Propuesta de Trabajo Profesional

Análisis de ecosistemas para la implementación de plataformas como servicio para despliegue de aplicaciones comunitarias, distribuidas y descentralizadas

Integrantes

Lucas Nahuel Sotelo Guerreño
102730
lsotelo@fi.uba.ar

Sebastián Bento Inneo Veiga
100998
sinneo@fi.uba.ar

Joaquín Matías Velazquez
105980
jvelazquez@fi.uba.ar

Joaquín Prada
105978
jprada@fi.uba.ar

Tutor
Ariel Scarpinelli
ascarpinelli@fi.uba.ar

Índice

1. Resumen

En el siguiente trabajo se analizan distintos ecosistemas y tecnologías que se pueden utilizar para el despliegue de aplicaciones web comunitarias de manera distribuida y descentralizada.

Mediante tres casos de uso que ilustran diferentes características: un sitio web informativo, un repositorio de conocimiento y un mensajero en tiempo real, se comparan ventajas y desventajas del despliegue de cada uno de ellos en IPFS, blockchain y Hyphanet/Freenet, así como también se documenta el proceso del mismo.

2. Palabras Clave

Distribuido. Sistema. Comunitario. Descentralizado. Aplicación.

3. Abstract

The following work analyzes different ecosystems and technologies that can be used to deploy community web applications in a distributed and decentralized manner.

Using three use cases that illustrate different features: an informational website, a knowledge repository and a real-time messenger, the advantages and disadvantages of deploying each of them on IPFS, blockchain and Hyphanet/Freenet are compared, as well as the process to do so is documented.

4. Keywords

Distributed. System. Community. Decentralized. Application.

5. Introducción

Hoy en día, al querer desplegar una aplicación o sitio web comunitario, lo más común es hacerlo a través de un servicio de alojamiento (AWS, Azure, Google Cloud, entre otros) por la comodidad y facilidad que estas ofrecen, alquilando sus servidores para guardar y procesar datos.

Esto puede llegar a traer problemas para este tipo de aplicaciones. Uno de estos problemas puede ser monetario, ya que muchas veces estas aplicaciones dependen de donaciones o voluntarios para sustentarse, como es el caso de Wikipedia. Como también puede suceder que se encuentre en una zona de censura, lo cual facilita su bloqueo al ser servicios centralizados; entre otros problemas más.

Sin embargo, existen otros ecosistemas alternativos que se asemejan mucho más a la filosofía de estas aplicaciones, y que ayudan a combatir estos problemas. En donde las aplicaciones pueden estar alojadas por sus propios usuarios, donando su computo o espacio, y así logrando una descentralización.

En el siguiente documento presentamos un análisis sobre la infraestructura existente, donde es posible la implementación de plataformas para el despliegue de este tipo de aplicaciones, recabando las bondades y desventajas que cada una tiene.

Para esto se crearon diferentes casos de uso que representan posibles aplicaciones sobre esta metodología alternativa analizando su viabilidad. Entre ellos, se encuentran un sitio web estático, una enciclopedia colaborativa y una aplicación de comunicación en tiempo real.

6. Estado del Arte

En esta sección describiremos en qué se diferencian las aplicaciones descentralizadas de aquellas centralizadas, cuáles son las ventajas (y desventajas) del modelo de aplicación distribuido, y qué tecnologías existen actualmente para asistir en la creación de dichas aplicaciones.

6.1. Introducción a arquitecturas de red

Comunicar distintas computadoras es un trabajo que requiere coordinación por parte de todas las partes, protocolos para estandarizar la información que se transmite, e infraestructura para poder envíar cada bit de origen a destino. Entonces, se debe diseñar una red coordinada para poder ofrecer los distintos servicios a través de Internet. Para ello, existen dos arquitecturas principales.

Cliente-Servidor

Presente en la gran mayoría las aplicaciones de Internet, el modelo *Cliente-Servidor* consiste en mantener un nodo central (servidor), quién se encarga de manejar la interacción entre los demás nodos (clientes), y entre clientes y el mismo servidor. Este modelo se clasifica como **centralizado**, debido a que la subred de sistemas depende del nodo servidor, y los clientes no tienen manera de comunicarse sin él ante una eventual caída del servidor.

