo *scrum* tiende a ser un conflicto mayor ya que los *scrum* diarios tienden a durar más que los 15 minutos que se estipularon en un principio y puede que los miembros del equipo no sepan lo que hacen otros, y esto puede crear confusión.

Lo aconsejable en este caso es dividir los equipos en equipos entre 5 y 9 personas.

¿Los *sprint* deben ser sincronizados o no?

Suponiendo que tenemos a más de un equipo *scrum* trabajando en el mismo producto, nos plantemos la idea de que si se tendría que tener los *sprint* sincronizados. Esto quiere decir, ¿se tendrían que empezar y acabar al mismo tiempo o, por el contrario, deberían solaparse?

Si lo realizábamos de forma solapada siempre tendríamos un *sprint* cercano a finalizar y otro recién comenzado, lo que implicaría que habría entregas constantes y la carga del dueño del producto podría distribuirse en el tiempo de forma equitativa (Ilustración 3.10).

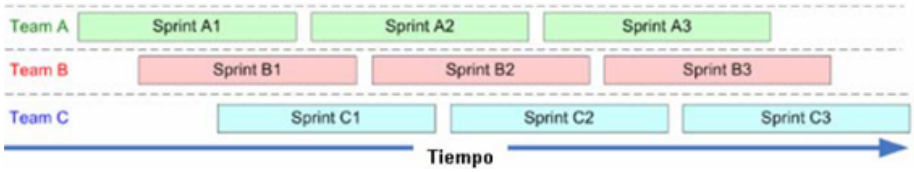

Figura 3.7: Tiempo solapado

Pero viendo los *sprint* sincronizados el autor del libro se dio cuenta que todos los equipos podrían trabajar en el mismo *sprint* y realizar las reuniones correspondientes para una mejor planificación; se lograría también se lograría una menor sobrecarga y se realizarían menos reuniones de planificaciones y *demos*.

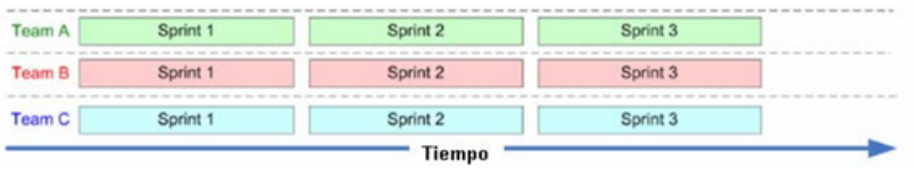

Figura 3.8: Tiempo sincronizado

3.15. Asignación y distribución de las personas en los equipos

Hay dos estrategias para la asignación de las personas a los equipos cuando hay varios para el mismo producto. Los podrá crear el dueño o bien los trabajadores del proyecto. Lo óptimo sería crear equipos multifuncionales, es decir, que sepan desenvolverse en varios ámbitos. Por otra parte, no es recomendable tener miembros en el equipo a tiempo parcial.

También se realizan *scrum* de *scrums* donde los jefes de cada *scrum* se reúnen periódicamente para y analizar cómo va el avance, aclarar las posibles dudas que puedan surgir y estudiar los siguientes puntos a tratar antes de celebrar la próxima reunión.

En cuanto a la distribución siempre intentaremos acercar el ancho de banda de la comunicación entre los miembros del equipo separados. Se intenta programar por parejas, así conseguimos que las personas socialicen, intentar realizar el *scrum* diario y intentar que tengan la misma versión de la pila, para tener los datos sincronizados.

Los métodos más comunes para la comunciación son webcam, salas de runiones, salaz zoom, programas de intercambio express, etc

Offshoring

Al tener varios equipos en diferentes localizaciones, se recomiendan tres opciones para intentar gestionar la situación eficientemente utilizando *scrum*.

1. Equipos separados: los miembros del equipo no-equipo conjunto de *scrum* trabajan independientemente de la localización y simplemente intentaran sacar la producción de cada uno de los objetivos que se van marcando.

2. Miembros de equipos separados: se intenta que se conozcan entre sí todos los miembros del equipo y también tener un método de comunicación adecuado entre las dos localizaciones. Se persigue que los equipos sean pequeños para intentar formar un solo equipo *scrum* (independientemente de las localizaciones).

3. También se puede dar el caso que haya gente trabajando desde casa por diferentes motivos, este es el caso mas positivo ya que están cerca de la oficina y se puede intentar llegar a acuerdos para que algún día trabaje desde la oficina.

3.16. Lista de comprobación del *Scrum Máster*

En este último capítulo se realizan las comprobaciones de las rutinas de los *Scrum Máster* se enumerarán todas las cosas que son susceptibles a olvidar.

Comienzo del *sprint*: se realiza una planificación del *sprint*, se anuncia mediante correo electrónico el comienzo de un nuevo *sprint* y se actualizar el documento de estadísticas donde se añaden las velocidades estimadas.

Todos los días: nos aseguramos en el comienzo del *sprint* diario que se incluyen todas las historias de la pila, notificaremos al dueño del producto todos los cambios que se comentan, mantenemos la pila y el *burn-down* siempre actualizados y intentaremos solucionar todo tipo de problemas y si no, informar al dueño o al jefe de desarrollo.

Final del *sprint*: realizamos la*demo* de *sprint*, informamos a toda la gente de la realización de la *demo* con antelación, realizamos una retrospectiva de *sprint* con el equipo y el dueño del producto y actualizamos el cuaderno de estadísticas de *sprint*, en el cual incluimos la velocidad real y los puntos clave de la retrospectiva

Conceptos teóricos *Blockchain*

Una vez acabado el libro, "*Scrum y XP desde las trincheras*", el siguiente paso en la realización del proyecto es el estudio y la creación de la tecnología *blockchain*. Estuvimos observando los puntos más importantes para empezar con el guion e intentar realizar una web que fuera útil para nuestro proyecto.

A continuación, hablaremos sobre dicha tecnología:

3.1. Origen del *blockchain*

En 1997, hace ahora 21 años, Adam Back inventó *hashcash*, cuyo primer uso era para combatir el correo basura o *spam*, pero actualmente se conoce más por el uso en Bitcoin [3] [4].

Hashcash [5] es una tecnología de prueba de trabajo o *proof of work (POW)* cuya estrategia es establecer un algoritmo que requiere un trabajo de computación para demostrar la verificación de quién manda el mensaje. Tiene mucho interés ya que, para mandar mensajes, tienes que estar dispuesto a realizar un pago por el medio de trabajo de la *CPU*(Unidad Central de Procesamiento), de esta forma si quieres mandar grandes cantidades de correo basura, tendrían que dedicar mucho tiempo en el en el rendimiento de la *CPU*.

En octubre de 2008 Satoshi Nakamoto publicó Bitcoin P2P *e-cash*, un sistema *peer-to-peer* de dinero electrónico sin intermediarios financieros ni de ningún tipo.

El 3 de enero de 2009 tuvo lugar el primer bloque de la *blockchain* de Bitcoin.

3.2. Que es *blockchain*

Blockchain [20] es un almacén de datos distribuidos. Este sistema crea copias en lo que se llama el libro mayor, esto significa que tiene una especie de registros digitales en la cual introducimos información y una vez esta es introducida, está ya no podrá ser borrada. Esta información se almacena en miles y millones de computadoras, también llamadas nodos. Un número razonable de nodos debe validar cada nuevo registro antes de que sea registrado. Una vez confirmado, el registro se almacena en el libro mayor y se propaga a través de la red de nodos participantes. En esta tecnología para poder añadir nueva información, se tiene que añadir bajo el acuerdo de todos los nodos.

Los contratos inteligentes, programas que se ejecutan automáticamente en el *blockchain* después de la finalización de la transacción, amplían aun más las capacidades de la tecnología. Los contratos inteligentes permiten el desarrollo de organizaciones autónomas y empresas que se pueden gestionar a través del consenso y el control de la multitud.

Blockchain es la descentralización en su expresión tecnológica. En vez de una autoridad central que adopte las decisiones y reglas de juego, los datos (criptomonedas, información, contratos, etc.) se transfieren, validan o almacenan de forma distribuida a lo largo de muchas redes de computadoras tendidas en diferentes partes del mundo sin ninguna jerarquía entre sí.

En el escenario que se trabaja bajo la cadena de bloques o *blockchain* aparecen:

1. Los bloques: conjunto de transacciones e información de la cadena. Están compuestos por un código numérico, el código del paquete a tratar y otro código numérico.
2. Mineros: son los ordenadores que permite la potencia computacional.
3. Nodos: son los ordenadores que almacenan y distribuyen la copia actualizada.

La cadena de bloque es una red P2P[2](red peer-to-peer) en la cual cada ordenador tiene la misma importancia.

[2] *Red peer-to-peer*: es una red de ordenadores en la cual estos actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Permite el intercambio de información entre los ordenadores que estén interconectados.

