UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación para compartir información basada en la tecnología blockchain

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Moreno Gil, Alex

Tutor/a: Vidal Oriola, Germán Francisco

CURSO ACADÉMICO: 2021/2022

Resum

L'objectiu d'aquest projecte és oferir una porta d'entrada a la tecnologia blockchain d'una manera intuïtiva i visual per a facilitar l'enteniment de les seues tècniques i patrons de disseny, cada dia més presents en el món digital. Per a això s'exposarà la seua utilitat, situació actual, casos d'ús i problemes. A més, es desenvoluparà pas a pas una aplicació blockchain en Python que mostrarà gràficament el seu funcionament intern per mitjà d'una aplicació web.

Paraules clau: blockchain, didàctica, Python, aplicació web

Resumen

El objetivo de este proyecto es ofrecer una puerta de entrada a la tecnología blockchain de una manera intuitiva y visual para facilitar el entendimiento de sus técnicas y patrones de diseño, cada día más presentes en el mundo digital. Para ello se expondrá su utilidad, situación actual, casos de uso y problemas. Además, se desarrollará paso a paso una aplicación blockchain en Python que mostrará gráficamente su funcionamiento interno por medio de una aplicación web.

Palabras clave: blockchain, didáctica, Python, aplicación web

Abstract

The aim of this project is to offer a gateway to the blockchain technology in a intuitive and visual manner in order to facilitate the comprehension of its techniques and designs patterns, increasingly present in the digital world. For that purpose, its utility, current situation, use cases and problems will be explained. Additionally, a blockchain application which displays grafically its inner behaviour through a web app will be developed in Python.

Key words: blockchain, didactic, Python, web app

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

1.1 Motivación

Desde la aparición del Bitcoin en 2008, el campo de la tecnología blockchain ha vivido una evolución frenética [1]. De la misma manera que existen quienes la presentan como una de las nuevas grandes tecnologías de nuestra época[2, 3, 4] o como el catalizador de la cuarta revolución industrial [5, 6], sus detractores limitan las aplicaciones de esta a campos muy concretos. Aún no está claro cuál de estas afirmaciones acabará por cumplirse, pero lo que está claro es: (1), que ha llegado para quedarse y (2), que nuevos avances y paradigmas no paran de aparecer.

Comenzando hace mucho pero de manera relevante a lo largo de la pasada década y más notablemente al principio de esta, la sociedad ha tenido que lidiar con la digitalización de la población. A pesar de agilizar muchas gestiones, ha quedado patente cómo ciertos sectores de la población se han visto superados por tecnologías a las que no estan acostumbrados[7] y que en la mayoría de casos nadie se ha molestado en explicarles.

Es por estos dos hechos, que se ha considerado que la existencia de material didáctico libre, sobre todo de nivel introductorio, debería considerarse de máxima importancia. Esta ha sido la motivación para desarrollar una aplicación interactiva y visual que asista a aquellos interesados en comprender la estructura y funcionamiento básicos que hay detrás de la tecnología Blockchain.

1.2 Objetivos

Dados los factores presentados en la motivación, se consideran de máxima prioridad los siguientes objetivos:

- **Tener potencial de aprendizaje:** El punto fundamental de cualquier elemento de aprendizaje debe ser la relación de lo presentado con el mundo real y como el sistema puede facilitar su aprendizaje.
- **Funcionamiento que respete las especificaciones presentadas:** Es importante que una herramienta didáctica, y más una que da posibilidad a usarse sin la supervisión de un tutor, no presente errores que puedan llevar al usuario a confusión o a aprender ideas erróneas.
- **Interfaz clara:** En cualquier entorno de trabajo habitual la interfaz debe ser lo más transparente posible y más aún en una herramienta de aprendizaje debe reducirse la necesidad de familiarizarse con la interfaz para poder usarla eficazmente.

- **Arquitectura simple:** La aplicación pretende ser entendible tanto para aquellos que quieran entender la tecnología simplemente ejecutándola como para aquellos con intenciones de comprender el código subyacente.

- **Código abierto:** Para cumplir con las motivaciones que han resultado en la creación de este trabajo es necesario que se publique de forma abierta, con una licencia poco restrictiva que facilite el desarrollo futuro.

1.3 Impacto esperado

Se espera que una vez desarrollada y publicada la aplicación resultante pueda usarse como un punto de entrada al aprendizaje de la tecnología Blockchain, tanto de manera autodidacta como en un entorno de educación convencional.

Además, el hecho de que la tecnología Blockchain esté intimamente ligada a las criptomonedas incentiva la proliferación de materiales introductorios intencionalmente parciales o engañosos que pueden llevar a una primera mala experiencia en el campo. Una posible popularización de una herramienta simple, interactiva y libre de intereses podría llegar a reducir estas exposiciones. Esta relación con nuevos factores económicos está comenzando el debate de si la Blockchain debería ser enseñada en escuelas de negocios [8]. El trabajo desarrollado pretende proporcionar herramientas para todos los ámbitos de aprendizaje de esta tecnología.

1.4 Tecnología Blockchain

Antes proceder con el cuerpo de la memoria, se considera necesario contextualizar el tema a tratar hablando de las bases de la tecnología Blockchain.

1.4.1. ¿Qué ofrece la tecnología Blockchain?

La blockchain es una tecnología de registro distribuido que permite una mayor transparencia, seguridad y resistencia a la censura de los datos. Almacena información de forma cronológica y transparente, lo que permite una mayor auditabilidad de la información. La Blockchain permite que los usuarios controlen sus datos y los compartan de forma segura. La inmutabilidad, descentralización y transparencia son las características fundamentales que propulsaron la popularización de esta tecnología.

A pesar de aparecer las primeras ideas hace más de tres décadas con el trabajo de Scott Stornetta [9] en 1991 sobre una manera de verificar la fecha de creación de archivos digitales, no fué hasta que Satoshi Nakamoto publicó su trabajo sobre una moneda virtual (Bitcoin) apoyada en esta tecnología [10] en 2008 que la sociedad se percató de su potencial.

1.4.2. Elementos fundamentales de la Blockchain

Los componentes base de una aplicación que se apoya en la tecnología Blockchain pueden variar en función de las necesidades de la aplicación que se esté implementando. A continuación se describirán los elementos fundamentales de una Blockchain "clásica" basada en el algoritmo de prueba de trabajo, pues es el que se va a tratar a lo largo de la memoria.

Bloque

Los bloques son la unidad fundamental de la Blockchain. Para poder operar correctamente un bloque debe estar compuesto por (1) Uno o más datos de lo que se pretenda almacenar en la Blockchain, (2) una marca temporal de su creación, (3) un valor hash obtenido a partir del procesamiento de todos los demás datos que lo componen, (4) el valor hash del bloque previo, 0 en caso de ser el primer bloque y (5) el llamado "nonce" o marca, un valor variable que se manipula con objeto de obtener diferentes valores en el hash propio.

La manipulación de este campo "nonce" para manipular el hash resultante es la base del algoritmo de prueba de trabajo, y es la acción que se conoce como minado de un bloque. La forma objetivo del hash es lo que se conoce como dificultad de la Blockchain, y se trata a continuación.

Cadena de bloques

La cadena de bloques es lo que le da nombre a la tecnología, es una estructura de datos que almacena todos los bloques. Los bloques de la cadena deben tener hashes que concuerden con la dificultad de la misma, esto es, que sus hashes se modifiquen mediante la variación del campo nonce para cumplir ciertos requisitos como por ejemplo comenzar por cierto número de caracteres concretos.

Para que una cadena sea válida todos sus bloques tendrán que tener un hash válido con respecto a la dificultad marcada y un hash previo válido de su bloque anterior.

Nodo regular

Cada uno de los programas en ejecución que tienen una cadena de bloques, admiten sobre esta operaciones de minado y comunican con sus iguales sus diferentes modificaciones.

Nodo de bootstrap

De manera análoga a como ocurre con los trackers en las redes torrent, los nodos de bootstrap son servidores de baja carga y alta disponibilidad que no almacenan la propia cadena de bloques. Su única función es recibir peticiones de nodos regulares que acaban de iniciarse para facilitarles la información necesaria para conectarse a sus iguales y pasar así a formar parte de la red.

Consenso

Al conformar la red un número arbitrario de nodos que modifican una misma cadena, es necesario que existan algoritmos que coordinen estas modificaciones y determinen qué información es valida. El algoritmo de consenso más extendido entre las diferentes aplicaciones de la Blockchain es el de "cadena más larga", que considera siempre la cadena más larga como la correcta.

CAPÍTULO 2
Estado del arte

La popularización de la tecnología Blockchain, ha facilitado la creación de gran cantidad de materiales de aprendizaje, aunque no siempre de la mejor calidad. Parafraseando a Dettling [11], internet ofrece gran cantidad de pequeñas instrucciones, pero el entendimiento de la Blockchain viene dado por una visión de conjunto.

2.1 Materiales de aprendizaje disponibles

Utilizando como base la teoría educativa propuesta por David A. Kolb y Ron Fry, [12], puede juzgarse la calidad real de los materiales disponibles en función de la abundancia o falta de esta con respecto a los 4 diferentes tipos de aprendizaje que propone Kolb. Mientras que una rápida petición a cualquier motor de búsqueda devolverá gran cantidad de material de aprendizaje pasivo, esto es, sin implicación directa del interesado mas allá de ver, leer o escuchar, resulta más complicado encontrar materiales interactivos.

2.2 Propuesta

Uno de los materiales interactivos encontrados suple parte de las pretensiones de este proyecto, pero necesita de una ampliación para hacerlo al completo. Anders Brownworth comparte en su página personal [13] y de manera abierta el GitHub [14] de una aplicación que imita el comportamiento de la lógica de los bloques que conforman una Blockchain. A pesar de ser un gran recurso de carácter experimental, toda la información mostrada es simulada y no se apoya en tecnología Blockchain real, además de no permitir la edición de la cadena y centrarse en un único aspecto de esta.

Se propone pues, ampliar la propuesta de Brownworth de manera que se pueda ver y experimentar el estado interno de una cadena de bloques, pero apoyando esta representación en un modelo real.

CAPÍTULO 3
Análisis del problema

De la revisión realizada al estado del arte puede concluirse que, aunque existen buenas herramientas para el aprendizaje, no son adecuadas por lo limitado de su especificación. Una persona interesada en el desarrollo sobre Blockchain deberá saltar entre diferentes herramientas, que es posible que se diseñasen con distintos paradigmas en mente (no de manera incorrecta, pues una Blockchain admite multitud de implementaciones), lo que puede llevar a confusión en el aprendizaje.

Un aprendizaje puramente visual sin una funcionalidad real por detrás puede presentar dificultades para su ampliación y representación de un modelo real, y aún si lo hiciese correctamente, un desarrollador que ha entendido lo básico no podría basarse en el comportamiento que ha aprendido para comenzar a entender el código. Además, el no estar ligado a la lógica que pretende representar permite tomar libertades creativas que pueden llegar a ser demasiado abstractas.

De la misma manera, a pesar de ser el enfoque tradicional, adentrarse en el código directamente puede resultar intimidante sin un feedback visual en un primer momento para programadores novatos, lo que es un factor que debe eliminarse al introducir a alguien interesado en un nuevo campo. Ciertos estudios [15] concluyen que el feedback en tiempo real combinado con un sistema altamente interactivo mejoran las notas de estudiantes de programación, por lo que se pretende seguir en cierta manera el mismo camino.

3.1 Solución propuesta

Es así pues que se considera que existe una necesidad en el sector que puede suplirse mediante una suerte de combinación entre soluciones ya existentes.

Se propone una aplicación Blockchain real, aunque de nivel introductorio y tecnologías base, acoplada a una interfaz gráfica de dos fases. Por una parte, se mostraría la réplica de lo que sería una aplicación en producción que utiliza la tecnología Blockchain para su funcionamiento interno, mediante la cual el usuario interactuaría. Por otra parte, una vista que mostraría la representación del estado interno de los diferentes nodos que conforman la red y sus respectivas cadenas de bloques, que permiten ver reflejadas las acciones realizadas en la otra parte y la aplicación de los diferentes algoritmos y procesos entre las cadenas.

Las acciones que el usuario debe ser capaz de realizar desde la interfaz gráfica para cubrir cierta cantidad de casos experimentales son: elegir un nodo objetivo de entre los disponibles para realizar acciones, enviar un mensaje y minar un bloque que contenga un mensaje enviado previamente. Además, una funcionalidad esencial para hacer prue-

bas aunque no estándar en una aplicación al uso: minado local, es decir, dar la orden a un nodo de que mine un bloque pero no notificarlo a sus iguales. Esta última funcionalidad es esencial para poder crear los diferentes casos de uso que pueden aparecer por descordinaciones u otras causas.

Una sesión habitual de uso podría transcurrir de la siguiente manera: El usuario accedería a la aplicación gráfica y acto seguido iniciaría el servidor Blockchain. Con los dos sistemas principales encendidos, introduciría la dirección del sistema Blockchain en la interfaz gráfica y se conectaría a él. En este punto, aparecerían en pantalla todos los nodos conectados a la red, con los que el usuario podría interactuar. Para ello, en primer lugar, eligiría con que nodo de la red quiere interactuar y acto seguido introduciría y enviaría un mensaje en la réplica de la aplicación de producción, por ejemplo "hola mundo!". Al enviar el mensaje, el usuario debería ver en la parte de la interfaz que representa el estado de las cadenas de bloques de cada nodo que un nuevo mensaje listo para minar con el mensaje "hola mundo!" ha aparecido solo en el nodo que previamente había marcado para interactuar. Al pulsar el botón de "Minar", un nuevo bloque aparece en la cadena del nodo seleccionado y se expande al resto de las cadenas, todo ello visible a través de la interfaz gráfica. Además, el mensaje también habrá aparecido en la réplica de aplicación, pues forma parte de una cadena que se considera válida. El usuario ahora podría querer forzar una situación de conflicto, para ello repetiría las acciones anteriores con los mensajes "A" y "B", en nodos diferentes y clicando el botón de "Minar localmente" en lugar del de "Minar". Como consecuencia, habrán aparecido dos cadenas aparentemente válidas del mismo tamaño pero diferentes en la interfaz gráfica. Ahora el usuario podría volver a repetir la acción en el nodo que contiene el bloque con el mensaje "B", con el mismo mensaje pero esta vez pulsando el botón de "Minar" normal. El usuario vería entonces que todos los nodos se han actualizado con la nueva cadena más larga, tienen dos bloques con el mensaje "B" y el bloque que contenía el mensaje "A" se ha dado por invalido y ha desparecido.