Entre los servicios de Internet más utilizados que utilizan esta arquitectura se encuentra la World Wide Web, el servicio de e-mail (SMTP), el servicio de DNS, entre otros.

Peer-to-Peer

El modelo **peer-to-peer(P2P)** consiste en una red **descentralizada** que tiene distintos nodos (pares) capaces de comunicarse sin necesidad de un nodo central, por lo que se puede considerar que cada nodo cumple la función tanto de servidor como de cliente a la vez.

BitTorrent El servicio más utilizado que implementa este modelo es la red de BitTorrent, que implementa el protocolo del mismo nombre para compartir archivos entre pares. Esta red logra que el mismo nodo que descarga un contenido de la red sea a la vez el servidor para otro nodo que quiera acceder a ese contenido.

6.2. Diferencias y ventajas de cada arquitectura

Ambos modelos tienen ventajas y desventajas, y por lo tanto distintos casos de uso. El modelo cliente-servidor actualmente es la arquitectura mas utilizada,

Resiliencia Cuando un servidor se encuentra fuera de servicio, toda la red que depende de él no funcionará en tanto no se restaure el servidor. Para contrarrestar esta vulnerabilidad del modelo, se desarrollaron métodos a lo largo de los años. Una manera de evitar que la red se vuelva inoperativa es la de alojar diferentes instancias del servidor en diferentes zonas geográficas. Además, se pueden implementar medidas para evitar la caída del servidor, como los técnicas de balanceo de cargas y fuentes de energía alternativas para evitar eventuales cortes de electricidad.

No obstante, una red peer-to-peer puede ser incluso más robusta. Si hay suficientes pares en la misma y están lo suficientemente dispersos geográficamente, la desconexión de uno de ellos no desactiva toda la red. Esto permite que el modelo peer-to-peer pueda ser resistente a cortes de energía masivos y desastres naturales, lo que lo hace un modelo ideal para servicios prioritarios.

Cabe destacar que en una red de una cantidad limitada de pares, en donde no hay redundancia del contenido que se distribuye, es posible que al desconectar uno de los pares parte del contenido

se vuelva no disponible. Por lo tanto, si bien la red seguirá activa, no tendrá toda la funcionalidad que si puede ofrecer un servidor mientras siga en línea.

Escalabilidad La escalabilidad de una red peer-to-peer aumenta con la cantidad de nodos disponibles, dado que hay más recursos y, en una red bien diseñada, la carga se distribuye equitativamente. En un modelo cliente-servidor, garantizar escalabilidad se puede tornar costoso. Un servidor con mayor capacidad para comunicaciones entrantes y volumen de información tiene hardware de un costo mayor. En casos de aplicaciones de uso masivo puede ser necesario multiplicar la cantidad de nodos servidores para satisfacer la demanda de clientes.

Control del contenido Un servidor, al ser la pieza central de la red a la cuál pertenece, debe soportar múltiples conexiones en simultáneo. Para aplicaciones de alto tráfico, esto requiere una infraestructura que los usuarios suelen no poseer. Una solución es tercerizar el alojamiento de la aplicación servidor en plataformas de Cloud Hosting como pueden ser AWS, Azure, Google Cloud, entre otras. Estos servicios mantienen los servidores de numerosas aplicaciones de Internet, y por lo tanto, tienen la capacidad de modificar, censurar, o remover cualquiera de ellas si así lo desean.

Una red P2P no sufre de estos problemas, ya que por naturaleza los usuarios son quienes la alojan. Por lo tanto, remover contenido de ella resulta mucho mas complejo. Esto evita la censura en zonas donde el acceso a Internet es controlado y/o limitado, pero también puede incentivar a la distribución de contenido ilegal.

Seguridad La seguridad en las aplicaciones cliente-servidor se ha investigado por mucho más tiempo debido a la popularidad de este modelo. Además, al ser centralizado, el propietario del servidor puede bloquear conexiones y eliminar contenido malicioso de su plataforma de forma transparente para los usuarios.

El modelo descentralizado, en cambio, no cuenta con el desarrollo en términos de seguridad. En este caso, el cliente es el responsable de conectarse a redes de confianza, o de hacerlo mediante VPNs (Virtual Private Networks) para mantener el anonimato mientras se integre una red P2P. Sin embargo, cualquiera sea el modelo utilizado por una aplicación, la mayor parte de la seguridad dependerá de que tan segura sea la aplicación.