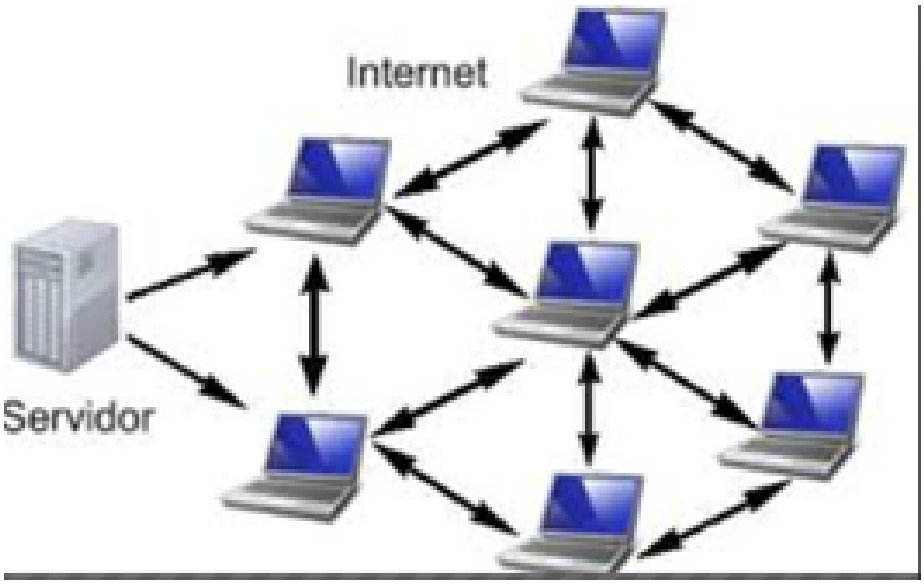

Figura 3.9: Red Peer-to-Peer

3.3. Proyectos que usan *blockchain.*

Hay muchos proyectos que usan blockchain, pero aquí destacaremos el uso de tres proyectos que se enfocan principalmente en esta tecnología *Bitcoin*, *Ethereum* y *Hyperledger*.

3.4. *Bitcoin*

Origen de *Bitcoin*

El origen del *Bitcoin* se da en 2007, el creador es: Satoshi Nakamoto y él originó este sistema de pagos electrónicos basado en pruebas criptográficas en vez de confianza, permitiendo que dos partes interesadas realicen transacciones directamente sin la necesidad de una tercera parte confiable. Este sistema dio origen a la criptomoneda *Bitcoin* y la tecnología con la que opera se denominó *blockchain* o cadena de bloques.

Bitcoin surge como una solución a la necesidad de los usuarios de un intercambio mundial facilitado por herramientas de pago transaccionalmente y que, a menudo, se traduce en la voluntad de no tener que emplear la moneda o los billetes.

Figura 3.10: Moneda Bitcoin

Para qué sirve Bitcoin

Bitcoin es una moneda, como el euro o el dólar estadounidense, que sirve para intercambiar bienes y servicios. La diferencia con otras monedas es que *Bitcoin* es una divisa electrónica que presenta novedosas características y destaca por su eficiencia, seguridad y facilidad de intercambio. Su mayor diferencia frente al resto de monedas se trata de una moneda que nadie la controla(descentralizada). La idea, plasmada en un *white paper* era crear un sistema de intercambio de efectivo electrónico P2P (*peer-to-peer*) que no pasara por instituciones financieras tradicionales.

Proceso de extracción

Proof of work o prueba de trabajo

La prueba de trabajo [47] es un proceso que nace con el objetivo de eliminar los ataques a redes informáticas a través de la realización de una prueba moderadamente difícil (ej. demostrar que no eres un robot) es una especie de *captcha*[3] para poder permitir la realización de la acción que se desee.

[3]Es un test de Turing público y automático para distinguir a los ordenadores de los humanos, sirve de medida de seguridad conocido como autenticación pregunta-respuesta.

Este algoritmo es usado para realizar las transacciones y poder obtener o producir nuevos bloques en la cadena. Con esto se consigue que los mineros compiten entre ellos para completar transacciones en la red y obtener recompensas.

La implementación de este algoritmo es un pequeño pequeño rompecabezas que depende de varios factores:

- Cantidad de usuarios.
- Potencia actual del equipo
- Carga que soporta la red.

Como dato, sabemos que el *hash* de cada bloque contiene el *hash* del bloque anterior, esto lo que nos garantiza es que tiene mayor seguridad y evita la posible violación del bloque.

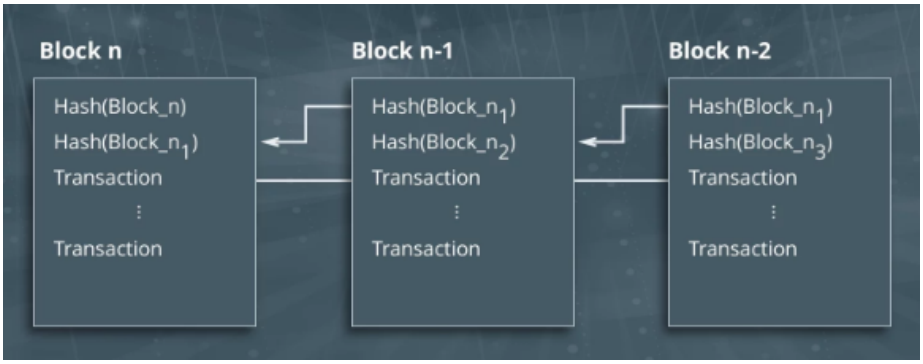

Figura 3.11: Prueba de trabajo

El tiempo medio de la formación de bloque es de diez minutos.

Proof of stake o prueba de participación

La prueba de participación [48] es un protocolo de consenso[4] que asegura una red de posesión de monedas. Con este sistema, las opciones de encontrar

[4]Protocolo de consenso: encontramos un problema ya que consiste en poner de acuerdo a múltiples procesos en una cosa conjunta. Cuantos menos fallos más fácil llegar a consenso, pero habitualmente la solución a estos problemas suele ser un algoritmo o algún tipo de protocolo antes establecido, esto siempre suele ser usado por los procesos que no tienen intenciones maliciosas.

un bloque de transacciones y recibir monedas son proporcional a la cantidad de monedas que tenemos almacenadas en nuestra cuenta.

Tenemos diferentes estrategias desarrolladas en forma de algoritmo:

- Prueba de participación pura: este algoritmo consiste en la creación de bloques mas fáciles para los que controlan gran cantidad de monedas.
- Prueba de deposito o *proof of deposit*: este algoritmo esta enfocado al momento en el que las monedas son usadas por el dueño del bloque y crear otros nuevos. Estas son congeladas hasta que se confirme un número determinado de bloques. Aquí no se recompensa a un minero por almacenar monedas no gastadas en el pasado, se recompensa a los que están dispuestos a tener monedas inmóviles durante un gran periodo de tiempo.

Dinero electrónico

Un *Bitcoin* no es más que una cadena de caracteres que está vinculada en un momento a una cartera. Un usuario crea una cartera, que contiene una dirección pública, la cual es anónima, y dicha cartera tendrá almacenadas las criptomonedas similar al funcionamiento de un banco electrónico. Esa dirección puede compartirse para recibir pagos. Toda transacción entre *wallets* queda registrada en el blockchain, la médula espinal de las criptomonedas. El *blockchain* registra todas las transacciones de esta moneda virtual que se han completado desde la creación de la cartera.

La divisa ahora mismo .ª fecha de 09/04/2019.ᵉˢ: 1 Bitcoin equivale a 7.233,81 €

3.5. *Ethereum*

Origen de *Ethereum*

En el 2014, surge Ethereum [6] de manos de Vitalik Buterin de origen ruso. Se trata de un investigador y programador de criptomonedas.

El desarrollo de *Ethereum* empezó siendo financiado por un *crowdsale*[5]

A mediados de 2015, se crearon 72 millones de monedas *Ether*, y actualmente esta cifra representa sobre le 70 % del total de Ether mundial.

[5] *Crowdfunding* [33]: práctica para financiar proyetos al recaudar por medio de Internet pequeñas cantidades de dinero de un gran número de personas.

Para qué sirve *Ethereum*

Ethereum [7] es una plataforma *open source*[6], descentralizada que permite la creación de acuerdos de contratos inteligentes entre pares, basada en el modelo *blockchain*. Cualquier desarrollador puede crear y publicar aplicaciones distribuidas que realicen contratos inteligentes. *Ethereum* también provee una ficha de criptomoneda que se llama *Ether*. Se puede intercambiar *Ether* entre cuentas diferentes y también es utilizado para compensar los nodos participantes por los cálculos realizados.

El *Ether* es la criptomoneda de *Ethereum*.

La divisa ahora mismo .ª fecha de 09/04/2019.es: 1 *Ether* equivale a 146,95€.

Ventajas del *Ether*

Las principales ventajas del *Ether*, con respecto a las monedas tradicionales, son similares a las otras criptomonedas y parecidas a al *Bitcoin*, y se pueden resumir en:

- No las controla ningún gobierno y no se puede contar su valor.
- Son más difíciles de falsificar que las monedas tradicionales €, $, £, etc
- No existen intermediarios cuando se realizan transacciones o pagos y esto hace que sea más barato el realizarlas.
- El sistema de código es abierto, por lo tanto, permite realizar mejoras, como nos pasa con el Bitcoin.

¿Qué es el Gas?

El Gas [32] es un proceso de pago que tiene como obligación establecer un determinado número de ejecuciones computacionales. Con esto lo que evitamos es fomentar los bucles infinitos y cualquier otro desperdicio en el código.