3.2 Requerimientos funcionales

A partir de la solución que se ha propuesto y con objetivo de estructurar el desarrollo, pueden especificarse los siguientes requerimientos funcionales:

- El subsistema blockchain debe poder almacenar mensajes
- El subsistema blockchain debe poder generar nuevos bloques con los mensajes almacenados
- El subsistema blockchain debe poder almacenar las direcciones de sus iguales
- El subsistema blockchain debe poder ejecutar un algoritmo de consenso para validar la mejor cadena entre sus iguales
- El subsistema blockchain debe permitir que se obtenga su cadena a través de una interfaz de red
- El subsistema blockchain debe poder enviar y recibir notificaciones de que ha minado un nuevo bloque a subsistemas blockchain y de interfaz gráfica
- El subsistema de interfaz gráfica debe permitir seleccionar una instancia del subsistema blockchain con la que interactuar
- El subsistema de interfaz gráfica debe poder enviar una orden de crear un nuevo mensaje a una instancia del subsistema blockchain

- El subsistema de interfaz gráfica debe poder enviar una orden de minado de bloque a una instancia del subsistema blockchain
- El subsistema de interfaz gráfica debe poder enviar una orden de minado local de bloque a una instancia del subsistema blockchain

CAPÍTULO 4
Diseño de la solución

Las pretensiones con respecto al desarrollo son imitar una arquitectura y funcionamiento reales de una Blockchain, así como una forma de poder observar tanto el funcionamiento interno como sus repercusiones en un sistema externo. Para ello el sistema propuesto integra un núcleo de funcionalidad dado por una Blockchain al uso formada por un nodo de bootstrap y un número n de nodos regulares. Conectado a esta Blockchain se encontrará una aplicación web o webapp que mostrará el estado interno de los nodos (tanto regulares como de bootstrap) y permitirá interactuar con ellos mediante una réplica de una aplicación de chat.

4.1 Arquitectura del sistema

El sistema está formado por tres componentes que forman dos subsistemas conectados entre sí.

En primer lugar, y formando el que podría llamarse ”subsistema blockchain”, se encuentran el nodo de bootstrap (encargado de servir de puerta de entrada para coordinar nodos recién inicializados con el resto de la red) y un número n de nodos regulares. Por otra parte y como único componente del ”subsistema webapp” existe un servidor web. El nodo de bootstrap mantiene su tarea usual en la Blockchain de ser el primer nodo al que se conectan los nodos regulares para conseguir las direcciones de sus iguales y poder pasar a formar parte de la red. Pero, además, también le facilitará esta información a la webapp.

Puede observarse un esquema de la arquitectura descrita en la figura 4.1.

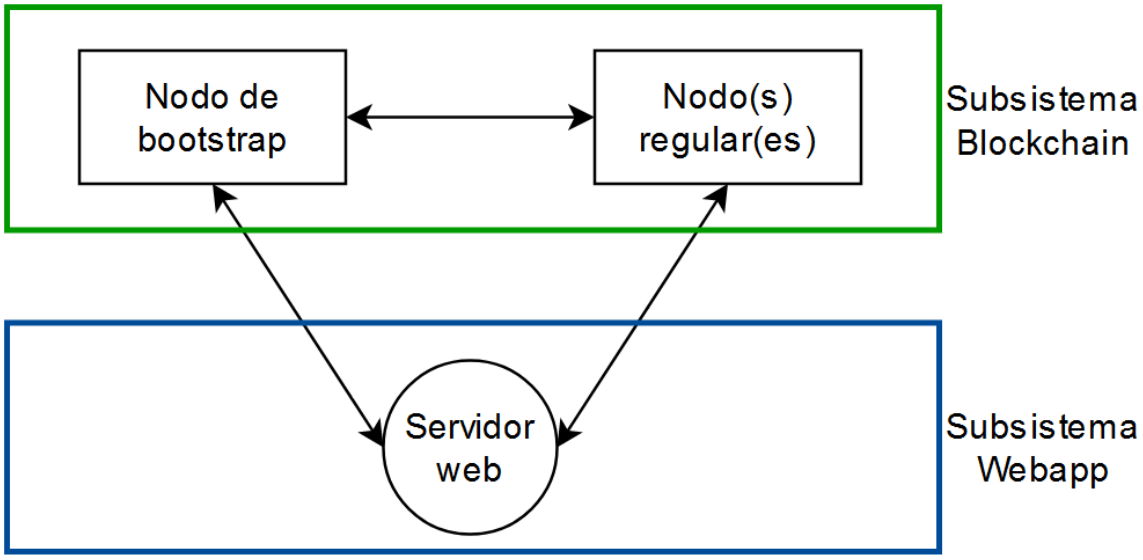

Figura 4.1: Arquitectura y sistemas principales

4.2 Diseño detallado

A continuación se describen de una manera más extensa las diferentes decisiones tomadas respecto al diseño tanto de los diferentes componentes que forman el sistema general como al de este mismo en su conjunto.

4.2.1. Subsistema blockchain

A pesar de buscar la simplicidad, reducir todo el comportamiento de la blockchain a un solo elemento elevaría el grado de abstracción haciendo más difícil aislar y entender qué funciones se llevan realmente a cabo. Por ello se ha decidido dividir el subsistema dos elementos que replicarán un nodo de bootstrap y un nodo regular.

Si bien es cierto que en una situación real podrían existir varios nodos de bootstrap para hacer la red más robusta, se considera que son comportamientos suficientemente alejados de la propia tecnología Blockchain y más relacionados con la ingeniería de redes y sistemas distribuidos. Es por esto que se ha decidido que la red a desarrollar esté formada por un solo nodo de bootstrap y n nodos regulares.

Los dos tipos de nodos mencionados deben ser capaces de comunicarse entre sí, y a su vez en tiempo real con un elemento externo (en este caso la webapp) que muestre sus estados internos. Por ello se proponen en diseño dos tipos de interfaces de red: una más clásica que no implica una conexión persistente, mediante URIs utilizada para coordinar los diferentes tipos de nodos entre sí, y una basada en sockets con la webapp para poder reflejar todos los estados y sus cambios en tiempo real.

La figura 4.2 representa el comportamiento definido para coordinar los diferentes tipos de nodos en este subsistema.

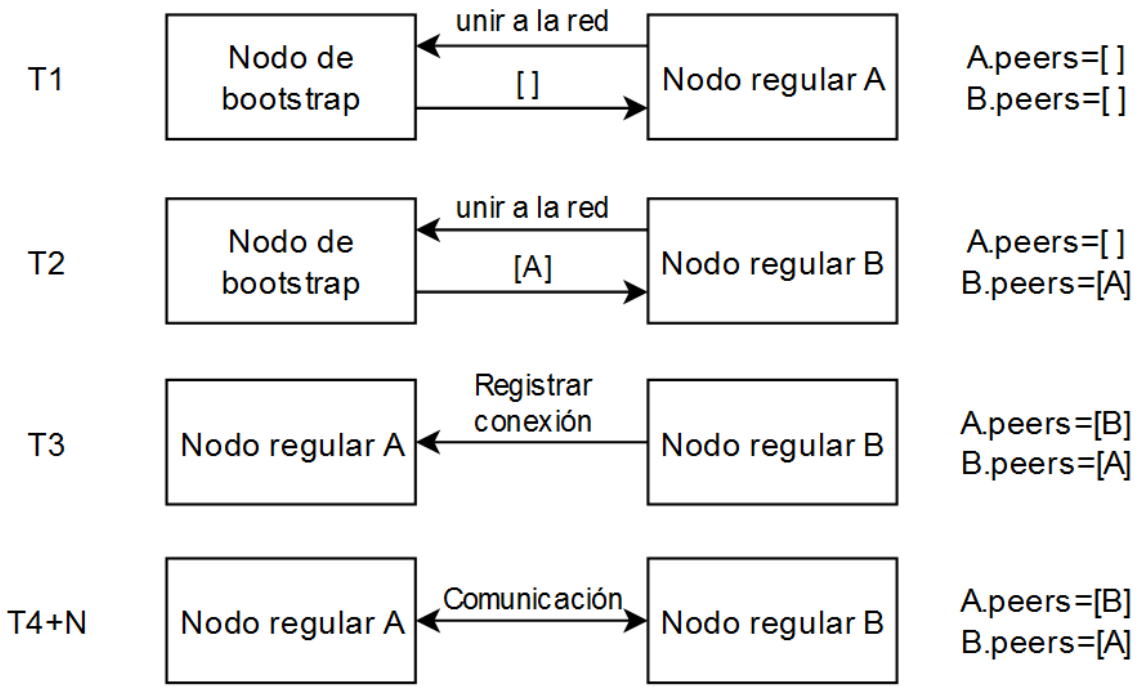

Figura 4.2: Arquitectura y sistemas principales

A continuación se hará una definición general de las clases y segmentos de código que los diferentes elementos del sistema deberán incorporar para implementar las funcionalidades planeadas.

No debe confundirse con el código final fruto del desarrollo, pues la implementación de los comportamientos que se definen necesitarán de lógica adicional, que aunque imprescindible para el correcto funcionamiento, será tangencial para el entendimiento de la estructura general del código, por lo que no se incluirá en esta sección.

Bootstrap node

Para conseguir la funcionalidad esperada en el nodo de bootstrap a la par que mantener el código legible, se ha decidido separar el código en 3 secciones:

- Clase que representa las diferentes conexiones.
- Segmento de código que gestiona las interfaces de conexión con nodos regulares.
- Segmento de código que gestiona las interfaces de conexión con webapps.

El deber del nodo de bootstrap es tener una lista de los nodos regulares que se conectan a él, para poder comunicársela a los siguientes que también decidan conectarse. También debe llevar un recuento de las webapps conectadas, para comunicarles los cambios en los nodos conectados. Además, debe contemplarse que tanto nodos regulares como webapps pueden desconectarse.

Teniendo en cuenta estos requisitos, la clase principal del archivo que contiene al nodo de Bootstrap estará definida tal y como se muestra en el listado 4.1.

```
class Bootnode:
   def __init__(self):
      """
      Inicializacion de las variables que almacenaran las direcciones IP
      tanto de los nodos  regulares como de las webapps que se ha conectado
      al nodo de bootstrap
      """

   def add_regular_node(self, ip):
      """
      Guardar la direccion IP de un nodo regular que ha decidido conectarse
      al bootnode.
      Esta accion debera notificarse a todas las webapps conectadas.
      """

   def remove_regular_node(self, ip):
      """
      Eliminar la direccion IP guardada de un nodo regular que ha decidido
      desconectarse del bootnode.
      Esta accion debera notificarse a todas las webapps conectadas.
      """

   def add_webapp(self, ip):
      """
      Guardar la direccion IP de una webapp que ha decidido conectarse
      al bootnode.
      Esta accion debera notificarse a todas las webapps conectadas.
      """

   def remove_webapp(self, ip):
      """
      Eliminar la direccion IP guardada de una webapp que ha decidido
      desconectarse del bootnode.
      Esta accion debera notificarse a todas las webapps conectadas.
      """
```

Listado 4.1: Plantilla para la clase principal del nodo de bootstrap

Una vez cubierto el comportamiento principal, solo queda definir las interfaces mediante las cuales podrá comunicarse tanto con nodos regulares como con webapps.

En primer lugar, las conexiones con los nodos regulares se definen como se muestra en el listado 4.2. Se utiliza el decorador del paquete Flask para que la función definida sea accesible una vez se inicialice el programa como servidor.

```
@app.route("/join-regular", methods=["POST"])
def join_bootnode_regular_node():
   """
   Recupera la informacion de conexion facilitada en el POST
   por el nodo regular y lo incluye en la lista de direcciones.
   """
```

Listado 4.2: Plantilla para definir una interfaz de comunicación de red con Flask

Por último, las conexiones con los webapp se estableceran utilizando sockets, por tanto se definen los eventos a los que se debe reaccionar tal y como se ve en el listado 4.3.

```
@socketio.on('connect')
def webapp_connected_socket_handler():
    """
    Prepara la gestion del evento de conexion en el socket.
    Incluye la webapp en la lista de direcciones y le manda a todas
    las webapps conectadas, la nueva incluida, el nuevo estado de la
    lista de direcciones.
    """

@socketio.on('disconnect')
def webapp_disconnected_socket_handler():
    """
    Prepara la gestion del evento de desconexion en el socket.
    Elimina la webapp de la lista de direcciones y le manda a todas
    las webapps aun conectadas el nuevo estado de la lista de direcciones.
    """
```

Listado 4.3: Plantilla para definir eventos de conexión con Flask-SocketIO

Es importante remarcar que todas las operaciones de añadir y eliminar direcciones se realizarán siempre a través de los métodos definidos en la clase Bootnode y nunca directamente, con tal de poder notificar a las webapps de cada cambio sin tener que repetir código para ello.

Regular node

El nodo regular tiene algo más de complejidad que el nodo de boostrap. Se ha organizado su código en 7 secciones principales:

- Clase que representa a un mensaje.
- Clase que representa a un bloque.
- Clase que representa la Blockchain.
- Clase que representa las diferentes conexiones.
- Segmento de código que gestiona las interfaces de conexión con nodos de bootstrap.
- Segmento de código que gestiona las interfaces de conexión con nodos regulares.
- Segmento de código que gestiona las interfaces de conexión con webapps.

La clase Mensaje no precisa de mucha explicación, la propia clase solo compuesta por su método de inicialización proporciona una forma de estructurar la información, tal y como se muestra en el listado 4.4.

```
class Message:
    def __init__(self, author, content, timestamp):
        """
        Inicializacion de las variables
        """
```

Listado 4.4: Plantilla para la clase Message

Por el contrario la clase Bloque, a pesar de también ser muy secilla, si que necesita de cierta funcionalidades relacionadas con los calculos que se le precisan, los cuales aparecen en el listado 4.5.

Cabe remarcar que se ha tomado la decisión que cada bloque contenga un solo mensaje, pues se considera que la idea de múltiples transacciones añade una complejidad no necesaria a la hora de entender el funcionamiento base de la tecnología Blockchain.

```
class Block:
   def __init__(self, block_num, message, timestamp, previous_hash, nonce=0):
      """
      Inicializacion de las variables
      """

   def compute_hash(self):
      """
      Calcula el hash del bloque a partir del resto de sus atributos
      """

   @staticmethod
   def confirm_hash(block):
      """
      Recibe un bloque al que ya se le ha asignado un hash y comprueba
      si es correcto
      """
```

Listado 4.5: Plantilla para la clase Block

La clase Blockchain es la más compleja de todo el nodo regular, y gestiona las diferentes operaciones a realizar con un Mensaje o Bloque. Se definen las especificaciones de los métodos en los listings 4.6 y 4.7.

```
class Blockchain:
   """
   Se define el valor de dificultad para el minado
   """
   difficulty = 2

   def __init__(self):
      """
      Inicializacion tanto de la estructura de datos que contendra
      como la que contendra los mensajes sin minar y la cadena como
      de el bloque origen, que se minara y agregara a la cadena.
      """

   def add_block(self, block, proof):
      """
      Recibe un bloque, comprueba su validez y en caso de ser valido
      lo agrega a la cadena.
      En caso de agregarlo, notifica a las webapps conectadas del
      cambio de estado.
      """

   def fifo_add_unmined_message(self, message):
      """
      Agrega un nuevo mensaje sin minar al final de la lista y notifica
      de ello a las webapps conectadas.
      """

   def fifo_pop_unmined_message(self):
      """
      Elimina el mensaje sin minar mas antiguo de la lista y notifica
      de ello a las webapps conectadas.
      """

   def is_valid_block(self, block, proof):
      """
      Comprueba si un bloque dado es valido con respecto a la Blockchain
      local.
      """

   @classmethod
   def is_valid_proof(cls, block, proof):
      """
      Comprueba si la prueba de trabajo de un bloque dado es valida,
      par ello comprueba tanto si la dificultad del hash del bloque
      coincide con la definida y si este hash se ha calculado
      correctamente.
      """

   @staticmethod
   def proof_of_work(block):
      """
      Prueba por fuerza bruta diferentes valores para el atributo
      nonce del bloque proporcionado hasta encontrar uno que
      proporcione un hash adecuado para con la dificultad
      establecida.
      """
```

Listado 4.6: Plantilla para la clase Blockchain, parte 1

```
@staticmethod
def check_chain_validity(chain):
    """
    Comprueba si todos los bloques que conforman una cadena dada
    son validos para confirmar la legitimidad de la misma.
    """

def mine(self):
    """
    En caso de no estar vacia la lista de mensajes sin minar, se
    inicializa un nuevo bloque que contiene al mensaje sin minar
    mas antiguo, se realiza la prueba de trabajo y se agrega a la
    Blockchain.
    """
```

Listado 4.7: Plantilla para la clase Blockchain, parte 2

En lo referente a la clase que gestiona las diferentes conexiones, se utiliza la misma especificación que en el listado 4.1, con la única diferencia de que no se notificará a las webapps conectadas de los cambios realizados en estas estructuras de datos, pues no es información que se planee mostrar.