Persistencia Como varias otras propiedades del modelo cliente-servidor, depende de la integridad del servidor. Si el almacenamiento físico de este se ve afectado, los datos pueden perderse definitivamente. Por esta razón, es común tener un respaldo de los datos de la aplicación en otro disco u otro nodo para evitar la pérdida total de datos.

En una red descentralizada, la persistencia depende de la aplicación utilizada. En el caso de BitTorrent, cada nodo que se conecte y descargue un archivo, podrá compartir ese archivo con otros nodos, y por lo tanto ese archivo contará con una redundancia adicional, la cuál crece a medida que más personas descargan ese archivo. A pesar de esto, en casos donde el archivo es poco compartido, puede volverse inaccesible si los nodos que lo contienen se desconectan de la red.

Latencia Dada una conexión a Internet promedio, las velocidades manejadas por las aplicaciones cliente-servidor suelen ser aceptables. Sin embargo, en zonas en donde la conexión es escasa, o en casos en donde el servidor está lejos del cliente, la velocidad de transferencia de la aplicación puede verse afectada. Además, no es infrecuente encontrar cortes en videollamadas, juegos, y demás aplicaciones de tiempo real que siguen esta arquitectura. Como en la mayoría de defectos del modelo cliente-servidor, se puede solucionar agregando múltiples instancias del servidor. Por ejemplo, es común almacenar películas, videos y demás contenido de aplicaciones de streaming en distintos servidores de CDN (Content Delivery Network). Estas redes minimizan la distancia entre el usuario y el servidor, agilizando así la transferencia del contenido.

A pesar de los avances en la optimización del modelo cliente-servidor, las redes descentralizadas, cuando son eficientes y están bien pobladas, suelen ofrecer incluso mejores resultados. Esto se debe

a que la fuente de un contenido puede estar presente en múltiples nodos, lo que aumenta la probabilidad de que un nodo cercano tenga el contenido solicitado. La velocidad de transferencia que puede proporcionar un vecino con el contenido que requerimos generalmente superará la ofrecida por un servidor.

Costos Los costos de alojar una aplicación peer-to-peer suele ser nulo, ya que los mismos usuarios de ella son los encargados de proporcionar la infraestructura de la red.

Al contrario, alojar la aplicación en un servidor conlleva tener un servidor disponible, o bien contratar un servicio de web hosting, cuya tarifa suele aumentar a medida que la aplicación escala. En la mayoría de los casos, el costo termina siendo significativo, por lo que una aplicación descentralizada es una opción viable en escenarios en los que no se desee invertir mucho dinero.

6.3. Ambientes y herramientas

Existen varios ecosistemas que apuntan a proveer un marco con el cuál desarrollar una aplicación descentralizada. A su vez, cada uno de ellos cuenta con herramientas especializadas para los diferentes tipos de aplicaciones.

6.3.1. IPFS

Un suite modular de protocolos para organizar y transferir datos, diseñado con los principios de 'content addressing' (recuperación de archivos en base a contenido y no en base al nombre o id) y una red peer-to-peer. Su principal caso de uso es para publicar datos como archivos, directorios y páginas web descentralizadas.[7]

Fleek Plataforma centralizada para alojar servicios. También es utilizada para almacenar datos, y hacer accesible contenido para el resto de la web mediante gateways de IPFS.[3]

OrbitDB Base de datos peer-to-peer descentralizada. Cuenta con diferentes modelos de datos, incluyendo documentos, eventos, y diccionarios clave-valor. Utiliza IPFS para guardar y sincronizar automáticamente los datos. [10]

libp2p Colección de protocolos y utilidades para facilitar la conexión y comunicación entre pares en IPFS. Entre sus herramientas, se encuentran diferentes mecanismos de seguridad, de transporte, y para descubrimiento de pares. [9]

6.3.2. Blockchain

Tecnología basada en una cadena de bloques de operaciones descentralizada y pública. Esta tecnología genera una base de datos compartida a la que tienen acceso sus participantes, los cuáles pueden rastrear cada transacción que hayan realizado.[5]

6.3.3. Alternativas

Freenet/Hyphanet Programa de código abierto para compartir datos peer-to-peer con enfoque en la protección de la privacidad. Opera en una red descentralizada, promocionando la libertad de expresión facilitando la anonimidad de los datos compartidos y eludiendo la censura.[4][6]

7. Problema detectado y/o faltante

Los servicios actuales que proveen de infraestructura tienden a ser muy costosos para las pequeñas comunidades que necesitan tener un servicio con alta disponibilidad, accesible para todos y barato de escalar. Actualmente tampoco existe un estándar para aplicaciones cuyo stack sea completamente distribuido usando peer-to-peer.