[6] *Plataforma source*: programas informáticos que permiten el acceso a su código de programación, lo que facilita modificaciones por parte de otros programadores ajenos a los creadores originales

El Gas persigue el objetivo final de controlar los precios de las transacciones. Al mismo tiempo, sirve para reducir los *spam* y poder asignar recursos en la red.

La meta es que cada usuario pague una comisión por los recursos que consume. Aquí se incluye el ancho de banda, la computación y el almacenamiento. Por ello, al realizar una operación que consuma unos recursos tendrá una comisión proporcional al uso que haga de esta.

3.6. *Hyperledger*

Origen de *Hyperledger*

El origen se sitúa en diciembre de 2015 y fue creado por la Fundación *Linux*, es una plataforma código abierto para *blockchain*, para apoyar a los *ledgers* distribuidos basados en la *blockchain*.

Es una plataforma privada de *Blockchain*. Es una colaboración global entre empresas y expertos para facilitar y mejorar la adopción de la cadena de bloques entre ellas.

Proyectos de *Hyperledger*

Actualmente en *Hyperledger* [19] se encuentran nueve proyectos, los cuales se pueden dividir en dos ramas principales:

1. Plataformas *blockchain*

    1) Hyperledger Burrow: impulsada por la *startup* Monax Industries y es una *blockchain* privada basada en código de Ethereum. Permite el despliegue de *smart contract* dados en Solidity.
    2) Hyperledger Fabric: es el proyecto mas conocido de Hyperledger. Orientado al mundo de empresas (IBM). Intenta facilitar la implementación en la plataforma multidisciplinar de cualquier modelo de uso. Permite el uso de Smart-contracts (*chaincodes*). Es flexible, incorpora como crear *chaincodes* en cualquier lenguaje.
    3) Hyperledger Indy: propone una solución a la identidad digital. Intenta que los datos tengan que ser gestionados por cada usuario. Procura no dar información a terceras personas.

4) Hyperledger Iroha: quiere ser una *blockchain* simple y modularizada que permita los *smart contract* desarrollados en Java, y pretende conseguir el menor uso de los datos personales.

5) Hyperledger Sawtooth: viene de Intel, una *blockchain* privada con uso empresarial que incorpora el despliegue de *smart contract*. También se desarrolla en Solidity.

2. Herramientas para la interacción con las plataformas

1) Hyperledger Composer: permite la creación de aplicaciones descentralizadas.

2) Hyperledger Explorer: intenta dar una interfaz para la monitorización de los nodos (*peers*) de Hyperledger Fabric. Tiene un servicio para la comunicación de *Blockchain* y la aplicación.

3) Hyperledger Cello: crea una solución que consiga un modelo de despliegue, "*blockchain* como Servicio". Reduce el esfuerzo de crear una infraestructura de nodos *blockchain*.

4) Hyperledger Quilt: el objetivo es la conexión entre distintas redes de *blockchain* mediante el protocolo Interledger (ILP). El protocolo intenta quitar los pagos entre distintos sistemas bancario. ILP permite realizar transferencias entre distintos "*ledgers*" (libro de cuentas) y permite la conexión de usuarios con las cuentas que tienen en distintos sistemas.

3.7. Plataforma de desarrollo: Solidity y D'app

Solidity

Solidity [8] es un lenguaje de alto nivel orientado a contratos. Su sintaxis es similar a la de *JavaScript* y está enfocado, específicamente, a la Máquina Virtual de Ethereum[7] (EVM).

En el ámbito de Solidity [9], los desarrolladores pueden escribir aplicaciones descentralizadas que se puedan implementar en automatizaciones de negocios mediante contratos inteligentes. Esto dejará un registro autorizado y con pruebas demostrables de los pasos realizados. En solidity

[7]EVM: software que interactúa con la red de Ethereum y permite ala creación de aplicaciones independientemente del lenguaje de programación usado.

tendremos una serie de modificadores, gracias a los cuales podremos pedir una cantidad de pago concreta o que la transacción tenga herencia, o una operación solamente la pueda realizar un usuario concreto, habrá posibilidad de crear arrays, eventos, funciones privadas, al ser un lenguaje actual y nuevo para nosotros, la información no era tan clara y abundante como en otros lenguajes.

Al decir que es un lenguaje de alto nivel nos referimos a un lenguaje de *Turing Complete*, este concepto fue creado por *Alan Turing*, y cuando se aplica a la tecnología *blockchain* se refiere a la capacidad de un lenguaje para poder resolver cualquier problema de computo y poder añadir nuevas reglas, como serían los bucles[8].

Construir sobre *Ethereum*

Las principales razones por la cual las *DApp* se construyen sobre Ethereum en vez de ser independientes son:

- Seguridad: construir esa *DApp* basada en *Ethereum* hará que sea la gran red de nodos de Ethereum la que procesen sus transacciones, aumentando exponencialmente su seguridad.
- Interoperabilidad: Están escritas en el mismo lenguaje (*Solidity*), por lo que todos los programadores que los programadores que lo conozcan lo saben interpretar, se pueden utilizar las transacciones y se pueden utilizar las unas con las otras (puede haber sinergias)

DApp

Herramienta de construcción, gestión de paquetes y asistente de despliegue para Solidity.

Las *DApp* [21] [22] son código *back-end*[9], el cual se ejecuta sobre la red P2P (*peer-to-peer*).

Son aplicaciones distribuidas que interactúan directamente con *blockchain*.

Con el uso de las DApp se permite descentralizar el código *Back-end* y los datos, esto quiere decir que son inmutables, no se pueden falsificar y solamente depende de la comunidad de usuarios que la utilizan.

[8] Bucle: una instrucción repetitiva para poder ejecutar los problemas.

[9] *Back-end*: es el que se encuentra en la parte del servidor(PHP, Python, Java) e interactúa con la base de datos verificando usuarios y montando la página

La aplicación descentralizada puede ser una aplicación móvil o una aplicación web que interactúa con un contrato inteligente para llevar a cabo su función.

Condiciones de una DApp

Las características que definen a una DApp son las siguientes:

1. Aplicación totalmente de código abierto, debe operar de forma autónoma, sin nadie que este controlando. Se pueden aplicar cambios, pero todos ellos tiene que ser decididos por consenso entre los usuarios.
2. Los datos de cada operación son almacenados en una *blockchain* pública y descentralizada.
3. La aplicación tendrá una ficha, la cual es necesaria para acceder y recompensar a los mineros que trabajen sobre dicha aplicación.
4. La propia aplicación generará fichas, dependiendo del algoritmo, para tener una prueba de su valor.

Tipos de *DApp*

1. Tipo I: estas son las que tendrían su propia cadena de bloques independiente. Podría ser el equivalente a un sistema operativo de un ordenador.
2. Tipo II: utilizan la *blockchain* de una aplicación descentralizada tipo I en vez de tener ellas una propia. Este tipo de DApp son protocolos que funcionan ya sea con sus propios tokens o con los *tokens* de la *blockchain* en la que operan. Ej: raiden network. Sería el equivalente a un programa con un propósito general, véase un Word o Excel.
3. Tipo III: utilizan el protocolo de una aplicación descentralizada de tipo II. Sería lo mismo que una solución de software especializada para ese procesador de texto, como un plugin o herramienta que añada servicios al Word, Excel o Dropbox.

3.8. *Smart contracts*

Los *smart contracts* tienen como objetivo eliminar intermediarios con el objetivo de simplificar los procesos para ahorrar costes al consumidor.

¿Qué es un *smart contract*?

Un *smart contract* [24] [25] o contrato inteligente es similar a un contrato y no es más que un acuerdo entre dos o más implicados con unas reglas que tendrán que aceptar todas las partes para, así, entender los objetivos a cumplir.

Anteriormente los contratos se cerraban de forma verbal o mediante documentos escritos, en cambio, en los contratos inteligentes son capaces de ejecutarse y poder cumplirse mediante ellos mismos, de una forma automática y autónoma, sin la necesidad de ningún intermediario o notario.

Los *smart contract* son unos *scripts* [10] que, a la vez, son los términos del contrato. Los *smart contract* tienen una validez total sin depender de nadie ya que es un visible por todos y no se puede cambiar al existir sobre la tecnología *blockchain*, que es la causante del carácter descentralizado, inmutable y transparente.

Ventajas de los *smart contracts*

Las principales ventajas de estos contratos son:

- Autonomía: no se necesita ningún intermediario ya que tú realizas el acuerdo y con esto esto se evita el problema de la manipulación de terceras personas
- Confianza: los documentos están encriptados y no hay manera de qué se pueda decir que se perdió.
- Respaldo: los documentos son guardados en varios ordenadores, por lo que con esta cadena de bloques estás bien cubierto ante la posibilidad de perdida.
- Seguridad: gracias a la criptografía, los documentos se mantienen a salvo y no esta la posibilidad de que se realicen *hackeos*.
- Velocidad: gracias a los contratos inteligentes podemos reducir notoriamente el tiempo empleado estos también utilizan un código de *software* que consigue automatizar las tareas y ahorrar tiempo.

[10] *Script*: es un documento que contiene instrucciones, escritas en códigos de programación.

- Ahorros: gracias a este tipo de contrato conseguimos ahorrar mucho dinero ya que que eliminamos la presencia de uno o varios intermediarios.
- Precisión: estos contratos son muy rápidos, económicos y evitan los errores de rellenar formularios.