La comunicación con el nodo de bootstrap viene dada por las funcionalidades declaradas en el listado 4.8.

```
"""
A pesar de solo existir uno, se imita la estructura de datos de
una aplicacion real, en la que existen una serie de nodos de
bootstrap por defecto.
"""
bootnodes = ["127.0.0.1:5000"]

def getPeersFromBootnodes(bootnodes, blockchain, peers):
    """
    El nodo intenta conectar con uno de los nodos de bootstrap
    por defecto, y agrega la lista de nodos regulares existentes
    en la red en su lista de iguales
    """
```

Listado 4.8: Plantilla de comunicacion entre nodos regulares y nodos de bootstrap

Existen casos más particulares en la conexión entre nodos regulares que entre un nodo regular y un nodo de bootstrap, los cuales tratan principalmente el establecimiento de una conexión bidireccional y el intercambio de datos relativos a sus cadenas de bloques. El listado 4.9 muestra sus especificaciones.

```
def consensus(blockchain, peers):
    """
    Comprueba con la lista de iguales si el propio nodo tiene una
    cadena de bloques mayor o de igual longitud que el resto de
    nodos regulares. De no ser asi, la sustituye por la mayor de
    entre sus iguales.
    """

def announce_new_block(block):
    """
    Anuncia un bloque a sus iguales para que lo agreguen a sus cadenas.
    """

@app.route('/add-block', methods=['POST'])
def network_add_block():
    """
    Prepara la interfaz para que los nodos regulares puedan comunicar
    nuevos bloques entre si.
    """

@app.route('/chain', methods=['GET'])
def get_chain():
    """
    Prepara la interfaz para que los nodos regulares puedan obtener la
    cadena de bloques de sus iguales.
    """

@app.route('/register-new-peer', methods=['POST'])
def add_new_regular_node_to_peers():
    """
    Prepara la interfaz para que en caso de un nodo regular A conocer
    la direccion de un nodo regular B, pueda hacer una peticion con su
    propia direccion con la intencion de que B conozca la direccion de A
    """
```

Listado 4.9: Plantilla de comunicacion entre nodos regulares

En la conexión de los nodos regulares con las webapps se definen las interacciones mencionadas previamente en la solución propuesta. Todas estas interacciones se realizan a través de sockets y son las mostradas en el listado 4.10.

```
@socketio.on('connect')
def webapp_connected_socket_handler():
    """
    Prepara la gestion del evento de conexion en el socket.
    Incluye la webapp en la lista de direcciones.
    """

@socketio.on('disconnect')
def webapp_disconnected_socket_handler():
    """
    Prepara la gestion del evento de desconexion en el socket.
    Elimina la webapp de la lista de direcciones.
    """

@socketio.on('new_unmined_message')
def webapp_new_message(data):
    """
    Prepara la gestion del evento de envio de mensaje en el socket.
    Utiliza la informacion proporcionada por el usuario para agregar
    un nuevo mensaje a la lista y notifica a todas las webapps conectadas
    de ello.
    """

@socketio.on('mine')
def webapp_mine():
    """
    Prepara la gestion del evento de minado en el socket.
    Realiza consenso, ejecuta el comportamiento de minado previamente
    definido en la clase Blockchain y notifica al resto de iguales del
    nuevo bloque.
    """

@socketio.on('mine-locally')
def webapp_mine_locally():
    """
    Prepara la gestion del evento de minado local en el socket.
    Igual que en la gestion del evento de minado, pero ni realiza
    consenso ni notifica a sus iguales del nuevo bloque.
    """
```

Listado 4.10: Plantilla de comunicacion entre nodos regulares y webapps

4.2.2. Subsistema webapp

La aplicación web que se plantea consiste en una sola página que permite ver el estado de las cadenas de bloques de todos los nodos regulares conectados, el estado de las conexiones activas del nodo de bootstrap a la par que la capacidad de interactuar con ambos. Con tal fin se plantean los diseños expuestos a continuación.

Interfaz gráfica

Ya que el objetivo principal de la interfaz es relacionar las consecuencias que ciertas acciones en una web al uso tienen en el estado de la tecnología que la sustenta, cualquier diseño que no consistiese en una sola página dificultaría relacionarlos.

Por ello se propone la disposición de la figura 4.3, dedicándose, en un ratio 1:3, la parte izquierda de la aplicación a una réplica de un sencillo foro y la derecha a una suerte de centro de control interno junto a una representación gráfica del estado de las cadenas de bloques de los nodos que componen la red.

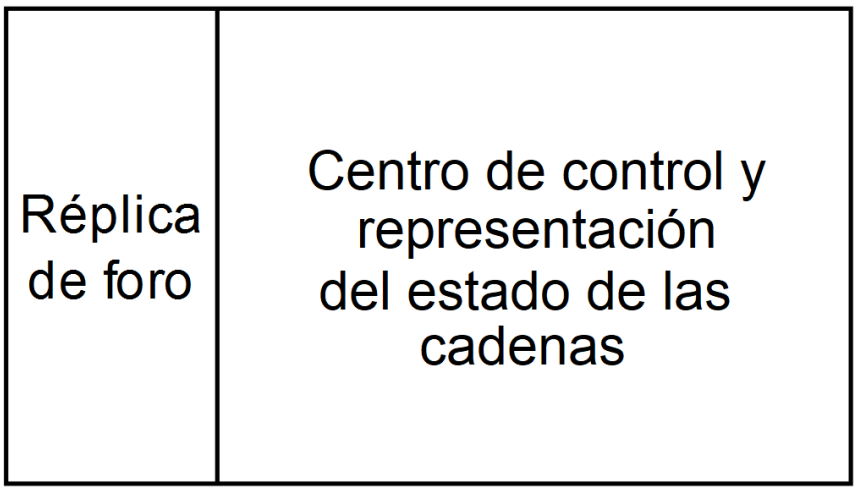

Figura 4.3: Esquema general del layout de la webapp

Entrando en más profundidad, la réplica del foro se dividiría en un cuadro de entrada de texto con opción de enviar un mensaje en la parte superior y un contenedor que muestre los mensajes guardados en la cadena más larga, como se ejemplifica en la figura 4.4.

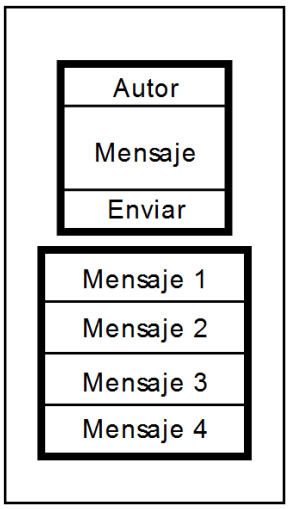

Figura 4.4: Esquema general del layout de la réplica del foro

Por su parte, la sección derecha de la pantalla se compondría de un centro de mandos en la parte superior que permitiría interactuar con los diferentes nodos tanto de bootstrap como regulares, y de un contenedor con representaciones de las cadenas de bloques de los nodos regulares conectados en la parte inferior, como muestra la figura 4.5.

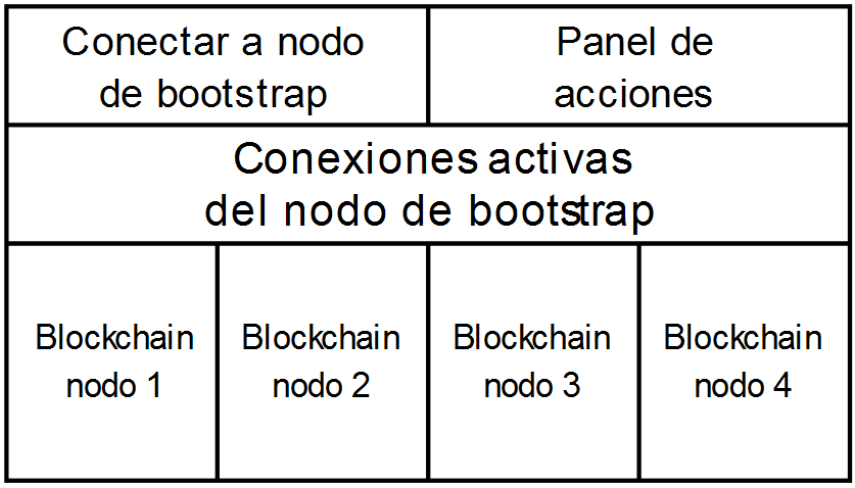

Figura 4.5: Esquema general del layout de centro de mandos y representación de las Blockchains

Lógica

En cuanto a la funcionalidad, se ha planificado separar la lógica en 3 ficheros de JavaScript con diferentes contextos, con llamadas monodireccionales, con la intención de facilitar el seguimiento del flujo de información, como se observa en el esquema de la figura 4.6.

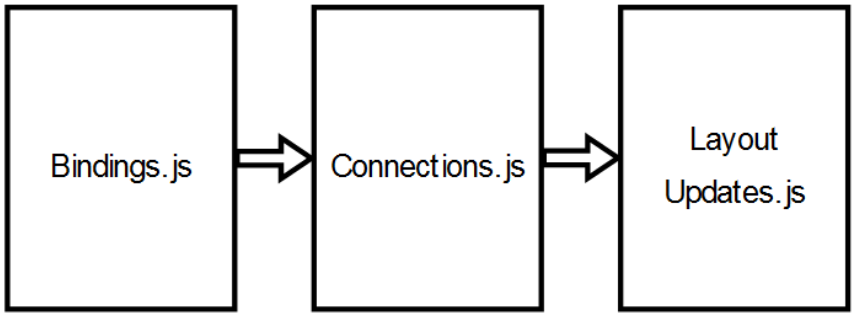

Figura 4.6: Relaciones de los scripts de la webapp

En cada uno de los ficheros que corresponden a estos scripts, y con tal de facilitar la trazabilidad y legibilidad del código, se ha seguido la convencención de añadir a todas las funciones y variables globales un prefijo igual al nombre del fichero y una barra baja o la ausencia de ella que representa si puede o no referenciarse desde otro fichero. En los esquemas de lógica que siguen solo se mostrarán los métodos públicos referenciables por ficheros externos, pues se consideran suficientes para representar la intencionalidad y estructura del código a pesar de ser una pequeña parte de este.

El primer script en el flujo de información sería Bindings, encargado de conectar elementos de la interfaz a los métodos correspondientes de la parte de conexiones. Se ha decidido no referenciar las funciones directamente en el HTML por claridad. No se considera necesario presentar una plantilla de este script ya que será muy breve y únicamente lo compondrán llamadas de una línea para relacionar elementos del DOM con funciones del fichero Connections.

Connections se encarga de implementar todos los comportamientos que debe presentar la aplicación, pues todas las interacciones que el usuario puede tener con la página implican una comunicación. Aquí se establecen los sockets de conexión con los nodos correspondientes y se gestionan los datos recibidos. Se comunica directamente con La-

youtUpdates. Se da una especificación general de la estructura de su lógica en el listado 4.11.

```
/**
* Se almacenan referencias tanto al socket del nodo de bootstrap
* como a los de los nodos regulares y sus direcciones IP correspondientes
* para poder utilizarlas en
*/

let _connections_bootnode_socket;
let _connections_regular_node_sockets = {};

function connections_connect_to_bootnode() {
   /**
   * Se crea un socket de conexion con la direccion IP del nodo
   * de bootstrap introducida en el campo correspondiente de la
   * interfaz grafica.
   */
}

function connections_selected_regular_node_mine() {
   /**
   * Envia un evento de minado por el socket del nodo regular
   * seleccionado como objetivo.
   */
}

function connections_selected_regular_node_mine_locally() {
   /**
   * Envia un evento de minado local por el socket del nodo regular
   * seleccionado como objetivo.
   */
}

function connections_selected_regular_node_new_unmined_message() {
   /**
   * Envia un evento de nuevo mensaje por el socket del nodo regular
   * seleccionado como objetivo.
   */
}

//Informacion relativa a los sockets

/**
* En la definicion de los sockets inicializados se gestionaran los eventos
* de recepcion de nueva informacion para llamar a las funciones
* correspondientes de LayoutUpdates.js
*
* Ademas, el socket del nodo de bootstrap se configurara para crear
* los nuevos sockets de conexion con nodos regulares cuando lleguen
* a traves de el las direcciones IP de estos
*/
```

Listado 4.11: Plantilla de lógica del fichero Connections.js

Por último, LayoutUpdates como su nombre indica es el script responsable de actualizar la información mostrada en pantalla en función de los datos que recibe de Connections. Sus principales métodos, referenciados por Connections se muestran en el listado 4.12.

```
/**
* Se almacena la informacion sobre los nodos regulares con sockets
* abiertos para poder trabajar sobre ella en lugar de reinicializar
* cada vez.
*/

let _layout_regular_nodes_data = {};

function
    layout_bootnode_update_active_regular_nodes(active_regular_node_ip_list) {
   /**
   * Actualiza la lista de nodos regulares conectados al nodo
   * de bootstrap.
   */
}

function layout_bootnode_update_active_webapps(active_webapp_ip_list) {
   /**
   * Actualiza la lista de webapps conectadas al nodo de bootstrap.
   */
}

function layout_regular_node_update_chain(regular_node_socket_id,
    regular_node_socket_ip, regular_node_updated_chain) {
   /**
   * Actualiza los bloques que conforman la cadena de un nodo regular
   * dado.
   */
}

function layout_regular_node_update_unmined_messages(regular_node_socket_id,
    regular_node_socket_ip, regular_node_unmined_messages) {
   /**
   * Actualiza los mensajes sin minar que almacenados en un nodo regular
   * dado.
   */
}
```

Listado 4.12: Plantilla de lógica del fichero LayoutUpdates.js

4.2.3. Ejecución

Se ha considerado necesario proporcionar una forma de ejecutar la aplicación para aquellos interesados únicamente en su utilización y no en el análisis de su código, extensión o implementación. Es por ello que se ha decidido construir un contendor Docker que contenga e inicialice todos los componentes del sistema con tal de que sea posible ejecutarlo sin tener que tener en cuenta instalaciones, dependencias y ejecuciones por terminal de los archivos y servidores necesarios para lanzar la aplicación al completo.

Puede observarse un esquema del lanzamiento de los servicios en la figura 4.7.