7.1. Costos

La inversión para el mantenimiento de servidores puede ser un obstáculo a la hora de proveer una red a sus usuarios cuando se trata de una aplicación pequeña o startup. En estos casos, es común sacrificar la escalabilidad en pos de mantener los costos del alojamiento de la aplicación bajos.

En el otro extremo del espectro, se encuentran las aplicaciones en declive. Debido a la falta de incentivos financieros para justificar el mantenimiento, muchas veces la solución es desconectar los servidores definitivamente. Esto es especialmente frecuente en videojuegos multijugador que no cuentan con la capacidad de crear servidores dedicados. En tales situaciones, la empresa propietaria puede desactivar los servidores, lo que resulta en que el videojuego se vuelva parcialmente o completamente inutilizable. [1]

7.2. Interrupciones del servicio

Dada la naturaleza del modelo cliente-servidor, no es infrecuente que estas redes se vuelvan inaccesibles. Esto puede ocurrir deliberadamente por temas de mantenimiento, o accidentalmente debido a modificaciones en la aplicación del servidor durante el mantenimiento o actualización de la misma.

Uno de los episodios recientes de mayor revuelo ocurrió el 4 de Octubre de 2021, día en el que la familia de aplicaciones de Meta -Whatsapp, Facebook e Instagram -, estuvo fuera de servicio por seis horas. Esto ocasionó que mas de 3500 millones de usuarios se vieran afectados. En un articulo posterior, Meta reveló que la causa se debió a una falla en la herramienta de auditoría de los comandos que se envían a la red global de datacenters de Meta, la cual evitó que se pudiera detener el comando que desconectaba los servidores de la red. [8]

7.3. Zonas de censura

Es común que aplicaciones o sitios web comunitarios sean sometidos a su censura. La centralización de los servicios ayuda a que sea mucho más fácil bloquear su acceso, dado que es más fácil identificar y bloquear un único punto de acceso. En contraste, los sistemas descentralizados, suelen ser más resistentes porque no dependen de un servidor o servicio central que pueda ser intervenido fácilmente.

Uno de los casos más conocidos y vigentes es la censura de Wikipedia. Donde algunos gobiernos bloquean su acceso en un determinado lenguaje y otros en su totalidad. [2]

8. Solución propuesta

Realizaremos la implementación de 3 casos de uso sobre los distintos ecosistemas a analizar. Presentaremos su código fuente, así como instrucciones para su deploy en las distintas plataformas. Adicionalmente presentaremos un análisis cualitativo y cuantitativo de las ventajas y desventajas de cada caso. A continuación se detalla en qué consiste cada caso de uso.

8.1. Sitio web informativo

Este sería el caso más simple, donde no se requiere ningún tipo de procesamiento. El contenido que tendrá este sitio será información sobre los casos de uso implementados, como también instrucciones o referencias de sus ecosistemas de despliegue. Será desarrollado con tecnologías puramente de front end y se experimentará con las herramientas de ambientes distribuidos y descentralizados. Esta página se irá completando a medida que avancemos en el trabajo.

8.1.1. Requisitos funcionales

- **Landing page del proyecto**: sitio web informativo donde se presente lo que se fue haciendo en este proyecto, información sobre los casos de uso y sus ecosistemas de despliegue.

8.2. Repositorio de conocimiento

Con este caso estaríamos analizando la capacidad de creación y modificación de contenido existente. La idea es que sea un servicio comunitario donde agregar información de distinta índole, similar a Wikipedia. Se podrán crear usuarios con roles de moderador y editor. Los moderadores tendrán mayor poder de decisión sobre qué contenido publicar y qué no, además de poder bloquear usuarios de ser necesario. Los editores se encargarán de agregar, eliminar y modificar el contenido del sitio buscando tener la información lo más actualizada posible. Este servicio contará con un front end, back end y una base de datos distribuidas utilizando la tecnología OrbitDB.