Se podría decir que gracias estos contratos inteligentes se puede garantizar un conjunto de resultados muy específicos, evitando las posibles confusiones y no da lugar a riñas o juicios. Se podría decir que es un gran cambio para los negocios y la tecnología de bloque.

Usos de los *smart contracts*

El mundo de los contratos inteligentes tiene mucho futuro por delante, pero ya se puede empezar a usar. Estos son algunos de los usos que se pueden dar:

1) Votaciones: sería extremadamente difícil que pueda ser manipulado ya que tendrían que ser descodificados y se requiere una potencia computacional excesiva para poder acceder a los registros digitales.
2) Gestión: con esto eliminamos las comunicaciones defectuosas y el flujo de trabajo iría mejor por su precisión, transparencia y sistema automatizado. Evitaríamos tiempo de espera para realizar aprobaciones y, así, los problemas internos o externos se podrían resolver por sí mismos.
3) Atención médica: se podrían almacenar los registros en cadenas de bloques con una clave para cada usuario y, así, asegurar que la investigación se está realizando de manera legal y confidencial.

Plataformas de *smart contracts*

Smart contracts en *Ethereum*

Cuando hablamos de *Ethereum*, estamos hablando de uno de los proyectos más famosos en el mundo de los *smart contract*. Esta plataforma de computación. Está basada en *blockchain* y permite ejecutar los *smart contract* en la red *peer-to-peer (P2P)* entre nodos sin necesidad de un servidor central. Esto se ejecuta en una máquina virtual descentralizada llamada *Ethereum Virtual Machine (EVM)*.

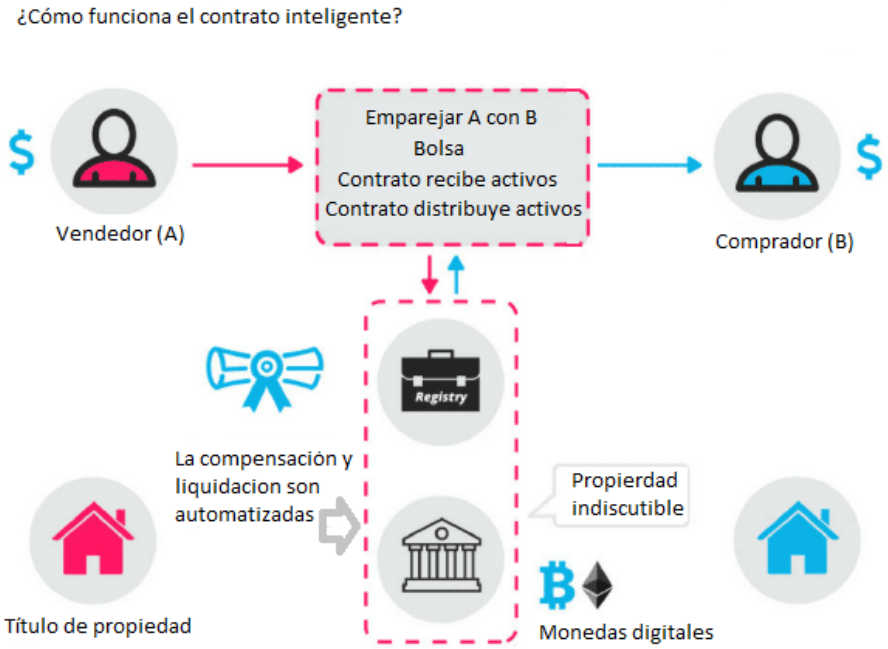

Figura 3.12: ¿Cómo funcionan los *smart contracts*?

3.9. Plataformas de desarrollo

A continuación veremos diferentes tipos de plataforma que son utilizados en la tecnología *blockchain*:

- *Eris*: Un software que permite a cualquiera crear su propia aplicación segura, de bajo costo y ejecutar en cualquier lugar utilizando *blockchain* y tecnología de contrato inteligente.

- *HydraChain*: su principal dominio de aplicación son las cadenas privadas o las configuraciones de la cadena de consorcios, especialmente en la industria financiera.

- *OpenChain*: es una tecnología de contabilidad distribuida de fuente abierta. Es adecuado para organizaciones que desean emitir y administrar activos digitales de una manera robusta, segura y escalable. Se realiza a tiempo real, con transacciones gratuitas y de forma inmutable y jerárquica.

- *EOS* [31]: es una de las plataformas descentralizadas más poderosas del mercado, es mas rápida, escalable y permite la creación de aplicaciones de forma mas eficiente que con *Ethereum*.

  Es *software* abierto y tiene un algoritmo diferente que es DPOS (*Delegated Proof of Stake*). Gracias ello se pueden producir bloques mediante sistemas de votos y aprobación. Esta plataforma fue desarrollada por *block.one* y permite millones de transacciones por segundo.

- *Litecoin* [26] es también otra criptomoneda similar a *Bitcoin*, también sustendada por la red P2P. Utiliza un algoritmo diferente ya que no requiere tantos recursos computacionales para su minería. El tiempo empleado para confirmar las transacciones es mucho mejor que en *Bitcoin* ya que solo tendremos que esperar cinco minutos para recibir las dos transacciones, mientras que en la red *Bitcoin* es necesario esperar diez minutos.

- *Waves* [30] posiblemente es uno de los proyectos mas completos en el mundo de las criptomonedas. Es una plataforma multifuncional basada en *blockchain* que ejecuta varias funciones útiles para los activos digitales. Es una plataforma descentralizada, transparente y esta diseñada para un uso masivo.

3.10. Diferencia entre *Ethereum* y *Bitcoin*

La principal diferencia [18] reside en los objetivos y las finalidades de cada uno de estos proyectos.

- *Bitcoin*: pretende ser un depósito de riquezas para llegar a ser, en un futuro cercano en una moneda que pueda ser adoptada a escala mundial y que pueda sustituir el dinero tal como lo conocemos.

- *Ethereum*: persigue convertirse en una plataforma en la que se puedan ejecutar contratos inteligentes y aplicaciones descentralizadas.

Otra diferencia es la cantidad máxima de moneda que va a ser producido: mientras que *Ethereum* no tiene techo máximo, *Bitcoin* tiene un máximo fijado en 21 millones. Ambas monedas se producen mediante un proceso de extracción.

Técnicas y herramientas

En este apartado describiremos las técnicas y herramientas que hemos usado para la realización de este proyecto.

4.1. Técnicas utilizadas para el funcionamiento de la página WEB

Solidity

Solidity [8] [9] [10] es un lenguaje de programación de alto nivel, está diseñado para crear y desarrollar contratos inteligentes que se ejecuten en la *Machine Virtual Ethereum*(EVM). La síntesis es similar a *Javascript*.

Al estar diseñado para el desarrollo de Contratos Inteligentes, se dice que es algo limitado en el uso, aunque este lenguaje también está considera en el ámbito de la programación como muy poderoso.

Figura 4.13: Logo *Solidity*

Solidity funciona con la creación de contratos "Smart Contracts" lo cual ayuda a que muchas partes de un negocio funcionen correctamente por si mismas y se pueda llevar un mejor control de registro de las acciones actividades mencionadas anteriormente.

HTML y *CSS*

HTML [39],**HyperText Markup Language** "Lenguaje de Marcas de Hipertexto" es un lenguaje de marcado que se utiliza para el desarrollo de páginas páginas web.

Al contrario de lo que mucha gente piensa, no te incluye el diseño gráfico de la página web, esta herramienta solo sirve para decir cómo va organizado el contenido de nuestra página web. Esto se consigue a través de las llamadas marcas de hipertexto "más común mente llamas etiquetas" o mediante su nombre en inglés como "tags".

Actualmente se puede decir que HTML es el lenguaje estándar que se ha impuesto en la visualización de páginas web, ya que es el que todos los navegadores del momento han adoptado.

En el proyecto empleamos *HTML* para la creacción de la parte *front* de la aplicación *web*.

Figura 4.14: Logo *HTML*

CSS [40] o Cascading Style Sheets (Hojas de estilo en cascada)se puede un considerar una buena elección porque permite implementar un diseño gráfico para definir y realizar la presentación de una página o documento estructurado escrito en un lenguaje de marcado [41][11].

CSS permite poder ser usado en la presentación de los documentos HTML. Con este lenguaje podemos elegir una amplia variedad de opciones de presentación véase: colores, tamaño de letra y posición de imágenes.

Figura 4.15: Logo *CSS*

El objetivo principal de CSS es intentar diferenciar o separar la estructura HTML de la presentación. La web sería lo que se encuentra en la parte inferior, el contenido, y el CSS se podría decir que es lo que hace que se vea el contenido, permitiendo al programador decidir el programador la forma que se vea.

[11]Lenguaje de marcado: consiste en codificar un documento el cual incorpora etiquetas que contienen información adicional que nos informa de la estructura del texto o de su presentación en cada momento

Para la elaboración de nuestros CSS nos hemos ayudado de la pagina w3school: https://www.w3schools.com/css/default.asp

PHP

PHP [14] es un lenguaje es de código abierto que está preparado, entre otras cosas, para el desarrollo web y puede ser incrustado en HTML. Esta fue una de las principales razones para utilizar este lenguaje.

Hypertext Pre-processor PHP [15], es un lenguaje de programación y posterior maquetado, ya que es posible implementar diversas instrucciones que consigan obtener resultados, y conseguir la comunicación con el servidor de datos para la web.