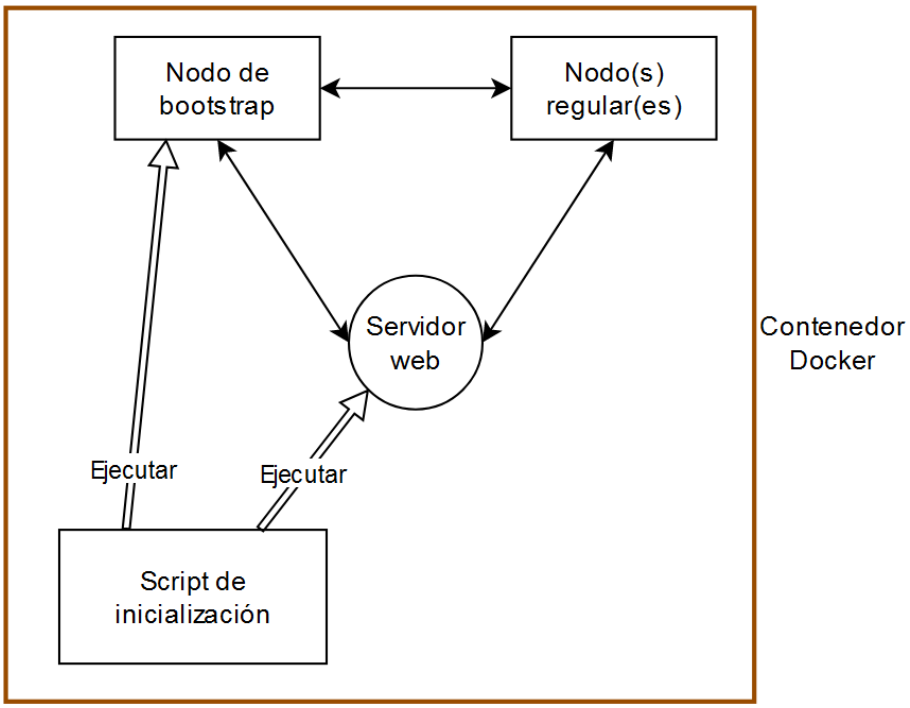

Figura 4.7: Arquitectura para una ejecución sencilla

4.3 Tecnología utilizada

En concordancia con el resto del proyecto, se ha buscado facilitar el camino a aquel usuario con un interés en el apartado técnico del proyecto, buscando tecnologías que necesiten el mínimo trabajo para comenzar a utilizarlas y de las que abunde la documentación.

- **HTML, CSS y JavaScript:** Tecnología estándar de construcción de webs. No se ha considerado necesaria la utilización de frameworks o metalenguajes de hojas de estilo para una web estática compuesta de una sola página.
  - **JavaScript - SocketIO:** Se ha utilizado este módulo como implementación de los sockets especificados como necesidad para que la webapp mostrase los estados de los nodos en tiempo real. Existe gran cantidad de documentación al respecto y es sencillo de implementar, además de coordinarse muy bien con la solución para implementar los sockets en los servidores (nodos) Python de la que se hablará a continuación.
- **Python:** Un lenguaje interpretado con una sintaxis simple y amplia cantidad de paquetes compatibles con la tecnología objeto de este proyecto han resultado le elección para la funcionalidad del "subsistema Blockchain". Además, existen gran cantidad de proyectos a tomar como referencia [16].
  - **Python - Flask:** Microframework utilizado para la implementación de la comunicación entre nodos. De nuevo, la amplia documentación y la sencillez de desarrollo lo convierten en la elección ideal para el proyecto.
  - **Python - Flask-SocketIO:** Paquete que coordina utilidades de dos tecnologías ya especificadas, perfecto para implementar los sockets por parte del servidor sin añadir complejidad extra al desarrollo.

- **Python - SimpleHTTPServer:** Paquete incluido con python3 que cubre completamente las necesidades requeridas para levantar una página web localmente. Ser capaz de hacer esto sin configuración previa y con un solo comando han sido los criterios utilizados para decidir su uso como servidor web.
- **Python - Eventlet:** Paquete que cubre una necesidad creada por Flask-SocketIO: no se soportan los websockets por defecto y se revierte al long polling. Elegido por ser el que mayor rendimiento ofrece con Flask-SocketIO y solo requiere su instalación y un par de líneas de código para que todo funcione como se ha planificado.

Es también mediante Python y sus paquetes Flask y SimpleHTTPServer que se ha implementado la comunicación entre nodos y el servidor web que sirve la webapp respectivamente.

- **Docker:** Esencial para que aquellos sin conocimientos técnicos puedan ejecutar la aplicación y utilizarla sin tener considerar dependencias o instalaciones.

CAPÍTULO 5
Desarrollo de la solución propuesta

El desarrollo de la solución propuesta se ha realizado sin mayor problema implementando las bases asentadas en el capítulo de diseño. No se incluyen las propias implementaciones por no alejarse de las que podrían suponerse de las especificaciones previamente expuestas, pero sí que se comentará la metodología de implementación seguida y un par de errores con sus respectivas soluciones que no fueron consideradas en la fase de diseño.

En lo relativo al testeo, dado el objetivo pedagógico del sistema desarrollado y la complejidad asociada al comprobar todas las situaciones de un programa distribuido multihilo formado por diferentes subsistemas, se considera que una comprobación exhaustiva siguiendo estrictos estándares de verificación queda fuera del ámbito de este proyecto.

5.1 Metodología de implementación

Se ha considerado que al ser una aplicación relativamente sencilla y rápida de implementar, además de no tener la necesidad de coordinar a un equipo, seguir una metodología compleja resultaría en un gasto temporal contraproducente.

A pesar de ello, es necesario estructurar el trabajo para medir la progresión y planificar el desarrollo, por lo que se ha seguido una metodología cuasi Scrum, con una tabla de requisitos inicial (figura 5.1) y objetivos y revisiones semanales.

Área de requisitos	Requisitos
Sistema Global	Investigación sobre tecnologías a utilizar
Subsistema Blockchain	Diseño de clases
Sistema Global	Diseño de comunicación
Subsistema Webapp	Diseño de la interfaz gráfica
	Semana 1 (Investigación y trabajo previo)
Subsistema Blockchain	Operaciones sobre las listas de nodos en el nodo de bootstrap
Subsistema Blockchain	Operaciones sobre la cadena de bloques en el nodo regular
Subsistema Webapp	Implementación de la interfaz gráfica de la webapp
Subsistema Webapp	Lógica de modificación de la interfaz gráfica
	Semana 2 (Lógica interna)
Subsistema Blockchain	Interfaces de comunicación con nodos regulares en el nodo de bootstrap
Subsistema Blockchain	Interfaces de comunicación con la webapp en el nodo de bootstrap
Subsistema Blockchain	Interfaces de comunicación con nodos regulares en el nodo regular
Subsistema Blockchain	Interfaces de comunicación con la webapp en el nodo regular
Subsistema Webapp	Interfaces de comunicación con nodos de bootstrap en la webapp
Subsistema Webapp	Interfaces de comunicación con nodos regulares en la webapp
	Semana 3 (Interfaces de comunicación)
Sistema Global	Test de conexión entre cada componente
Sistema Global	Test de funcionalidades
Sistema Global	Depuración de errores
	Semana 4 (Testeo)

Figura 5.1: Tabla de requisitos y planificación temporal de la implementación

5.2 Errores de coordinación de estado entre nodos de bootstrap y nodos regulares

Los nodos de bootstrap deben tener conocimiento de los nodos regulares que se han conectado a ellos y aún se encuentran activos. De no conocer exactamente qué nodos se encuentran activos y cuáles se han desconectado, al preguntar un nodo regular por su lista de iguales para poder coordinarse con uno e inicializarse, este podría recibir nodos inactivos en el conjunto. Si bien esto no es realmente un problema funcional porque los nodos regulares comprueban la lista recibida hasta encontrar un nodo activo, esta característica implicaba ”ensuciar” la lista de nodos regulares que se mostraría a su vez en la interfaz gráfica de la webapp.

Al haber implementado la comunicación entre nodos mediante interfaces Flask por URL en lugar de por sockets, no existía manera de saber si un nodo regular se había desconectado. Una solución podría haber sido añadir un comportamiento de cierre definido en los casos regulares, pero seguiría sin ejecutarse en caso de un cierre abrupto por un error o un corte de luz.

Se planteó la idea de implementar las conexiones entre nodos también mediante sockets, pero la comunicación de sockets servidor a servidor con la tecnología elegida para la implementación aparentaba una elevada complejidad que no casaba con el proyecto. Es por esto que se resolvió utilizar una sencilla solución de polling.

El nodo regular implementaría una interfaz /alive como se muestra en el listado 5.1 la cual consultaría periódicamente el nodo de bootstrap, tal y como implementa el código del listado 5.2, y en caso de no recibir respuesta, eliminaría al nodo regular de la lista de nodos activos. Se planteó también la posibilidad de añadir un sistema de "confianza" que puntuase las veces que un nodo conocido se conectaba y desconectaba para evitar comunicarlo como activo o aumentar el periodo, con tal de mejorar el servicio y bajar el coste computacional. Esta idea se descartó por no ser un concepto esencial de la blockchain, la heurística aplicada a la hora de considerar la implementación de nuevos desarrollos o tecnologías.

```
"""
Se define la ruta y los metodos por los cuales se accedera de manera
remota a la funcion alive_beacon.
"""
@app.route('/alive', methods=['GET'])
   def alive_beacon():
   return json_response(True, "Connection still alive")
```

Listado 5.1: Interfaz /alive en el nodo regular

```
class Bootnode:
   def __init__(self):
      #...

      """
      El metodo revisa la lista interna de nodos regulares que se
      han unido al nodo de bootstrap, realiza una peticion a la
      interfaz /alive y en caso de no recibir respuesta, elimina
      al nodo de la lista. Despues, el hilo duerme durante el
      intervalo especificado
      """
      def purge_inactive_regular_nodes(self, purge_interval):
         while True:
            for regular_node in self.regular_nodes_connected:
               try:
                  requests.get("http://{}/alive".format(regular_node))
               except:
                  self.remove_regular_node(regular_node)

            time.sleep(purge_interval)
```

Listado 5.2: Metodo de consulta de nodos regulares desde nodo de bootstrap

Un último e importante detalle a considerar de esta solución es que un bucle continuo, como es de esperar, bloquea el hilo de ejecución. Afortunadamente, la ejecución multihilo es suficientemente sencilla en Python como para no perturbar la esencia del proyecto. Es

a través de la lógica mostrada en el listado 5.3, que queda resuelto el problema de la concurrencia.

```
class Bootnode:
   def __init__(self):
   #...

   """
   Se inicia el proceso objetivo como un nuevo hilo, se le pasa
   como parametro el intervalo de tiempo entre el que tiene que
   purgar los nodos inactivos y se define como demonio para que
   en caso de finalizar la ejecucion del proceso principal, termine
   la de este tambien.
   """
   self.purge_interval = 1
   Thread(target=self.purge_inactive_regular_nodes, args=(self.purge_interval,),
       daemon=True).start()

   #...
```

Listado 5.3: Inicialización de ejecución multihilo

5.3 Concurrencia en la búsqueda de nodos de bootstrap

Dado que hasta contactar con uno de los nodos de bootstrap iniciales un nodo regular se encuentra completamente incomunicado, se consideró importante que este proceso fuese un bucle indefinido que hiciese las veces de guarda para la inicialización del nodo y las interfaces de comunicación.

En esta ocasión, al contrario que en el anterior error, que el bucle frenase la ejecución era un comportamiento deseado, pero la existencia de llamadas sleep para evitar un uso desmesurado de la CPU interfería en caso de querer detener el programa. Es por esto que se decidió implementar de igual manera una ejecución multihilo, pero esta vez se hizo uso de una llamada join para esperar a salir del bucle para poder seguir la ejecución en el hilo principal, tal y como se muestra en los listings 5.4 y 5.5.

```
#...

"""
Para cada nodo de bootstrap facilitado, intenta unirse a la red enviandole
su direccion IP y puerto.
En caso de no haber nodos de bootstrap, la ejecucion para durante un segundo.
En caso de recibir una error en la conexion, se salta el nodo de bootstrap.
En caso de recibir una confirmacion de que se ha unido al nodo de bootstrap,
exitosamente, lo comunica por la consola.
A continuacion intenta agregarse mutuamente con cada uno de los nodos
regulares activos que le ha proporcionado el nodo de bootstrap.
En caso de error, descarta ese nodo regular.
Una vez terminado el proceso, realiza consenso con los nodos regulares con
los que se ha conectado exitosamente, para inicializarse con la cadena mas
larga de la red.
"""
def getPeersFromBootnodes(bootnodes, blockchain, peers):
   while True:
      for bootnode in bootnodes:
         try:
            regular_node_port_json = {"port": regular_node_port}
            response = requests.post("http://{}/join-regular".format(bootnode),
                json=regular_node_port_json)
            if response.ok:
               response_data = response.json()
               if response_data["success"] == True:
               print("Connected!")
               for peer in response_data["data"]["peers"]:
                  try:
                     requests.post("http://{}/register-new-peer".format(peer),
                         json={'port':regular_node_port})
                     peers.append(peer)
                  except:
                     print("Error adding regular node {} to peers,
                         skipping...".format(peer))
               consensus(blockchain, peers)
               return
            else:
               print("Error connecting to bootnode {},
                   skipping...".format(bootnode))
               time.sleep(1)
         except:
            print("Waiting for bootnodes...")
            time.sleep(1)
```

Listado 5.4: Solución al problema de concurrencia buscando nodos de bootstrap, parte 1

```
    """
    Se inicializan las variables que se le van a proporcionar al metodo que
    se va a ejecutar en otro hilo.
    """
    blockchain = Blockchain()
    bootnodes = ["127.0.0.1:5000"]
    peers = []

    """
    En primer lugar se fija el anterior metodo como objetivo para la ejecucion
    multihilo, sus argumentos y se establece como demonio para que su ejecucion
    termine a la par con el proceso principal.
    A continuacion, inicia el hilo.
    Por ultimo, espera a que la ejecucion del hilo termino para seguir con la
    ejecucion del hilo principal.
    """
    peer_discovery_thread = Thread(target=getPeersFromBootnodes,
        args=(bootnodes, blockchain, peers,), daemon=True)
    peer_discovery_thread.start()
    peer_discovery_thread.join()

    #...
```

Listado 5.5: Solución al problema de concurrencia buscando nodos de bootstrap, parte 2

5.4 Dependencias surgidas del uso de websockets

El servidor por defecto implementado por el paquete Flask es un servidor de desarrollo, y como tal, no soporta ciertas funcionalidades. Una de estas funcionalidades es el uso de websockets, y su uso revertía al long polling por parte de socket-io. La principal solución a este problema fue instalar el gestor de eventos eventlet, un gestor de eventos que proporciona un servidor que sí que es capaz de aceptar conexiones websocket.