8.2.1. Requisitos funcionales

- **Edición:** los artículos dentro del repositorio deben ser editables por cualquier persona que ingrese al sitio, y este cambio debe verse reflejado (eventualmente) en las demás personas que accedan a ese artículo.
- **Historial de versiones:** cada artículo debe tener una lista de versiones anteriores, junto con hipervínculos con los cuáles acceder a ellas (mientras estén disponibles en la red).
- **Búsqueda:** una persona debe poder realizar una búsqueda global de todos los artículos.

8.3. Mensajero en tiempo real

Este caso se enfoca en la capacidad de la infraestructura de enfrentarse a situaciones de *tiempo real* como puede ser un chat de texto o de audio. En particular nos centraremos en el caso de chats de texto para un grupo de usuarios en donde los mensajes sean públicos. Este servicio también contará con front end, back end y una base de datos donde persistir los mensajes.

8.3.1. Requisitos funcionales

- **Usuarios:** se deben contar con usuarios que puedan iniciar sesión con una contraseña.
- **Grupos públicos:** grupos de chat de texto, donde cualquier usuario puede ingresar y ver los mensajes del resto, así como también participar enviando sus propios mensajes.

9. Metodología

Para gestionar todo el trabajo utilizaremos un tablero Kanban en donde tendremos las tareas categorizadas como a hacer (TO-DO), haciendo (DOING) y hecha (DONE). Dispondremos de tableros específicos para cada caso de uso con la misma idea.

Tendremos reuniones semanales que serán usadas como punto de control, donde revisaremos lo que hicimos la última semana y definiremos pasos a seguir para las siguientes. También nos servirá para detectar posibles ajustes o cambios de rumbo que puedan llegar a surgir.

La modalidad sería completamente virtual y asincrónica (excepto por la reunión semanal antes mencionada). Esto no quita que nos mantendremos en constante comunicación mediante un chat grupal, lo cuál pueda generar reuniones espontáneas para sesiones de pair o mob-programming.

10. Experimentación y/o validación

De los casos de uso esperamos responder las siguientes incógnitas:

Costos ¿Cuánto nos cuesta desplegar y mantener un servicio en cada ecosistema?

Facilidad de despliegue ¿Qué tan fácil es desplegar en cada ecosistema?

Viabilidad ¿Que tan viable es crear una aplicación comunitaria para cada uno de estos ecosistemas?

11. Plan de actividades

Realizamos un cronograma tentativo de la totalidad del trabajo, incluyendo el desarrollo de cada caso de uso, el despliegue en cada ecosistema y su documentación asociada.

Cada caso de uso incluye una etapa de *Discovery* en la cuál definiremos su alcance y lo desglosaremos en tareas más concretas.