Figura 4.16: Logo *PHP*

Uno de los grandes motivos por los que se ha utilizado PHP, aparte de por su uso durante el proceso del grado, es por ser una multi-plataforma gratuita. Con esta razón daría igual el sistema operativo con el que estuviéramos trabajando.

4.2. Técnicas utilizadas para el la conexión entre la web y nuestro Smart Contract

Truffle

Truffle [17] es un *framework* de desarrollo de *Ethereum* que facilita la labor a la hora de desarrollar, testar y desplegar los *smart contracts*.

Un *Framework*: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Truffle nos ofrece la posibilidad de:

- Compilación, enlace y despliegue de Smart contracts desde el propio *framework*.
- Depuración y *testing* automatizado de contratos.

- *Framework* con *scripts* de despliegue y migraciones en redes públicas y privadas.
- Acceso a cientos de paquetes externos y gestión con *EthPM & NPM*.
- Aplicación de una consola interactiva para comunicación directa con los contratos.
- Interacción con contratos mediante *scripts* externos.

Una capacidad de *Truffle* es desplegar nuestros *Smart contracts* de manera muy sencilla a distintos entornos. Los cuales se podrán modificar desde el *truffle-config.js*. Hablaremos de su intalación en el apartado de los anexos.

Remix

Remix [49] es un IDE basado en navegador web que le permite escribir contratos inteligentes de Solidity para luego implementar y ejecutar el contrato inteligente.

La página es mediante Internet y es: `https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.24+commit.e67f0147.js` *Remix* te dejará seleccionar el tipo de versión para la compilación y desde dónde realizaremos la ejecución de nuestro *smart contract*, que será una red local como es *Ganache* o será una red exterior.

Figura 4.17: Logo *Remix*

El objetivo de usar este programa es conseguir lo siguiente:

- Crear los contratos inteligentes y depurar la ejecución.
- Poder saber el estado y las propiedades de los *smart contract* que se han creado anteriormente.
- Reducir errores de codificación y poder aplicar mejoras, con un análisis del código.

Remix es normalmente utilizado para pequeños contratos, ya que no necesita una instalación y lo podremos usar tanto *on-line* como *off-line*,

pudiendo descargar un archivo y, a la hora de ejecutar, poder usar el "index.html"

Para los programas más grandes tendremos la opción de utilizar node.js (del cual hablaremos a continuación) o Docker entre otros.

Node.js

Node.js [43] [44] es un entorno en tiempo de ejecución multiplataforma de código abierto.

Esta concebido como entorno de ejecución de JavaScript y esta orientado a eventos asíncronos, el objetivo es construir aplicaciones en las redes escalables.

Por cada conexión callback este sera ejecutado, pero si no tiene trabajo que realizar node.js estará en modo hibernación, con node.js no tenemos que preocuparnos por si el proceso se bloquea, node.js siempre realiza I/O directamente esta es la razón por la que nunca se bloquea. Esta es una de las grandes razones por es se desarrolla en sistemas escalables.

Metamask

MetaMask [45] es un complemento de navegador que permite a los usuarios realizar transacciones de Ethereum a través de sitios web regulares.

MetaMask [46] es una herramienta *plugin* que sirve como puente entre las *dapps* y el navegador Chrome. Al usar esto nunca se compromete la seguridad, siendo posible el uso de varias cuentas y teniendo la suerte de no tener que llegar a usar un nodo Ethereum de forma completa.

La aplicación fue desarrollada con la intención de acercar la tecnología blockchain al usuario a través de una serie de DApps[12], las cuales funcionan como una extensión del navegador. Con ellas tenemos las opciones de:

- Acceder a servicios
- Firmar transacciones blockchain
- Administrar tokens
- Administrar varios monederos

[12] *"decentralized application"* o aplicaciones descentralizadas

Todos estos puntos anteriores se intentan conseguir en una interfaz amigable y segura.

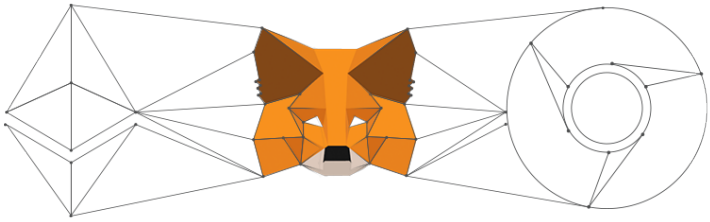

Figura 4.18: Metamask

La idea es integrar todos los servicios, incluyendo aquellos de Ethereum, en el navegador.

Una de las grandes preguntas que nos surgen al usar Metamask, es si es segura, de acuerdo con los investigado y leído hasta la fecha en diferentes paginas web no ha sufrido ningún daño, y tampoco se tiene constancia de que se haya realizado robos de criptomonedas por ningún hacker.

Metamask utiliza un sistema de seguirdad al cual se le llama HD, este consigue mantener todas las claves locales cifradas y gracias a ello se cumple el objetivo que las Dapps no tengan la posiblidad de acceder a ellas.

El sistema sistema de seguridad que tiene es conocido como "palabras semillas", las cuales tienen un orden concreto y sirven para recuperar la cuenta en el caso de que esta fuera robada.

Algo contra lo que intenta luchar Metamask es la suplantación de identidad, phishing, también conocido como billetera on-line. Metamask lleva tiempo intentando combatir este problema.

Ganache

Ganache [42] sirve para poder desarrollar aplicaciones en Ethereum y se realiza sobre una red local para realizar las pruebas.

Las pruebas serán como si estuvieramos trabajando sobre la red Ethereum de verdad, pero con la ventaja de que requiere tantos recursos y el desarrollo es más rápido.

Este programa es un simulador de la *blockchain* para Ehtereum en el que podremos desplegar contratos, ejecutar nuestros *tests*, ejecutar comandos o inspeccionar el estado de la red.

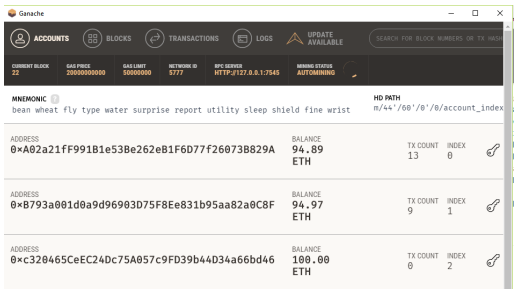

Figura 4.19: Ganache

4.3. Herramientas utilizadas para el funcionamiento de la página web, funcionando con base de datos

Visual Studio Code

Visual Studio Code [12] es un editor de texto con el cual hemos realizado gran parte de nuestro proyecto, tanto para el HTML como para la realización de los contratos una vez pasados por la plataforma Remix así como para realizar los procesos correspondientes para coger formato de nuestro contrato con JSON.

Nos decantamos por usar Visual Stuido [13] code ya que la edición de código no está limitada para los programas C# o Visual Basic, sino que acepta mas programas como php, html, solidity ...que son los principales percusores de nuestro trabajo final.

Otra parte positiva de este editor es la función "IntelliSense", la cual permite al usuario de este programa poder ahorrar tiempo al escribir instrucciones, ya que tiene una función de auto-completar con el propio editor, esto logra hacer que el trabajo sea mas productivo, acortar la escritura y no cometer errores por la sintaxis.

XAMPP

La siguiente herramienta de la que hablaremos será de la creación de la base de datos, en este caso hemos querido usar XAMPP [16](X[13] Apache, MySQL, PHP, Perl.)

[13]X: hace referencia a que sirve para cualquiera de los diferentes sistemas operativos

En el apartado 5, "Aspectos relevantes del desarrollo del proyecto" y en los anexos hablaremos mas específicamente sobre este programa. Explicaremos cómo realizar la instalación y el uso que le damos en nuestra aplicación. Por el momento daremos una breve introducción de por qué hemos seleccionado esta en vez de otro programa similar.

Una vez lo tengamos instalado, podremos realizar crear, editar, consultar y gestionar bases de datos en MySQL de una manera eficaz y limpia.

Al usar este programa, hemos retomado cómo poder administrar las bases de datos y su acceso a una base de datos desde PHP, así como realizar diferentes SELECT y sacar el resultado por página web.

4.4. Control de versiones y documentación

En el siguiente apartado veremos las herramientas que utilizamos para el control de las versiones y la herramienta empleada para el desarrollar del documento de este proyecto.

Látex

LaTeX [38] es un sistema de preparación de documentos. Con este programa podemos cartas, tesis, presentaciones y cualquier tipo de documento que quisieras imprimir en papel o mostrar en pantalla.

La principal opción por la que usamos *LaTex* es porque tiene una mejor calidad de tipografía en comparación con los editores convencionales.

Para el proyecto usamos *TexMaker* en la versión para *Windows*.

GitLab y GitHub

Gitlab [34] o *GitHub* [35] son servicios web para el control de versiones y desarrollo de software colaborativo basado en *Git*.

Utilizamos este programa para mostrar el proyecto finalizado. El repositorio se tendrá acceso en el siguiente repositorio [36],pero este repositorio es privado por lo cual hemos creado uno público en *gitHub* y se podrá observar la información en la pagina [37].