El problema viene dado por la incompatibilidad de eventlet con los mecanismos multihilo por defecto de Python, por lo que entraba en conflicto con las soluciones planteadas a los dos problemas anteriores. La resolución de este problema pasa por utilizar la función que eventlet ofrece para parchear las ejecuciones multihilo de un programa, adaptándolas a su modelo de ejecución, tal y como se muestra en el código referenciado en el listado 5.6, ubicado al principio del fichero de ambos nodos.

```
"""
Intenta importar el paquete de eventlet y en caso de hacerlo con exito,
parchea la ejecucion multihilo en el programa actual.
En caso de fallar la importacion, se mantiene el modo por defecto de
ejecucion, pero los websockets funcionaran por long polling en lugar
de por sockets verdaderos.
Ademas, comunica por pantalla si la importacion ha sido exitosa.
"""
async_mode = None

if async_mode is None:
   try:
      import eventlet
      async_mode = 'eventlet'
      eventlet.monkey_patch()
   except ImportError:
      async_mode = 'threading'

print('async_mode is ' + async_mode)
```

Listado 5.6: Solución al problema de utilización de long polling en lugar de websockets

CAPÍTULO 6
Demostración

Se ha creado un flujo de ejecución compuesto de casos de uso, los considerados suficientes y necesarios sin caer en repetición para hacer las veces de una pequeña guía dividida en 3 secciones que ejemplifica las capacidades del sistema desarrollado.

Estas secciones consisten en la conectividad de los diferentes subsistemas que conforman el proyecto, el uso de la webapp que es la interfaz que el usuario utilizará para interactuar con el mismo y un breve apartado que referencia el comportamiento puramente visual de la interfaz en caso de desbordar ciertos datos su contenedor asignado.

6.1 Conexiones

Se juzgará la correcta conexión entre los tres componentes principales (la webapp, el nodo de bootstrap y los nodos regulares) mediante la sección de información del nodo de bootstrap de la interfaz gráfica de la webapp, situada en la parte superior derecha. Tal como se ha planteado en el diseño, las comunicaciones entre la webapp y los nodos se realizan a través de sockets, por lo que se apreciarán los diferentes cambios, en caso de haberlos, en tiempo real.

6.1.1. Inicialización de la webapp

En primer lugar, como se muestra en la figura 6.1 se lanza el servidor web para arrancar la webapp y que esta empiece a aceptar peticiones. Una vez ejecutado el comando, el navegador muestra la página accediendo a la dirección IP mostrada por consola.

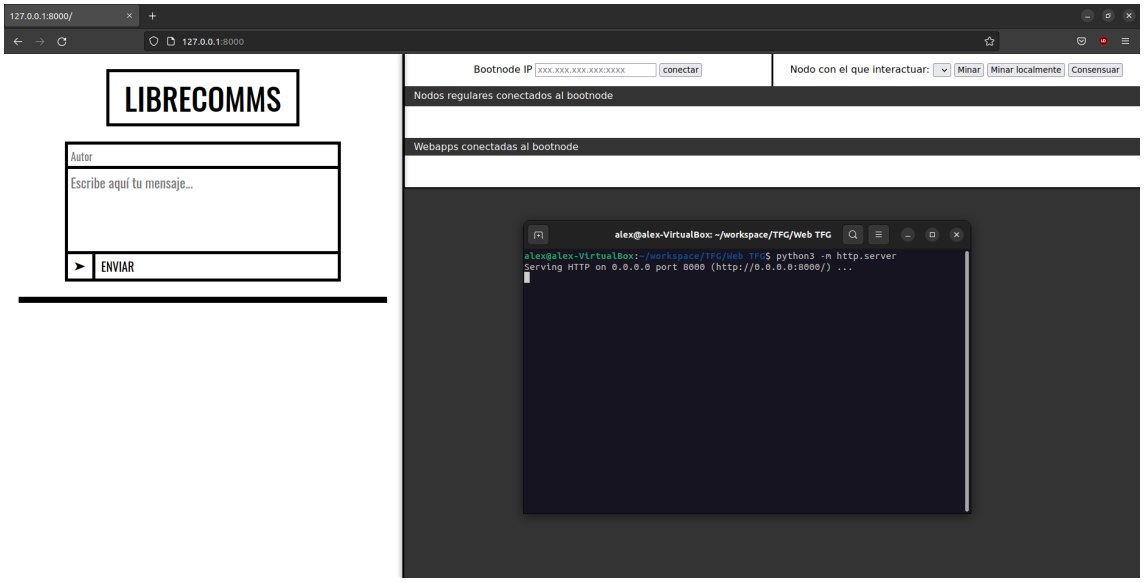

Figura 6.1: Inicialización del servidor web

6.1.2. Inicialización del nodo de bootstrap

Seguidamente, puede observarse la correcta inicialización del nodo de bootstrap, al ejecutarlo y este empezar a mostrar de forma periódica una lista con las direcciones IP de los nodos regulares activos conectados.

Una vez completada la inicialización del nodo de bootstrap puede procederse a introducir la dirección IP del mismo en el campo "Bootnode IP" en la parte superior del centro de la webapp. Al pulsar en conectar, se muestra en el apartado "Webapps conectadas a bootnode" la dirección IP de la propia webapp, lo que indica que el socket que conecta la webapp con el nodo de bootstrap funciona de manera adecuada. El resultado de este proceso se muestra en la figura 6.2

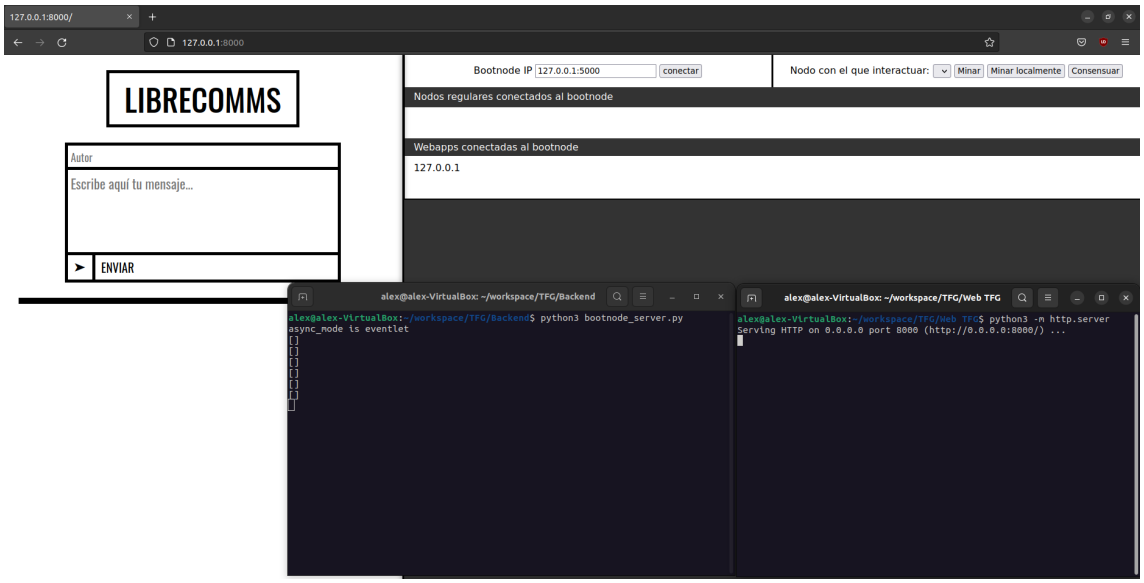

Figura 6.2: Inicialización del nodo de bootstrap

6.1.3. Inicialización de un nodo regular

Por último, se comprobará la conexión del nodo regular tanto con la webapp como con el nodo de bootstrap. De manera similiar a este último, el lanzamiento del programa seguido del mensaje "Connected!" índica que el nodo regular se ha ejecutado sin errores.

En cuanto a la conexión, tal y como se aprecia en la figura 6.3, el nodo regular se ha conectado exitosamente tanto al nodo de bootstrap (en primer lugar) como a la webapp.

Se observa la dirección IP del nodo regular, tanto en la información mostrada por la consola ejecutando el nodo de bootstrap que indica los nodos regulares actualmente conectados, como en la webapp que muestra esta misma información en la sección "Nodos regulares conectados al bootnod", que se nutre de esta misma lista proporcionada por el propio nodo de bootstrap.

Asimismo el socket entre la webapp y el nodo regular se ha establecido con éxito, como puede deducirse de la aparición de una columna que representa la Blockchain interna de este nodo, marcada con su propia dirección IP, conteniendo esta únicamente el nodo origen al acabarse de conectar y no existir otros nodos regulares registrados en el nodo de bootstrap con los que poder sincornizarse y ejecutar el algoritmo de consenso.

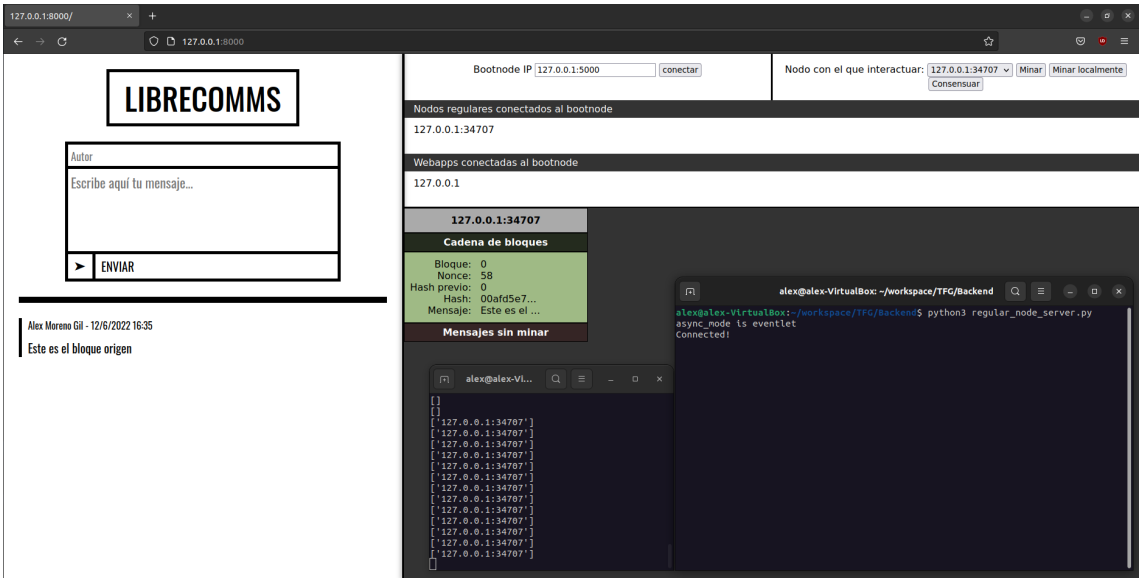

Figura 6.3: Inicialización de un nodo regular

6.2 Funcionamiento de la webapp

A continuación se mostrarán las acciones necesarias para llevar a cabo una comprobación general de todos los comportamientos implementados.

6.2.1. Envio de un mensaje

Una vez conectada la webapp al nodo de bootstrap y al menos a un nodo regular, pueden comenzar a ejecutarse operaciones que modificarán el estado de las cadenas de bloques de estos últimos. Para realizar la operación principal de enviar un mensaje de la webapp a un nodo para que este pueda minarlo, en primer lugar habrán de introducirse un nombre de usuario y un mensaje como tal en el cuadro de texto correspondiente en la parte izquierda de la webapp, tal y como se muestra en la figura 6.4

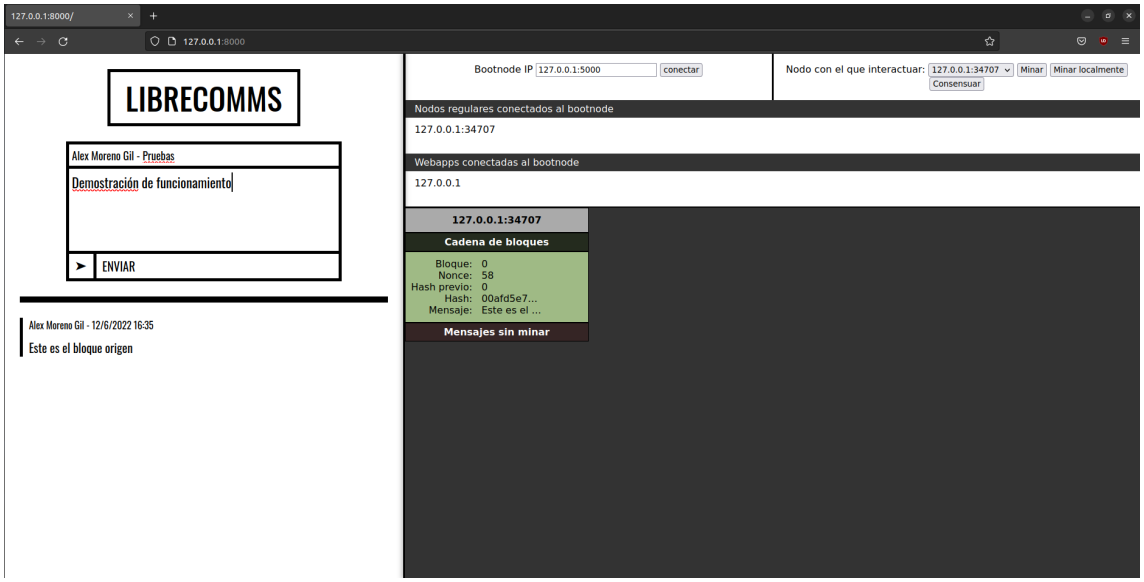

Figura 6.4: Datos de entrada en un nuevo mensaje

Seguidamente, puede apreciarse en la figura 6.5 como en el nodo correspondiente al fijado para interacción en la esquina superior derecha, se ha creado un nuevo elemento en la columna correspondiente bajo el titulo ”Mensajes sin minar”, con un texto que se corresponde al previamente introducido.

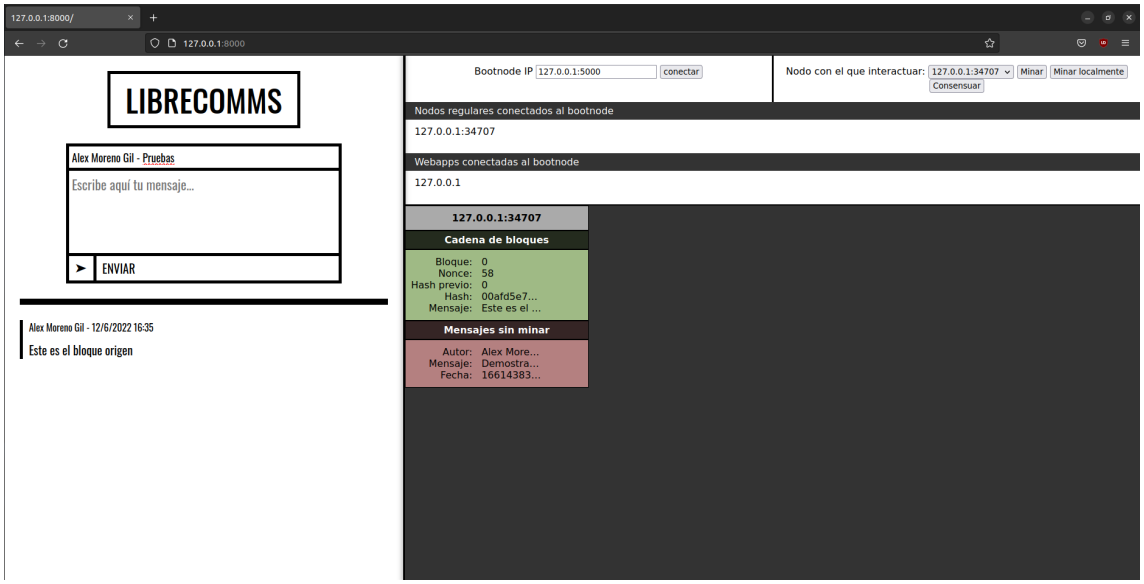

Figura 6.5: Mensaje listo para ser agregado a un bloque y minado

6.2.2. Minado de un bloque

Para terminar de comprobar el proceso de agregar un nuevo bloque a la Blockchain del nodo seleccionado, la figura 6.6 muestra el resultado de presionar el botón ”Minar” situado en la parte superior derecha de la pantalla. Puede verse como en la columna correspondiente al nodo marcado para interactuar su mensaje más antiguo se ha agregado a un bloque y ha pasado a formar parte de la sección ”Cadena de bloques”. Por último, la webapp muestra correctamente en la parte izquierda un nuevo mensaje en la sección que hace las veces de foro.