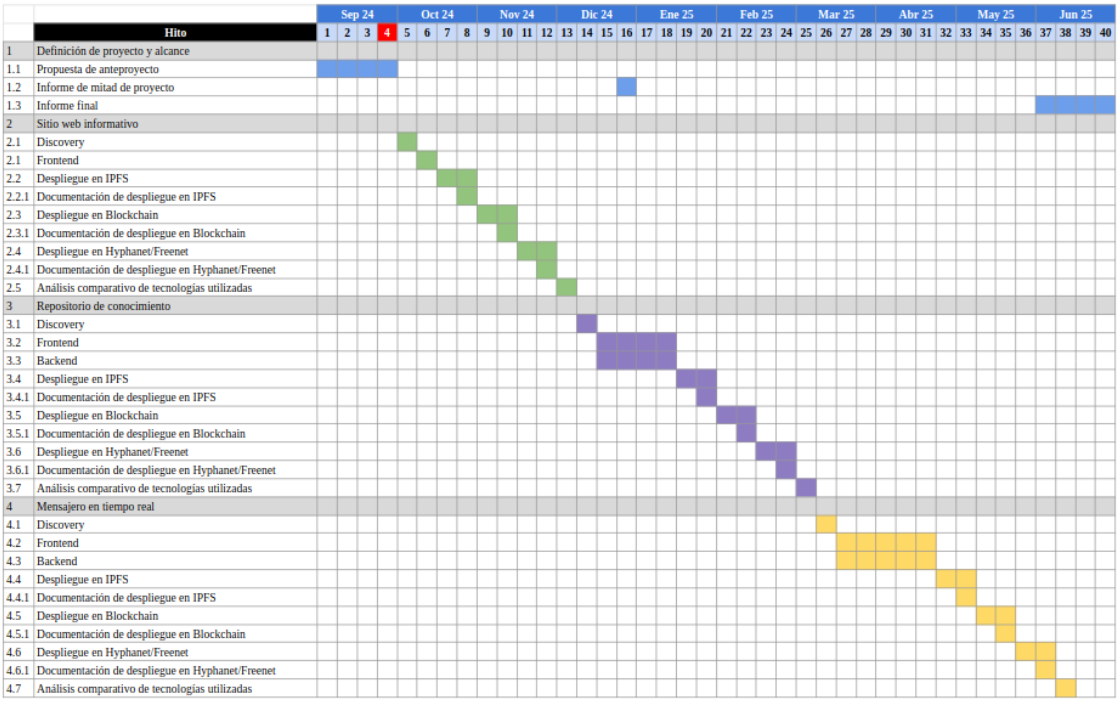

Figura 1: Cronograma tentativo

1. **Definición de proyecto y alcance**
   1.1. Propuesta de anteproyecto (Semanas 1 a 4)
   1.2. Informe de mitad de proyecto (Semanas 16 a 17)
   1.3. Informe final (Semanas 37 a 40)
2. **Sitio web informativo** (Semanas 5 a 13)
   2.1. Discovery
   2.2. Frontend
   2.3. Despliegue en IPFS
      2.3.1. Documentación de despliegue en IPFS
   2.4. Despliegue en Blockchain
      2.4.1. Documentación de despliegue en Blockchain
   2.5. Despliegue en Hyphanet/Freenet
      2.5.1. Documentación de despliegue en Hyphanet/Freenet
   2.6. Análisis comparativo de tecnologías utilizadas
3. **Repositorio de conocimiento** (Semanas 14 a 25)
   3.1. Discovery
   3.2. Frontend
   3.3. Backend
   3.4. Despliegue en IPFS
      3.4.1. Documentación de despliegue en IPFS
   3.5. Despliegue en Blockchain
      3.5.1. Documentación de despliegue en Blockchain
   3.6. Despliegue en Hyphanet/Freenet
      3.6.1. Documentación de despliegue en Hyphanet/Freenet
   3.7. Análisis comparativo de tecnologías utilizadas
4. **Mensajero en tiempo real** (Semanas 26 a 38)
   4.1. Discovery
   4.2. Frontend
   4.3. Backend
   4.4. Despliegue en IPFS
      4.4.1. Documentación de despliegue en IPFS
   4.5. Despliegue en Blockchain
      4.5.1. Documentación de despliegue en Blockchain
   4.6. Despliegue en Hyphanet/Freenet
      4.6.1. Documentación de despliegue en Hyphanet/Freenet
   4.7. Análisis comparativo de tecnologías utilizadas

12. Referencias

[1] Brnakova, J. (s.f.). *Game decommissioning: When beloved games shut down*. https://www.revolgy.com/insights/blog/game-decommissioning-when-beloved-games-get-shut-down-and-online-worlds-disappear
[2] *Censorship of Wikipedia*. (s.f.). https://en.wikipedia.org/wiki/Censorship_of_Wikipedia
[3] *Fleek*. (s.f.). https://fleek.co
[4] *Freenet*. (s.f.). https://freenet.org
[5] Hurtado, J. S. (s.f.). *Qué es Blockchain y cómo funciona la tecnología Blockchain*. Consultado el 25 de septiembre de 2024, desde https://www.iebschool.com/blog/blockchain-cadena-bloques-revoluciona-sector-financiero-finanzas
[6] *Hyphanet*. (s.f.). https://www.hyphanet.org/index.html
[7] *IPFS*. (s.f.). https://ipfs.tech
[8] Janardhan, S. (s.f.). *More details about the October 4 outage*. https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/
[9] *libp2p*. (s.f.). https://libp2p.io/
[10] *OrbitDB*. (s.f.). https://orbitdb.orghttps://orbitdb.org