Aspectos relevantes del desarrollo del proyecto

En este apartado se van a mostrar los aspectos más importantes del proyecto y los procesos que hemos seguido para el correcto funcionamiento de nuestra pagina con la debida conexión con la red Ethereum. Hablaremos de las herramientas utilizadas, pero la instalación de la mismas y la finalización de la página lo comentaremos en el apartado de los anexos.

El trabajo comenzó con la visita de la compañía HP a la universidad a exponer y realizar una charla sobre los TFG que se podían realizar con ellos durante el curso, para así empezar a realizar trabajos con una empresa real para un trabajo determinado.

5.1. Comienzo del proyecto

El trabajo comienza a partir de un proyecto que propone la empresa HP(Hewlett Packard) sobre aplicación de tecnología Blockchain a una cadena de distribución de productos.

Una vez realizamos las primeras sesiones de para conocer al tutor de la empresa de HP: Pablo Tejedor García.

En la primera sesión decidimos los tiempos para realizar las diferentes tareas y ver la complejidad de cada una de ellas, todas las reuniones han sido realizadas vía llamada remota o presencialmente en la sede de León.

5.2. Primeras sesiones

En la primera sesión me recomendó la lectura del libro *[Scrum y xp desde las trincheras]* el cual hemos comentando anteriormente, y la realización de programas básicos de prueba para la realización de la página en Visual Studio ya que le dije que habíamos usado esta herramienta durante la carrera y según algunos artículos era posible la conexión con la tecnología Blockchain, mediante la red web3 y la aplicación de la cual hablaremos mas adelante Metamask.

5.3. Primer contacto con *Ethereum*

Una vez acabado y resumido el libro y realizados los primeros programas con Visual Studio, dimos el paso a la realización de nuestros primeros *smart contract*

En primer lugar, se decidió realizar un tutorial para tener unas tomas básicas en el ámbito de los *smart contract*, ya que era la primera vez que utilizaríamos este tipo de lenguaje.

Este tutorial se realiza entero mediante Internet, se trata de una serie de capítulos los cuales te enseñan desde cero la creación de un contrato Solidity.

Hablaremos de ello en el apartado de trabajos relacionados en el punto siguiente.

Una vez concluido este proceso de iniciación en el mundo de Solidity, el siguiente paso era realizar pequeños contratos con cambio de dinero entre cuentas Ethereum, pero para empezar todas estas operaciones las realizamos mediante la opción localHost, esto lo conseguimos mediante dos formas.

5.4. Realización de contratos en modo *LocalHost*

Para la realización de esto se empezó a usar una página de la cual no nos separemos a lo largo de nuestro proyecto, Remix.

Gracias a esté sitio web los smart contract que creemos se podrán probar directamente mediante la compilación.

Podremos seleccionar la versión con la que estamos trabajando y una vez compilado y sin errores, iremos a la pantalla de *DEPLOY & RUN TRANSACTIONS* una vez en esta página tendremos varias opciones

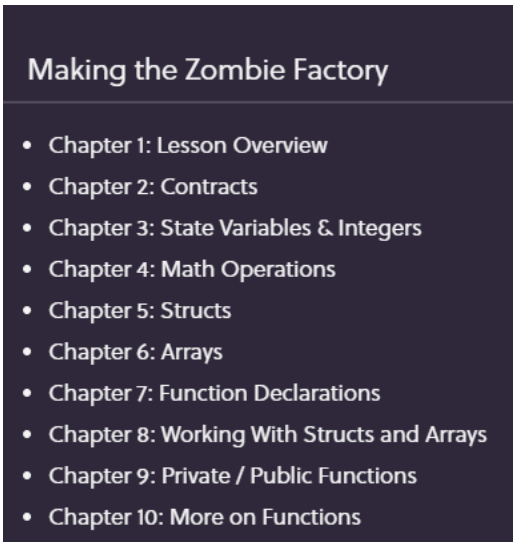

Figura 5.20: Funciones: CryptoZombies

En la opción *Enviroment*, podremos elegir como se realizaran nuestros contratos:

- JavaScript VM: esto nos crea 5 cuentas con una cantidad de 100 Ether en cada una, esto será una opción local para pequeñas pruebas, cuando cerreremos o ejecutemos otro programa la cuenta volverá a reiniciarse.
- Injected Web3: esta opción nos conectará directamente con Metamask, una vez que demos a *deploy*, desplegar.
- Web3 Provider: en esta opción nos conectará con el servidor local de Ganache y nos dará acceso a las 10 cuentas que hay disponibles.

Con todas estas opciones nos creará un primer contrato, sin ningún valor mas que el gas de transacción.

Gracias al desplegable Account podremos seleccionar la cuenta en la que estemos, siempre y cuando no estemos en Injected Web3, ya que con esta opción siempre manejaremos la red desde Metamask.

Otro apartado es Gas limit se podrá indicar cuanto queremos y en caso de no poner un valor que sea inferior, nos mostrara error en este apartado.

El ultimo parámetro que se puede modificar sera el valor que queremos indicar (Wei, Gwei o Ether), en caso de que nuestro contrato tenga una

cantidad de divisa a enviar y no pongamos el valor, nos mostrará error de cantidad invalida.

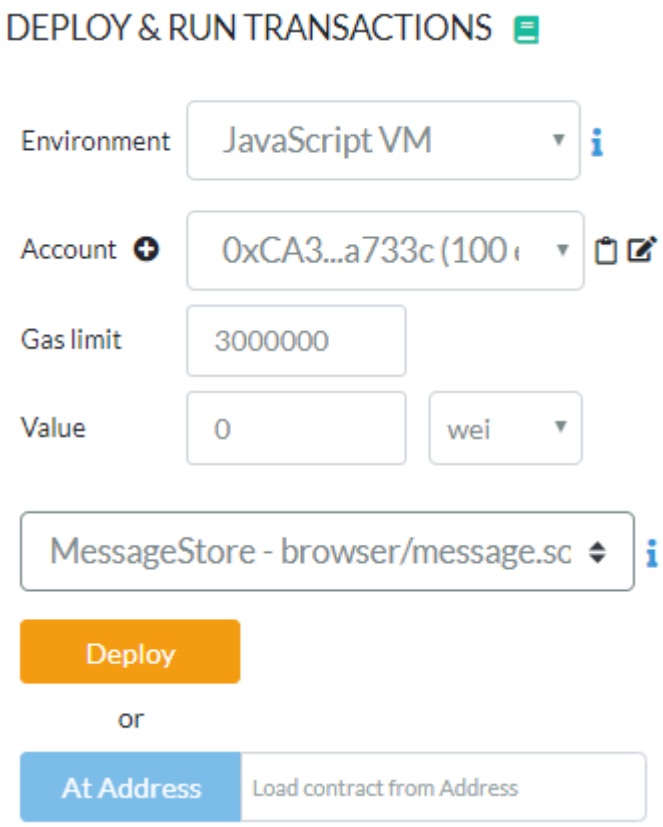

Figura 5.21: Ejecutar contrato desde Remix

5.5. Creación de contratos desde Visual Studio

Una vez fue entendida la herramienta con Remix, el siguiente paso fue la creación de una DApps, con las herramientas Visual Stuido, Tuffle, node.js y Ganache.

Primeramente tendríamos que instalar Node.js (https://nodejs.org/en/), una vez que fuera instalada instalaremos Truffle, y a continuación crearemos o ejecutaremos una carpeta la cual contendrá los siguientes archivos:

- contracts/: contiene los archivos Solidity que tendrán los contratos inteligentes.
- migratios/: sistema utilizado por Truffle para manejar las implementaciones de los smart contract.
- test/: aquí realizaremos las pruebas Solidity y JavaScript para los smart contract
- truffle-config.js: archivo de configuración de Truffle.

Una vez tengamos la carpeta creada, escribiremos los smart contract, indicando la versión que deseemos usar de Solidity, cuando este finalizado nos encargaremos de compilar (truffle compile), y a continuación los migraremos[14] a nuestra cadena de bloques.

Anteriormente hablábamos de la herramienta Metamask, ahora para la conexión de desde Visual Studio a nuestro contrato, será necesario la instalación, la podremos descargar desde https://metamask.io/, es una extensión disponible para Chrome, Firefox, Opera, Brave, iOS y Android. También necesitaremos la aplicación Ganache, https://www.trufflesuite.com/ganache, este programa nos surtirá en modo local de diez cuentas, para poder trabajar con ellas, pero una vez que cerremos todo lo realizado de las transacciones se perderá.

El proceso de instalación de las herramientas anteriormente mencionadas lo comentaremos en el apartado de anexos.

Una vez tengamos Metamask tendremos que se crear o seleccionar la cuenta localhost:7545 será la encargada de conectar con Ganache, importando previamente las cuentas (con el código privado de la clave).

Una vez tengamos todos creado, desde la terminal de Visual Studio (ctrl+ñ) tendremos que ejecutar npm run dev esto hará que el servidor se inicie y abra una pestaña en el navegador con el contenido de la dapp.

5.6. Programas creados antes del programa final

Antes de la creación final hemos realizado dos programas, ambos con conexión perfecta en ganache desde metamask, pero al intentar modificar

[14]Migración: *script* de implementación creado para alterar el estado de los contratos de su aplicación.