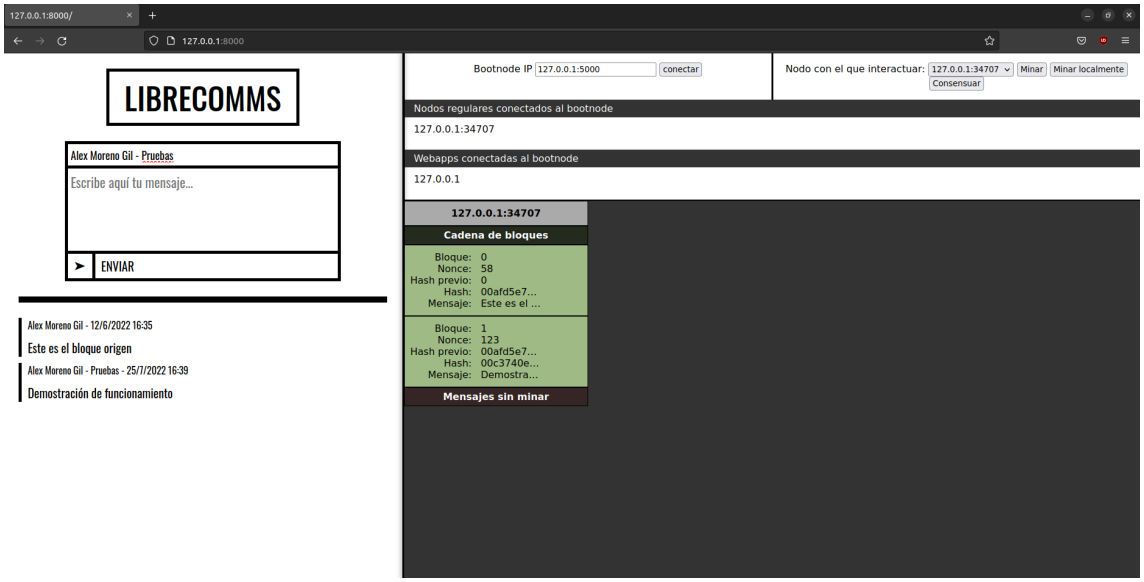

Figura 6.6: Bloque minado con éxito

6.2.3. Agregar más nodos regulares

Una vez comprobado el correcto funcionamiento con un solo nodo regular, se pondrán a prueba los comportamientos de consenso y coordinación que realmente definen una red Blockchain.

Para agregar un nuevo nodo regular, se ha ejecutado de nuevo su programa correspondiente tal y como se referencia en la figura 6.3. La consola, de nuevo mostrando un correcto comportamiento, ofrece la misma información que la figura mencionada. Es por esto por lo que se ha obviado en la captura 6.7, que permite observar que el nodo de bootstrap ha aceptado la conexión del nuevo nodo regular y la muestra a través de la webapp en la sección "Nodos regulares conectados al bootnode". Por su parte, puede afirmarse que la webapp ha establecido con éxito el socket con el nuevo nodo regular, pues una nueva columna que lo representa ha aparecido en la parte derecha de la pantalla.

Se aprecia también el comportamiento esperado en la conexión de un nodo regular a una red con una Blockchain activa, pues en lugar de aparecer únicamente con el bloque origen como el primer nodo regular en la figura 6.3, el nuevo nodo ha ejecutado el algoritmo de consenso con la lista de iguales que le ha proporcionado el nodo de bootstrap, y al reconocer una cadena de bloques más larga que la propia por parte del primer nodo regular, la ha copiado.

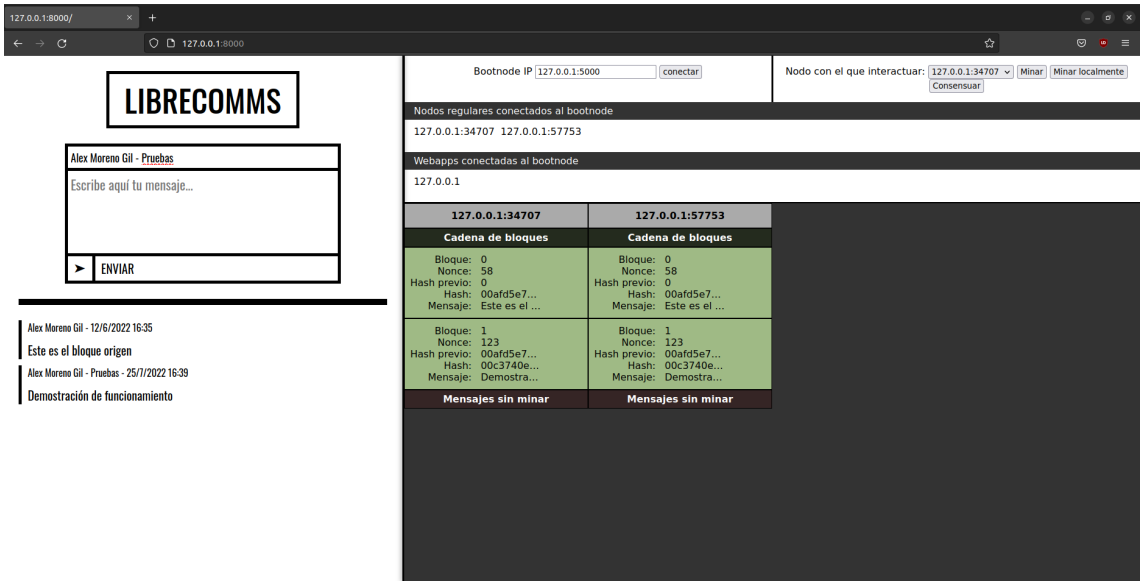

Figura 6.7: Agregar nodos regulares adicionales

6.2.4. Cambio de nodo regular objetivo de interacción

Continuando con la comprobación de conexiones, se procederá a cambiar el nodo con el que interactuará la webapp. Para ello, en primer lugar se extenderá el desplegable en la esquina superior derecha, marcado como "Nodo con el que interactuar". La figura 6.8 muestra que, en efecto, se muestra tanto el nodo regular iniciado en origen como el más reciente, sobre el que habrá que presionar para cambiar el objetivo de los diferentes comandos de la webapp.

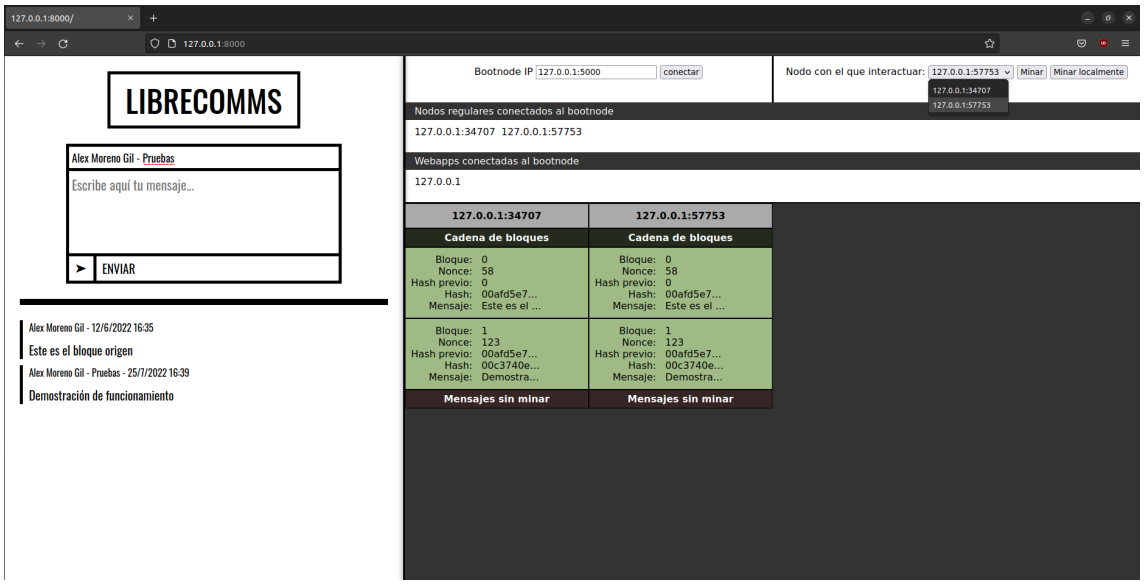

Figura 6.8: Cambio de nodo regular objetivo de interacción

Una vez seleccionado el nuevo objetivo, se añadirá un mensaje de manera análoga a como se realizó en la figuras 6.4 y 6.5. El resultado reflejado en la figura 6.9, coincide con lo esperado y el mensaje aparece como pendiente de minar en el nuevo nodo regular seleccionado como objetivo.

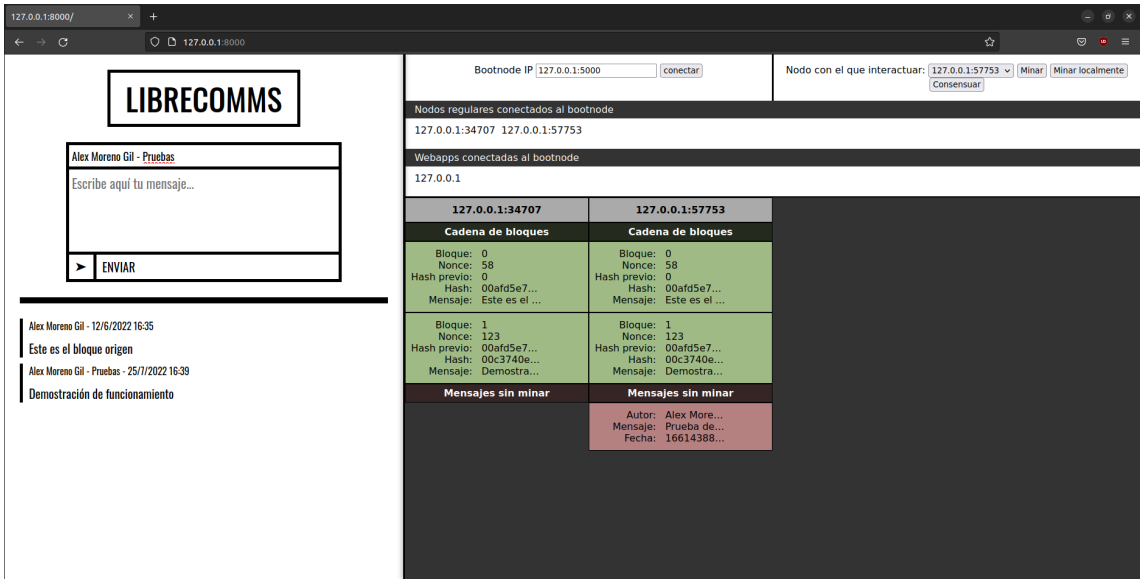

Figura 6.9: Mensaje no minado en diferente nodo regular objetivo

6.2.5. Forks y cadenas desiguales

Desde este punto, para evitar la repetición de casos ya probados y dar pie a otros nuevos, se presionará el botón "Minar localmente". Una vez más, los resultados de esta acción tal y como se muestran en la figura 6.10, coinciden con la funcionalidad esperada.

Como puede observarse en la columna del nodo regular objetivo, el minado local únicamente ha afectado a su propia cadena de bloques, y no ha comunicado la información a su igual. Por otra parte no solo se comprueba así el correcto funcionamiento del flujo de posteo y minado local, sino también la capacidad de la webapp para reconocer un nueva cadena más larga entre los nodos regulares a los que se encuentra conectada. Puede apreciarse este detalle en la réplica de la aplicación de foro en la parte izquierda de la pantalla, pués ahora muestra 3 mensajes, tantos como bloques en el último nodo regular agregado.

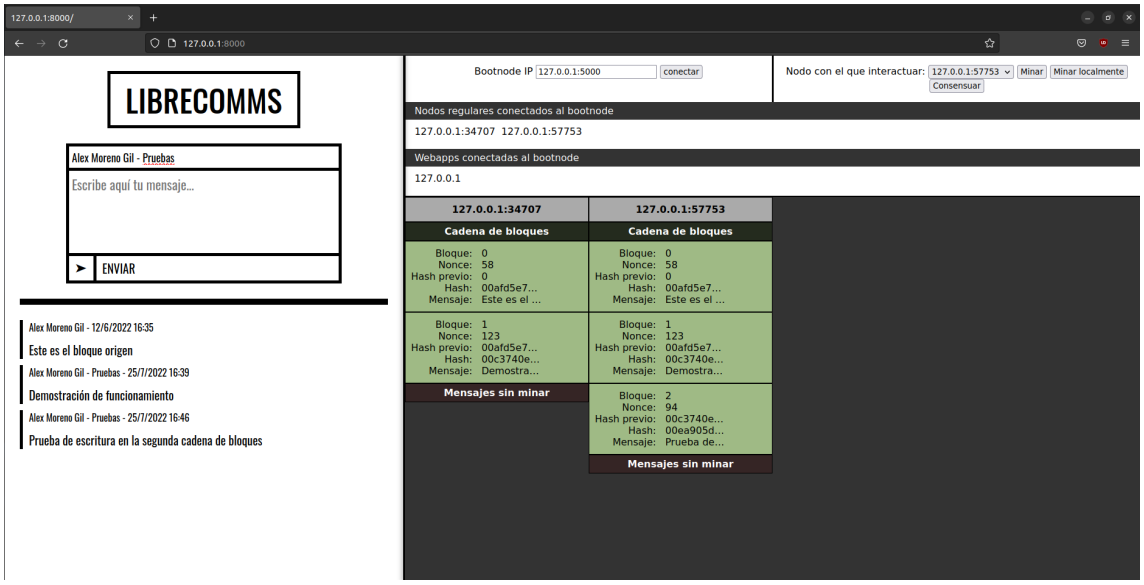

Figura 6.10: Minado local y nueva cadena más larga

Partiendo del estado anterior va a buscarse una situación de conflicto entre las dos cadenas de bloques para comprobar los comportamientos correspondientes. Con tal fin, se pretende añadir un mensaje y minarlo en el nodo regular inicialmente añadido, pues este pasaría a ser su tercer bloque, pero ya existe una cadena de tres bloques en su igual.

Para ello, se repetirán los pasos comentados en la figura 6.8 para cambiar el nodo objetivo de nuevo al inicial, y se creará y mandará un mensaje de manera análoga a la mostrada en las figuras 6.4 y 6.5 respectivamente. Todo ello lleva al estado consistente y esperable mostrado en la figura 6.11.