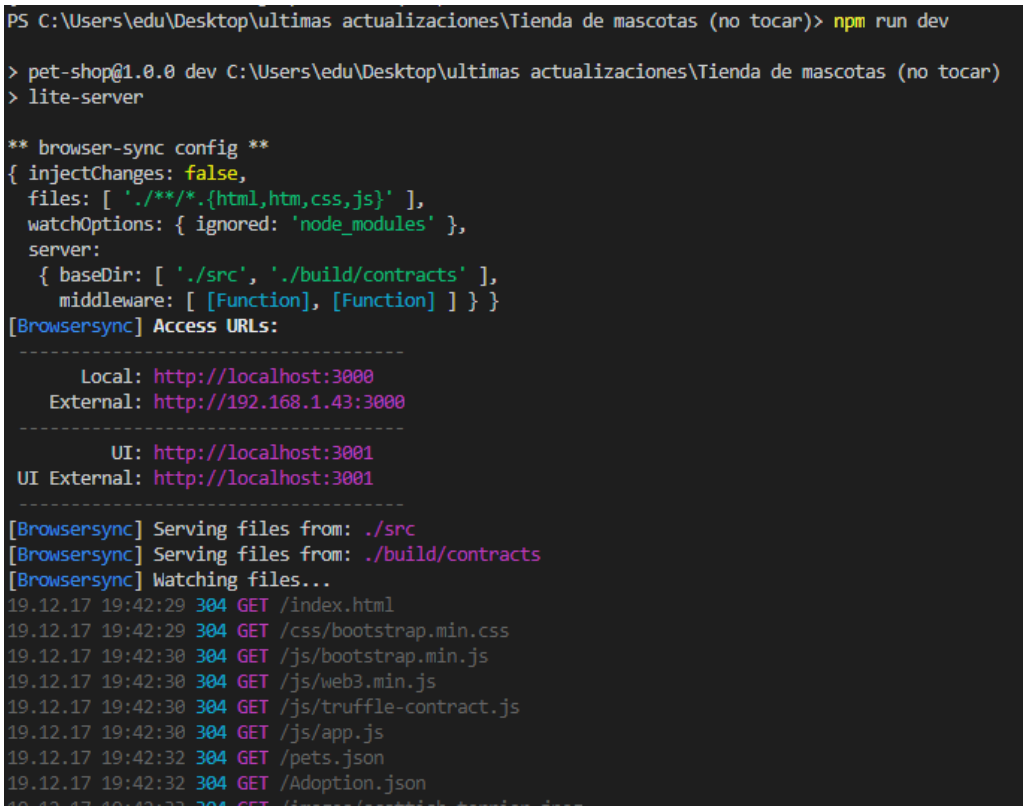

Figura 5.22: Muestra de la ejecucción desde Visual Studio

las redes y tener que usar una red principal o privada, surgieron los mayores problemas, bien por problemas de divisa en Ether en las cuentas (debido a su dificultad de conseguir esta moneda en las redes, recordamos que en las redes locales que manejábamos anteriormente las divisas nunca se acaban ya que si reiniciamos la cuenta el dinero volvería al estado inicial de 100 Ether).

Los archivos que hemos creado son un servicio de compra de autobuses y una tienda de mascotas

5.7. Empezando con redes privadas

Al finalizar los contratos mediante la red con Ganache y comprobar que se realizaban correctamente el siguiente paso que nos propusimos fue la conexión de los smart contract con una red privada como son Ropsten, Rinkeby, Kovan.

Lo primero que hay que hacer es ingresar Ether en la cuenta, investigando por Internet descubrimos que había una página la cual permitía recaudar dinero pero solamente para la red Rinkeby.

El proceso para conseguir Ether consta de las siguientes partes:

1 Iremos a la página https://plus.google.com/, también se podrá realizar mediante la red de Twitter o Facebook y ahí tendremos que publicar el número de cuenta donde deseemos recibir el Ether.

2 Una vez publicado copiaremos la url de nuestro post.

3 Por ultimo en la pagina https://www.rinkeby.io/#faucet pegaremos la url y seleccionaremos el Ether que deseemos recibir, dependiendo la cantidad que escojamos no podremos realizar una nueva petición hasta que pasen los días establecidos.

4 Cuando se complete el proceso, podremos comprobar la cuenta y comprobar que el Ether aumento la cantidad seleccionada previamente.

Ahora que ya tenemos dinero en nuestra red, podremos empezar a realizar smart contract desde Visual Stuido y conectar con las redes privadas mediante la aplicación que hace de intermediaria llamada Metamask.

5.8. Problemas de conexión Web3

En este punto en el cual ya disponíamos de dinero en nuestra red, intentamos la realización de poder ejecutar desde Visual Stuido a la red Ethereum, pero ya no en modo local como usábamos Ganache, si no realizarlo con Web3 y que realizara la conexión o con una red pública o privada.

Esto nos llevo mas de un quebradero de cabeza, ya que la conexión nos daba errores, por problemas de versiones de nuestro solidity y por asíncronos.

Nos planteamos el realizar una base de datos, la cual recogiera todos los datos y estos poder mostrarlos posteriormente, esto sin poder enseñarlos en la página de Etherscan.

Pero investigando y probando resulto que con la red privada Ropsten si que era posible realizar los contratos.

5.9. Creación de la página

Después de haber comprobado que nos permitía realizar los contratos de forma correcta y en la red privada, llego el momento de realizar la página web.

La cual hemos realizado con el programa XAMPP, es sistema de gestión de base de datos en MySQL y con Visual Stuido.

La pagina web hemos creado estas pantallas:

a Inicio de sesión

b Nueva cuenta

c He olvidado la contraseña

d Mostrar opciones

e Añadir producto

f Consultar último producto

g Consultar productos existente

h Administrador de la cuenta

i Modificar datos del propio usuario

Hablaremos más detenidamente del *font-end* y *back-end*, así como de la instalación de XAMPP en el apartado de los anexos.

Trabajos relacionados

En este apartado tendremos la oportunidad de hablar acerca de trabajos similares al que hemos realizado en este proyecto.

Ya que no hay ninguno que se relacione con trabajos de la universidad de Burgos, hablaremos sobre los encontrados en Internet y los que se realizan en la empresa HP.

Al comenzar el trabajo final, el tutor académico me pregunto sobre mi conocimiento de Blockchain y la programación con Solidity, al ser principiante con esta tecnología me insto a seguir estos cursos para empezar a familiarizarme con este tipo de programación:

CryptoZombies [50]: es una página, en la cual, tienes que seguir una serie de capítulos en la cual te enseñan la programación para la creación de contratos inteligentes en Solidity.

Enseñan los conceptos básicos y los tipos de variables que hay, a demás de como implementar funciones y poder ejecutarlas.

Pet-shop [51]: tenemos un tutorial, el cual, nos explicará como poder crear una Dapp, y sera un sistema para poder adoptar mascotas. Nos enseña la utilización de:

- Configurar el desarrollo en el que trabajaremos.
- Configurar un proyecto con Truffle.
- Escribir nuestro propio *smart contract*.
- Compilación del *smart contract*.

- Migrar y ejecutar el *smart contract*.
- Crear la interfaz de usuario.
- Interactuar con metamask para dar los pasos para la compra de la mascota.

Crypto Kitties : es una de una de las grandes aplicaciones con mas fama en el mundo de las *DApps*.

Con esta aplicación tendremos una familia de gatos "digitales" [52], y el gato que se forme podrá valer una cierta cantidad dependiendo sus cualidades.

Cabe destacar que cada gato en esta página es un *token* indivisible de Ethereum, así que cualquier compra, venta o transacción que se pueda realizar conllevara un gasto de minería.

Para poder interacutar con esta aplicación tendremos que tener instalado el plugin de Metamask y una cuenta con Ether suficiente para poder realizar las operaciones deseadas.

Brave [53]: es un navegador web de código abierto el cual esta basado en *Chromium*[15].

Este navegador tiene la capacidad de bloquear los anuncios(incluso lleva una cuenta de cuantos anuncios bloqueo) y una gran cantidad de rastreadores en linea, y asegura proteger la privacidad de los usuarios, reduciendo los datos con los anunciantes.

Según anuncian en la sus redes *Brave* es mas rápido que sus competidores, Chrome y Safari, en ordenadores es dos veces mas rápido y en dispositivos móviles puede llegar a ser hasta ocho veces mas rápido.

Tiene también versión para móvil en las versiones de Android y iOS.

Aragon : es un proyecto que pretende eliminar los intermediarios en el proceso de creación y mantenimiento de estructuras organizativas. Esto es posible a través del uso de la cadena de bloques.

[15] *Chromium*: [54] es un navegador web gratuito y de código abierto desarrollado por Google.

6.1. Empresas y organizaciones que usan *blockchain*

Hay múltiples empresas que han empezado a utilizar la tecnología *blockchain* en el día a día para mejorar la gestión u ofrecer al publico un servicio de más calidad.

A continuación, expondremos algunos ejemplos de estas empresas [23], las cuales las podremos dividir en diferentes sectores como:

Comercio

- *Walmart*: es una multinacional que quiere aprovechar el registro digital para mejorar las gestiones y los seguimientos de datos para usar en el día a día.

Envío de paquetería

- *FedEx*: una empresa de envíos que permite mejorar el proceso de resolución de disputas con los clientes. Quiere usarlo para identificar el tipo de datos, almacenarlo en una *blockchain* y también para almacenar registros en futuros envíos o problemas.

Protección de datos

- *Facebook*: explora el uso de *blockchain* para mejorar la seguridad de los datos y poder ofrecer mayor seguridad a los usuarios.
- *Google*: realiza la misma operación que Facebook para poder proporcionar seguridad a todos sus usuarios.

Transporte

- *British Airways*: lo realiza para administrar la información sobre los vuelos mas importantes en el aeropuerto de Londres y Miami. Con ello verifican la identidad de las personas que viajan en el vuelo, mediante la base de datos que se encuentra en *blockchain*.
- *Toyota*: para mejorar la conducción autónoma.

Bancos

- *MasterCard*: la compañía de **EE. UU.** lo utiliza para realizar los pagos de forma instantánea.
- *BBVA*: también quiere conseguir la rápida circulación de la moneda mediante la transferencia.
- *Banco Santander*: tiene una aplicación que ejecuta pagos internacionales, mediante los dispositivos móviles (*smartphones*). La transferencia puede ser efectiva en torno a los 40 segundos.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Tras haber finalizado con la realización del proyecto, podemos decir que hemos cumplido la mayoría de los objetivos y los requisitos que se propusieron al iniciar el mismo.

El objetivo principal, era la creación de contratos inteligentes en la red Ethereum o las redes privadas que ofrece esta misma, mediante una página web, en un principio hemos trabajado todo mediante la red Localhost, con la posibilidad de en un futuro crear un dominio y que fuera de ámbito general.

Gracias a la investigación y realización de este proyecto, he aprendido a usar nuevos lenguajes de programación (Solidity, React, Ethereum) y a coger experiencia en otros que ya había visto en la carrera pero con apenas experiencia en ellos (HTML, CSS, PHP).

En otro caso, gracias a la colaboración de HP en este proyecto de investigación, he aprendido el funcionamiento de una empresa. También nos hemos encontrado con limitaciones en el desarrollo del proyecto, ya que al no estar tan desarrollada la tecnología sobre blockchain, la información no era tan abundante en las redes como en otros lenguajes de programación.

Como apunte final, cabe destacar que los problemas o errores encontrados durante la realización del proyecto, me han ayudado a investigar e intentar buscar soluciones de forma más o menos efectiva dependiendo del error. Este proyecto ha sido un proceso de auto aprendizaje contando con la ayuda de Pablo Tejedor (tutor de HP) y de Ángel Arroyo (Tutor académico).

7.2. Líneas de trabajo futuras

Posibilidades de expansión

- Ampliación del catalogo CNAE para dar mayor especificación al producto.
- Ampliación de la aplicación para no usar solo la red ethereum sino también otras blockchain públicas o privadas.
- Creación de un sistema de usuarios ampliado (diferentes tipos de empresa según su sector de actividad).
- Mejoras visuales de la aplicación.
    a) Tracking del producto (incorporación de servicio de mapas tipo Google Maps a la app). usar marcado GPS.
    b) Añadir fotografía del producto añadido.
    c) Anexar documentación oficial sobre el producto y introducirlo en la blockchain.
- Publicar la página en servicio en linea, así la pagina estará siempre disponible para su uso.

Bibliografía

[1] Información Blockchain, https://www.binance.vision/es/blockchain/history-of-blockchain, 2019.

[2] Henrik Kniberg, http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf, 2019.

[3] Bitcoin, breve introducción https://bitcoin.org/es/, 2019.

[4] Bitcoin, ¿qué es?, https://www.queesbitcoin.info/, 2019.

[5] HashCash, https://www.queesbitcoin.info/, 2019

[6] Ethereum,https://es.wikipedia.org/wiki/Ethereum#cite_note-wall-2, 2019.

[7] ¿Que es y para que sirve *Ethereum*?, https://en.wikipedia.org/wiki/Ethereum, 2019.

[8] Solidity, https://solidity-es.readthedocs.io/es/latest/, 2019.

[9] Solidity, ¿Para que sirve?, https://www.miethereum.com/smart-contracts/solidity/, 2019.

[10] Solidity, más datos, https://www.miethereum.com/smart-contracts/solidity/#toc6, 2019.

[11] Truffle, https://www.trufflesuite.com/, 2019.

[12] Visual Stuido Code, https://es.wikipedia.org/wiki/Visual_Studio_Code, 2019.

[13] ¿PorquéV isual Stuido Code?, https://blogs.itpro.es/eduardocloud/2016/08/22/visual-studio-code-que-es-y-que-no-es/, 2019.

[14] PHP: introducción, https://es.wikipedia.org/wiki/PHP, 2019.

[15] PHP: breve explicación, https://www.deustoformacion.com/blog/programacion-diseno-web/7-caracteristicas-lenguaje-php-que-lo-convierten-uno-mas-potentes, 2019.

[16] Base de Datos XAMPP, http://panamahitek.com/bases-de-datos-mysql-con-xampp/, 2019.

[17] Truffle, https://www.trufflesuite.com/, 2019.

[18] Diferencia Ethereum vs Bitcoin, https://www.luno.com/learn/es/article/whats-the-difference-between-bitcoin-and-ethereum, 2019.

[19] Hyperledger y proyectos de Hyperledger, https://www.eleconomista.es/economia/noticias/8899454/01/18/Hyperledger-la-Blockchain-privada-que-todos-tenemos-que-conocer.html, 2019.

[20] Blockcahin, http://cryptoparaprincipiantes.org/plataformas-de-desarrollo-blockchain.html#Que_es_Blockchain, 2019

[21] Dapp y tipos de Dapp, https://www.miethereum.com/smart-contracts/dapps/#toc10, 2019.

[22] ¿Que son las Dapp?, https://iuta.education/noticias/que-es-una-dapp/,2019.

[23] Empresas que apuestan por Blockchain, https://criptotendencia.com/2018/02/18/que-empresas-estan-adoptando-la-tecnologia-blockchain/, 2019.

[24] ¿Qué son los *smart contract*?, https://blockgeeks.com/guides/es/contratos-inteligentes/, 2019.

[25] , ¿Qué son los *smart contract*?, https://academy.bit2me.com/que-son-los-smart-contracts/,2019.

[26] , ¿Qué es *litecoin*?, https://efxto.com/divisas/criptomonedas/litecoin, 2019.

[27] ¿Qué es *tron*?, https://www.brokeronline.es/tron/,2019.

[28] ¿Qué es *neo*?, https://www.criptolog.com/que-es-neo/, 2019.

[29] ¿Qué es *neo*?"https://www.fayerwayer.com/2018/04/neo-plataformas-blockchain/, 2019.

[30] ¿Qué es *waves*?, https://coinlist.me/es/altcoins/waves/, 2019.

[31] ¿Qué es *EOS*?, https://www.bitcobie.com/eos/, 2019.

[32] ¿Qué es el *Gas*?, https://www.criptonoticias.com/criptopedia/que-es-ethereum-eth/, 2019.

[33] ¿Qué es el *crowdfunding*?", https://en.wikipedia.org/wiki/Crowdfunding, 2019.

[34] ¿Qué es *GitLab*?, https://es.wikipedia.org/wiki/GitLab, 2019.

[35] ¿Qué es *GitHub*?, https://es.wikipedia.org/wiki/GitHub, 2019.

[36] Repositorio GitLab, https://gitlab.com/HP-SCDS/Observatorio/2018-2019/ubu-bc4distribution/tree/paginaWeb, 2019.

[37] Repositorio GitHub, https://github.com/ers0026/ProyectoFinal, 2019.

[38] ¿Qué es *Latex*?, https://www.latex-project.org/about/, 2019.

[39] ¿Qué es y para que se usa*HTML*?, https://es.wikipedia.org/wiki/HTML, 2019.

[40] CSS, ¿Qué es y para que se usa?https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada, 2019.

[41] Lenguaje de marcado, ¿qué es?https://es.wikipedia.org/wiki/Lenguaje_de_marcadoL, 2019.

[42] ¿Qué es y para que se usa *ganache*?, https://truffleframework.com/ganache,2019.

[43] Node, ¿Qué es?, https://es.wikipedia.org/wiki/Node.js,2019.

[44] Acerca de Node.js, https://truffleframework.com/ganache,2019.

[45] Metamask, https://metamask.io/,2019.

[46] Metamask, como herrmienta https://www.criptonoticias.com/aplicaciones/metamask-puente-ethereum-navegador/,2019.

[47] *Proof of work* o prueba de trabajo, https://academy.bit2me.com/que-es-proof-of-work-pow/, 2019.

[48] *Proof of stake* o prueba de participación, https://es.wikipedia.org/wiki/Prueba_de_participaci%C3%B3n, 2019.

[49] *Remix*, ¿qué es?, https://theethereum.wiki/w/index.php/Remix, 2019.

[50] *CryptoZombies*, https://cryptozombies.io/es/course/, 2019.

[51] *Pet-shop*, https://truffleframework.com/tutorials/pet-shop, 2019.

[52] *Crypto Kitties*, https://www.criptonoticias.com/tutoriales/tutorial-cria-intercambia-cryptokitties-gatitos-viven-blockchain-ethereum/ 2019.

[53] *Brave*, https://brave.com/, 2019.

[54] *Chromium*, https://es.wikipedia.org/wiki/Chromium_(navegador), 2019.