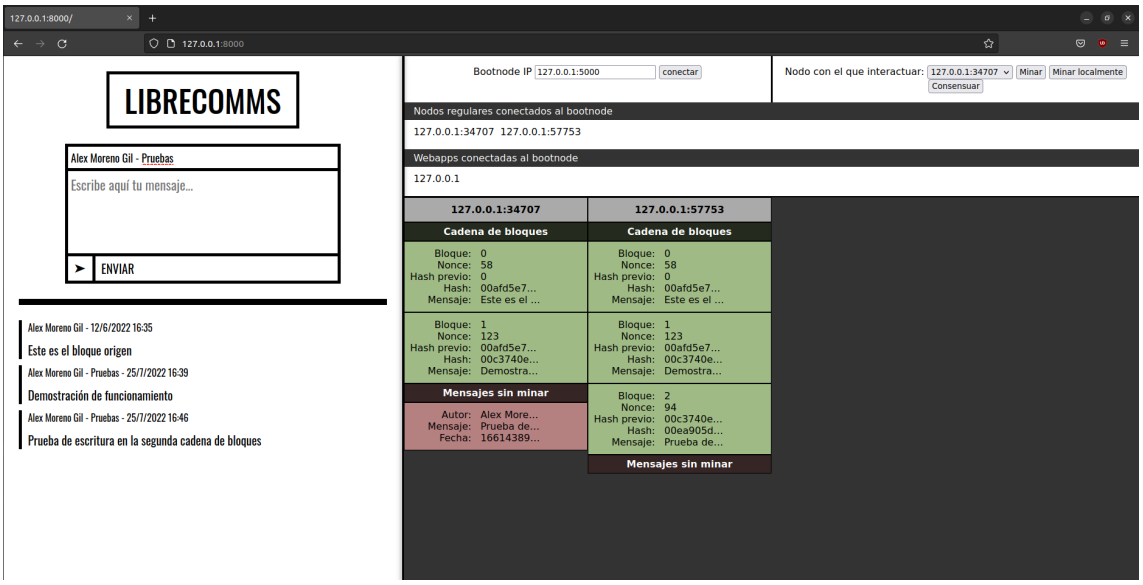

Figura 6.11: Preparación de la primera situación de conflicto

Para finalizar la primera situación de conflicto, sencillamente se presionará el botón ”Minar” con el nodo sobre el que se acaba de crear un mensaje no minado como objetivo. Es importante presionar ”Minar” en lugar de ”Minar localmente” para que el nodo trate de compartir el nodo que acaba de minar y consensuar la cadena más larga con sus iguales.

Se presentan los resultados en la figura 6.12, en la que se puede apreciar que el conflicto se ha resuelto tal y como se había planteado: El primer nodo minó el bloque, pero al ejecutar el algoritmo de consenso reconoció no tener la cadena más larga y lo descartó. Seguidamente, copió la cadena más larga y volvió a minar el bloque, que esta vez si que pudo transmitir a su igual de forma exitosa, al compartir la misma cadena base. Por su parte la webapp muestra los nuevos bloques tanto como nuevos mensajes en la réplica del foro como en las columnas correspondientes a los nodos regulares.

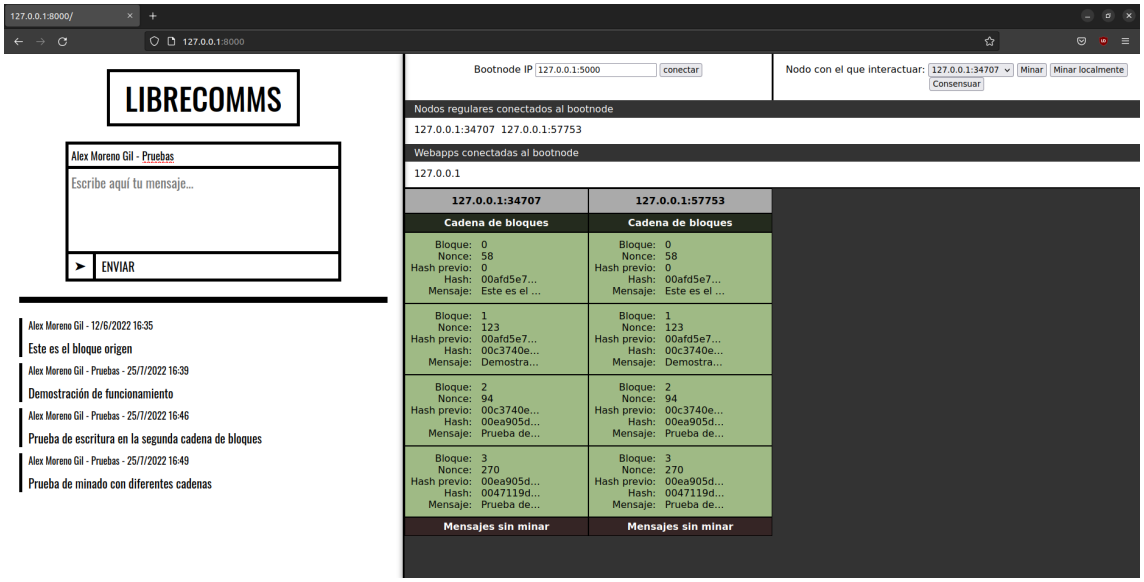

Figura 6.12: Resolución de la primera situación de conflicto

Para terminar con las pruebas de la sección de conexiones se generará un último caso de conflicto que cubre los comportamientos definidos restantes.

En primer lugar, se partirá de dos nodos regulares con cadenas nuevas. Para ello se frenará la ejecución de los dos nodos con los que se ha trabajado hasta ahora y se levantarán dos nuevos tal y como se mostraba en la figura 6.3. Una vez levantados se seguirá en ambos el siguiente flujo: seleccionar nodo como objetivo tal como en la figura 6.8, se creará un nuevo mensaje (con el texto A para el primer nodo y B para el segundo) como en 6.4 y 6.5 y se minará de manera local análogamente a como se mostró en la figura 6.10.

Este conjunto de acciones resultan en la captura de pantalla mostrada en la figura 6.13. Puede verse en esta que los resultados son congruentes con las acciones que se han tomado y las pruebas correspondientes a las que se ha hecho referencia.

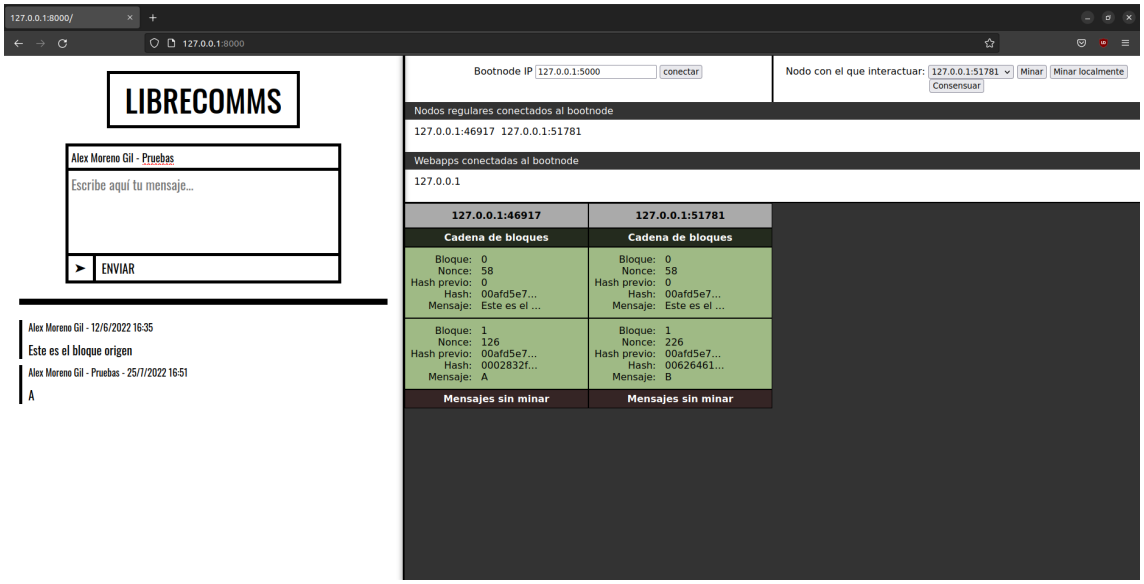

Figura 6.13: Preparación de la segunda situación de conflicto

Finalmente, se creará y minará (de manera estándar, no localmente) un último mensaje y bloque respectivamente, como se ha indicado anteriormente, en el segundo nodo regular. La figura 6.14 muestra concordancia con lo esperado, el bloque al haber sido minado con éxito ha sido notificado por el nodo a su igual, que al no encontrarlo adecuado ha ejecutado el algoritmo de consenso. Al aprender que no tiene la cadena más larga, ha desechado la propia y ha copiado la otra. Tanto la réplica del foro como las columnas de representación de la Blockchain reflejan esto.

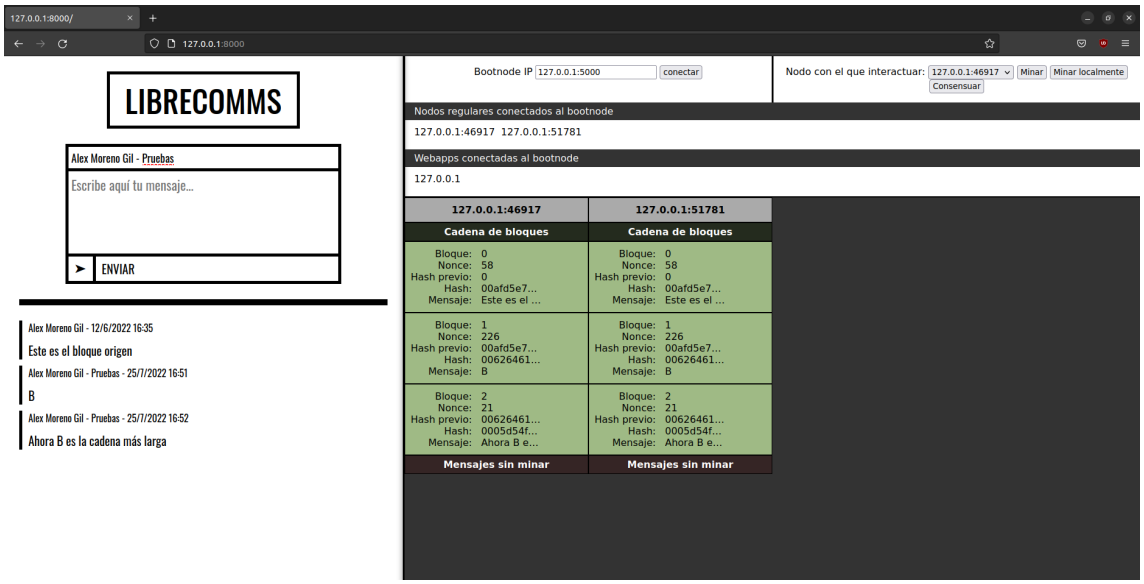

Figura 6.14: Resolución de la segunda situación de conflicto

Como último detalle antes de terminar la sección de conexiones, cabe destacar que no se ha comprobado el funcionamiento del botón "Consensuar". Esto es por estar ligado a comportamientos que se han ejecutado de manera satisfactoria en las pruebas anteriores, por lo que se considera redundante.

6.3 Adaptación de la interfaz a valores elevados de datos a mostrar

Para finalizar el capítulo de pruebas de comprobará que los métodos utilizados para gestionar los valores desboardantes en la interfaz gráfica funcionan debidamente y no se está perdiendo utilidad en favor de estética.

6.3.1. Valores en los bloques

Puede observarse en la figura 6.15 como al posicionar el cursor del ratón sobre cualquiera de los valores de un bloque especificados como desbordantes mediante "...", se muestra el valor al completo de la propiedad.

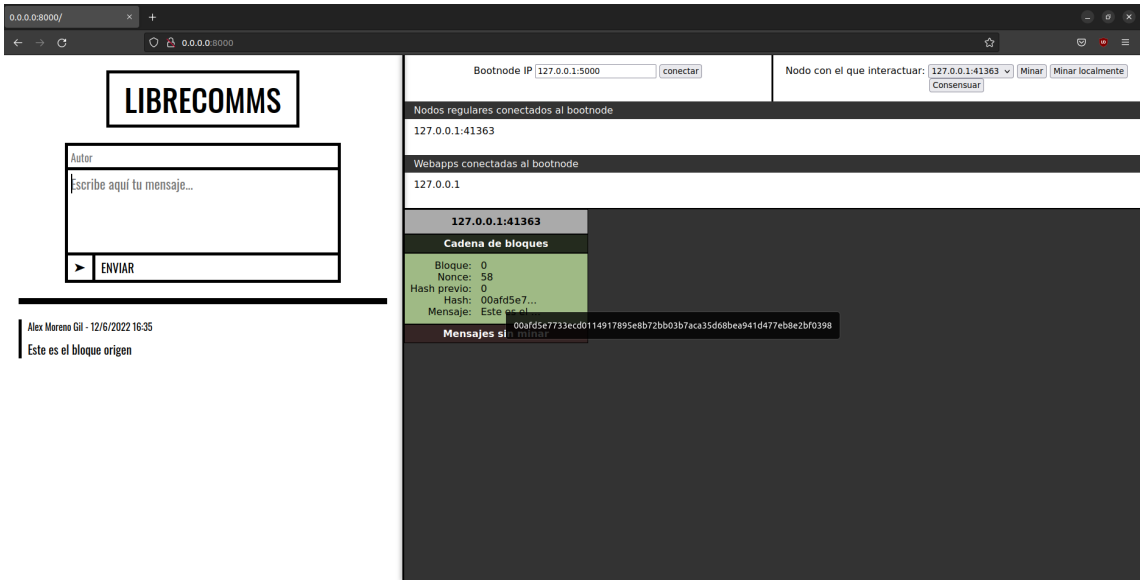

Figura 6.15: Valor de un bloque al completo

6.3.2. Columnas largas

Tal y como se ha contemplado en el desarrollo, la figura 6.16 muestra que agregar un alto número de mensajes sin minar o bloques no afecta a la usabilidad y un simple scroll permite verlos todos tanto en las columnas de representación de las cadenas de bloques como en la réplica del foro en la parte izquierda.

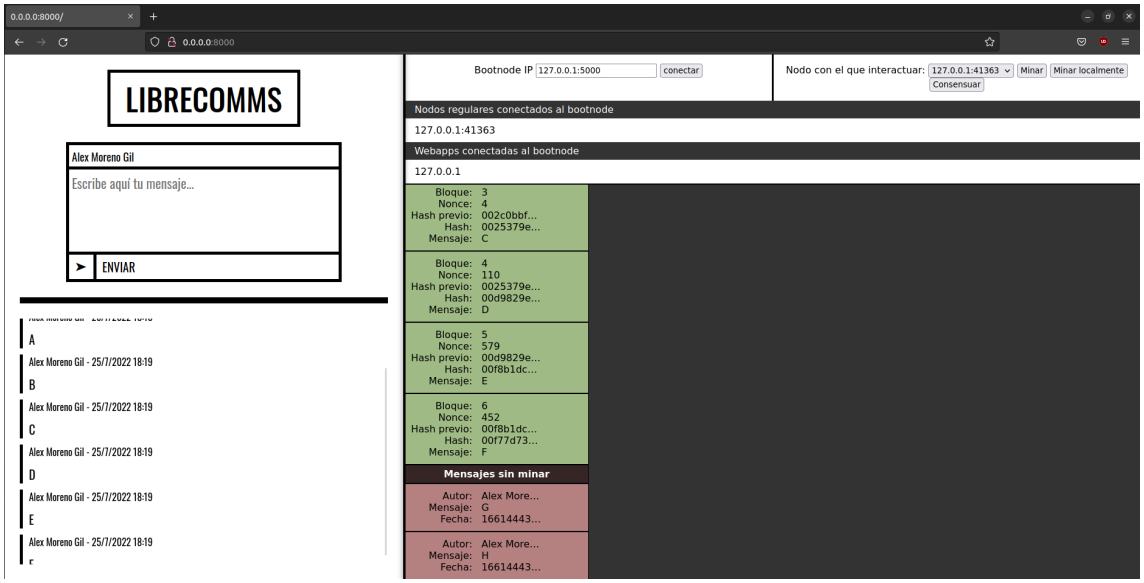

Figura 6.16: Scroll en vista de Blockchains y réplica de foro

CAPÍTULO 7
Conclusiones

Como se puede observar a lo largo del diseño y el desarrollo del sistema, los objetivos nunca han dejado de ser el hilo conductor del proyecto. Han sido estos los que han marcado las especificaciones y requisitos necesarios para el desarrollo de un sistema capaz, que cumple las expectativas de la visión generalista de una solución al problema que se ha dado como motivación principal para la realización del proyecto.

Se revisan a continuación con un mayor grado de detalle los objetivos planteados en un primer momento, y óomo y en qué grado la solución implementada los ha cubierto:

- **Funcionamiento que respete las especificaciones presentadas:** En el apartado de pruebas queda patente el correcto funcionamiento de la aplicación, mostrando el comportamiento esperado en cada uno de los casos y representando de una manera fiel a una hipotética aplicación equivalente encontrada en el mundo real, empleada en producción por una organización dada.
- **Interfaz clara:** Una paleta de colores que sigue las convenciones comunes en la tecnología que el usuario conocerá en la mayoría de los casos, dos secciones bien diferenciadas para no causar confusión en sus propósitos y un panel de control claro y organizado con todos los elementos necesarios sin que exista redundancia. Se considera que se han cumplido las pretensiones visuales respecto a una herramienta educativa como se planteó en un primer momento.
- **Arquitectura simple:** Puede observarse en la etapa de diseño como este ha sido un punto fundamental, y como se ha respetado, consiguiéndose así un sistema relativamente complejo pero formado por simples componentes con clases y relaciones bien definidas y organizadas y sencillas de seguir.
- **Código abierto:** Se ha publicado el código desarrollado en [17] con una licencia poco restrictiva de tipo MIT, lo que cumple el objetivo propuesto y permite el uso y la modificación del código a todo aquel interesado sin ningún tipo de problema.

Más allá de los objetivos propuestos inicialmente y en un nivel más personal, el proyecto ha sido capaz de reflejar y hacer ver la escala temporal y de esfuerzo necesaria para llevar a cabo un proyecto al completo de una manera que otras experiencias en un entorno más clásicamente académico a lo largo del grado no han sido capaces de imitar. Estas nuevas referencias y necesidades son pues un éxito en sí mismo aplicables de manera inmediata en futuros proyectos tanto personales como profesionales.

CAPÍTULO 8
Relación con estudios cursados

El proyecto propuesto se encuentra ampliamente ligado a los conocimientos adquiridos durante la carrera a través de las diferentes asignaturas. No como piezas independientes, sino como conocimientos compatibles que se han combinado permitiendo formar una aplicación mediante el uso conjunto de sus diferentes enseñanzas.

8.1 Asignaturas

Como ejercicio de introspección y de una manera organizada podría clasificarse de la siguiente manera la aportación de cada una de las asignaturas más involucradas:

- **Arquitectura global del proyecto y específica del "subsistema Blockchain":** Esta área se profundizó lo suficiente en **Sistemas y servicios en red** como para asentar las bases en el conocimiento de las aplicaciones distribuidas, como la muy parecida red torrent, que han servido como base de conocimiento sobre la que empezar a construir.
- **Diseño e implementación de la Webapp:** Un sistema compuesto por varios factores, presentando **Desarrollo web** los conocimientos necesarios para la configuración de un servidor web que permitiese lanzar una página web con todos los archivos relacionados. Por su parte **Interfaces persona computador** y **Desarrollo centrado en el usuario** han facilitado la consecución del objetivo principal de presentar una interfaz limpia, intuitiva y de alta usabilidad mediante la enseñanza de principios básicos de diseño y codificación de elementos visuales.
- **Comunicación entre sistemas:** Tanto las bases de JavaScript asentadas por **Tecnología de sistemas de información en la red** como los principios de redes de ordenadores y funcionamiento de sockets presentados por **Redes de computadores** han resultado de gran ayuda para la configuración de una correcta conexión entre sistemas. Una vez lograda esta conexión, han sido los conocimientos impartidos por **Integración de aplicaciones** los que han facilitado en gran medida el correcto envio y recepción de los datos en un formato correcto.
- **Concurrencia:** Ciertas funcionalidades puntuales del proyecto han requerido de ejecución concurrente para poder implementar el comportamiento planificado, temas altamente relevantes tratados tanto en **"Concurrencia y sistemas distribuidos"** como en **"Computación paralela"**.
- **Programación:** A un nivel mucho más general, las bases de algorítmica y elección de las adecuadas estructuras de datos enseñadas en **Introducción a la informática**

y la programación, **Programación** y **Estructuras de datos y algoritmos** han sido de gran utilidad.

- **Organización del proyecto y redacción de la memoria:** Como cabría esperar **Gestión de proyectos**, la asignatura planificada especialmente para ello ha sido de gran ayuda en la planificación y realización de la idea que dio comienzo al proyecto.

8.2 Competencias transversales

Cabe remarcar también las diferentes competencias transversales adquiridas o mejoradas a lo largo del grado que más impacto han tenido en la realización de este trabajo:

- **CT1 - Comprensión e integración:** La interiorización y utilización de nuevos elementos y tecnologías conforme se van presentando, como suele darse en las sesiones de laboratorio, permiten adaptarse a las diferentes necesidades del proyecto durante su desarrollo y planificación.
- **CT3 - Análisis y resolución de problemas:** La adaptabilidad y comprensión alrededor de los diferentes problemas que pueden surgir en un desarrollo o en cualquier otra área es posiblemente una de las habilidades más valiosas, no solo (aunque especialmente) en el desarrollo de un proyecto, sino en muchos aspectos de la vida.
- **CT5 - Diseño y proyecto:** Los diferentes pequeños proyectos desarrollados durante la carrera agilizan y preparan a la hora de afrontar un equivalente mayor como este proyecto.
- **CT11 - Aprendizaje permanente:** Educar al cerebro para aprender ciertos conocimientos de alto interés y repasarlos con frecuencia facilita el no atascarse en problemas o desafíos de fácil resolución mediante la aplicación de conocimientos base como los impartidos en el grado.

CAPÍTULO 9
Trabajos futuros

El sistema planteado cubre las necesidades que motivaron el desarrollo del proyecto, y funciona tal y como se esbozó durante el diseño. Aún así, es un proyecto que se presta a un desarrollo futuro por diferentes vías con tal de mejorar su potencial, la simplicidad de su ejecución y capacidad de experimentación para mejorar la experiencia didáctica.

A continuación se exponen algunas de las líneas de desarrollo que podrían seguirse a partir del estado actual del proyecto:

- **Ampliar la variedad de algoritmos:** El uso del algoritmo de prueba de trabajo, a pesar de ser el más extendido, es responsable de un gran gasto energético[18]. Es por esto que se están buscando alternativas y algunas están encontrado una gran adopción como el uso del algoritmo de prueba de participación[19]. La implementación de diferentes algoritmos a elegir sobre el proyecto pueden ayudar a ejemplificar el funcionamiento y diferencias para cubrir un caso que el proyecto actual no contempla.
- **Mayor fidelización para con la realidad:** Se han tomado ciertas decisiones de abstracción durante la fase de diseño que, a la par que simplifican el entendimiento de la Blockchain, lo alejan de un caso de uso real. La implementación de varios nodos de bootstrap, la capacidad de ampliar la lista local de nodos de bootstrap de forma persistente o guardar otros nodos regulares para hacer las veces de nodos de bootstrap o el hecho de que los bloques puedan aceptar más de un mensaje por bloque minado son desarrollos que acercarían la solución propuesta a la realidad.

Bibliografía

[1] Dabbagh M., Sookhak M. and Safa N. S. The Evolution of Blockchain: A Bibliometric Study. *IEEE Access PP(99):1-1*, 2019.

[2] Khanna, A.; Sah, A.; Bolshev, V.; Jasinski, M.; Vinogradov, A.; Leonowicz, Z.; Jasiński, M. Blockchain: Future of e-Governance in Smart Cities. *Sustainability*, 2021.

[3] Mandrita Banerjee, Junghee Lee, Kim-Kwang Raymond Choo. A blockchain future for internet of things security: a position paper. *Digital Communications and Networks*, 2018.

[4] Bhaskar, P., Tiwari, C.K. and Joshi, A. Blockchain in education management: present and future applications. *Interactive Technology and Smart Education*, 2021.

[5] Danson Kimani, Kweku Adams, Rexford Attah-Boakye, Subhan Ullah, Jane Frecknall-Hughes, Ja Kim. Blockchain, business and the fourth industrial revolution: Whence, whither, wherefore and how? *Technological Forecasting and Social Change*, 2020.

[6] Moon, S. H. The Role and Opportunity of Blockchain in the Fourth Industrial Revolution. *The Journal of the Convergence on Culture Technology*, 2019.

[7] Casamayou A. Personas mayores y tecnologías digitales: desafíos de un binomio. *Psicología, Conocimiento y Sociedad 7(2)*, noviembre 2017–abril 2018.

[8] Bheemaiah K. Why Business Schools Need to Teach About the Blockchain. *http://dx.doi.org/10.2139/ssrn.2596465* , 2015.

[9] Haber, S., Stornetta, W.S. How to Time-Stamp a Digital Document. *Menezes, A.J., Vanstone, S.A. (eds) Advances in Cryptology-CRYPTO' 90*, 1991.

[10] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *https://bitcoin.org/bitcoin.pdf*, 2008

[11] Dettling, W. *Business Information Systems and Technology*. Springer International Publishing AG, 2018.

[12] Kolb. D. A. and Fry, R. Toward an applied theory of experiential learning. *C. Cooper (ed.), Theories of Group Process*, 1975.

[13] Herramienta de aprendizaje interactivo de la blockchain https://andersbrownworth.com/blockchain/.

[14] Código abierto de herramienta de aprendizaje interactivo de la blockchain https://github.com/anders94/blockchain-demo.

[15] J. Seanosky et al. Real-Time Visual Feedback: A Study in Coding Analytics. *IEEE 17th International Conference on Advanced Learning Technologies*, 2017.

[16] Aplicación blockchain básica en Python https://recursospython.com/guias-y-manuales/aplicacion-blockchain-desde-cero/.

[17] Código abierto de herramienta de aprendizaje interactivo de la blockchain https://github.com/alex-azul/TFG-Didactic-Python-App.

[18] de Vries A., Gallersdörfer U., Klaaßen, L., Stoll, Christian. Revisiting Bitcoin's carbon footprint. *Joule*, 2022.

[19] Fahad Saleh. Blockchain without Waste: Proof-of-Stake. *The Review of Financial Studies*, 2021.

APÉNDICE A
Objetivos de Desarrollo Sostenible

Los objetivos de desarrollo sostenible son un conjunto de metas propuestas el 25 de septiembre de 2015, acordadas de manera global a alcanzar para mejorar la calidad de vida de las personas en todo el mundo. Se propusieron con las metas de la consecución de un mundo más justo en el que no existan diferencias económicas tan severas, la conservación del mundo natural y el seguimiento de un camino hacia una sociedad prospera para todos. Cada uno de estos puntos se subdivide en una cantidad de objetivos concretos y alcanzables que permitirán alcanzar las metas propuestas. Los objetivos que se han considerado con relación al proyecto propuesto son los que se muestran en la figura A.1

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**		■		
ODS 2. **Hambre cero.**				■
ODS 3. **Salud y bienestar.**				■
ODS 4. **Educación de calidad.**	■			
ODS 5. **Igualdad de género.**			■	
ODS 6. **Agua limpia y saneamiento.**				■
ODS 7. **Energía asequible y no contaminante.**			■	
ODS 8. **Trabajo decente y crecimiento económico.**				■
ODS 9. **Industria, innovación e infraestructuras.**		■		
ODS 10. **Reducción de las desigualdades.**			■	
ODS 11. **Ciudades y comunidades sostenibles.**				■
ODS 12. **Producción y consumo responsables.**				■
ODS 13. **Acción por el clima.**				■
ODS 14. **Vida submarina.**				■
ODS 15. **Vida de ecosistemas terrestres.**				■
ODS 16. **Paz, justicia e instituciones sólidas.**				■
ODS 17. **Alianzas para lograr objetivos.**				■

Figura A.1: Relación con los diferentes Objetivos de Desarrollo Sostenible

La solución que se ha tratado a los largo de esta memoria está directa o indirectamente relacionada con algunos de estos puntos de la manera que a continuación se explica, lo que permite aportar para construir un mundo mejor.

- **Fin de la pobreza:** Aunque de manera indirecta, la tecnología Blockchain y su característica inmutabilidad apuntan maneras para moldear la sociedad de tal manera que se aleje el factor humano de gestiones burocráticas y económicas. Esto a su vez implica una menor facilidad de corrupción y un mayor manejo de las finanzas personales, ambos factores importantes para ayudar en la lucha contra la pobreza.
- **Educación de calidad:** Es el tema central que motiva la misma creación del proyecto. Se considera de suma importancia el acceso a herramientas de educación imparciales y con una barrera de entrada lo más baja posible, así como la formación en nuevas tecnologías que pueden llegar a adoptarse de manera masiva y afectar a toda la población.
- **Igualdad de género:** Un factor común en la mayoría de los escenarios donde se considera la implementación de la Blockchain como un avance es la automatización y eliminación del factor humano. El hecho de que ciertos procesos relacionados con lo económico, legal, político y social se cumplan de manera determinista e inquebrantable siguiente el código definido previamente aisla a todas estas gestiones de los prejuicios que podría aportar cualquier persona involucrada.
- **Energía asequible y no contaminante.** Actualmente la tecnología Blockchain, en gran parte debido a las criptomonedas, es la causante de un gasto energético masivo debido al algoritmo de prueba de trabajo utilizado en el minado. Facilitar la formación en este campo incentiva la aparición de nuevos profesionales capaces de crear alternativas más limpias y eficientes.
- **Industria, innovación e infraestructuras:** Facilitar el aprendizaje de la Blockchain ayuda a que haya mas profesionales con capacidades de implementarla en situaciones reales, innovando y creando así nueva infraestructura software.
- **Reducción de las desigualdades:** Del mismo modo que con la igualdad de genero, el tratamiento de ciertas materias sensibles por un ente imparcial evita las discriminaciones directas o indirectas que puedan agregar las manos de cada persona por la que pase una determinada gestión.

Puede observarse de estas afirmaciones que dado el enfoque didáctico del proyecto muchos de estos objetivos no se alcanzan de manera directa, sino a través de todos aquellos que lo utilicen a modo de entrada a esta nueva tecnología para poder ayudar en una multitud de diferentes campos.