Proyecto Bitcoin

[95.08] Taller de Programación I
Primer cuatrimestre de 2023

Alumno	Número de padrón	email
Daniel Spacek	88707	dspacek@fi.una.ar
Agustina Schmidt	103409	aschmidt@fi.uba.ar
Agustina Fraccaro	103199	afraccaro@fi.uba.ar
Kevin Untrojb	97866	kuntrojb@fi.uba.ar

Índice

1. Resumen **2**

2. Glosario **2**

3. Desarollo **2**
3.1. Diseño . . . 2
3.1.1. Application Manager . . . 2
3.1.2. Transaction Manager . . . 2
3.1.3. Block Broadcasting . . . 3
3.1.4. Front End . . . 3
3.2. Implementación . . . 3
3.2.1. Módulos del Nodo . . . 3
3.2.2. Flujos de operaciones . . . 3

4. Conclusiones **5**

1. Resumen

En el presente informe ser va a exponer el desarollo del proyecto Bitcoin abordando los desafíos técnicos y las decisiones de diseño que hemos enfrentado durante el proceso de desarrollo.

2. Glosario

Glosario que se utilizará en el informe:

- Nodo: referencia al Nodo Bitcoin que se desarolló en este proyecto
- Front End: FE, se refiere a la interfaz grafica
- UTXOs: es el conjunto de Unspent Transaction Outputs (UTXOs) en una blockchain, contienen información sobre los saldos en las cuentas para las transacciónes.

3. Desarollo

3.1. Diseño

Para poder diseñar el Nodo se dividió las responsabilidades de las distintas funcionalidades. Las responsabilidades son, la interfaz gráfica, escuchar a otros Nodos que envían bloques/transacciones, manejar las cuentas de los usuarios, manejar las uxtos y orquestar todo el nodo. Se propuso crear la noción de *Managers* que tomen cada una de las responsabilidades.

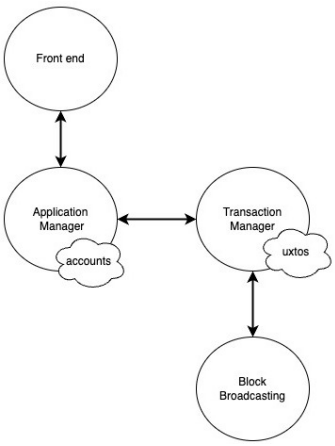

Figura 1: Relación entre los *Managers*.

3.1.1. Application Manager

El Application Manager es el que orquesta todo el nodo, es el encargado de iniciar la aplicación y cerrar todos los hilos. Además tiene como responsabilidad manejar las cuentas de los usuarios. Tiene conexión con el Front End y con el Transaccion Manager.

3.1.2. Transaction Manager

El Transaction Manager es responsable de manejar las UTXOs, ademas de ordenar los llamados concurrentes que se hacen. Tiene conexión con el hilo de Block Broadcasting y con Application Manager

3.1.3. Block Broadcasting

El Block Broadcasting es el encargado de escuchar nuevas transacciones y bloques. Al procesar nuevos datos, envía la información al Transaction Manager para obtener los UTXOs.

3.1.4. Front End

El Front End es la interfaz gráfica, tiene conexión con el Application Manager.

3.2. Implementación

Para implementar la comunicación entre Managers se utilizó un patrón de actors con *actores*, de esta forma se utilizan los canales como buffer de mensajes para sincronizar los llamados asincrónicos que pueden tener, garantizando exclusión mutua sin la necesidad de utilizar Locks. Cada uno de los Manager tiene un hilo que escucha cada mensaje que se recibe. Se implementó un mensaje dedicado al cerrar todos los hilos generados utilizando para el cierre de la aplicación.

3.2.1. Módulos del Nodo

El Proyecto está organizado en una estructura de carpetas y archivos que refleja la arquitectura de los módulos implementados. En el siguiente diagrama, se detalla la estructura de directorios junto con los módulos y archivos asociados

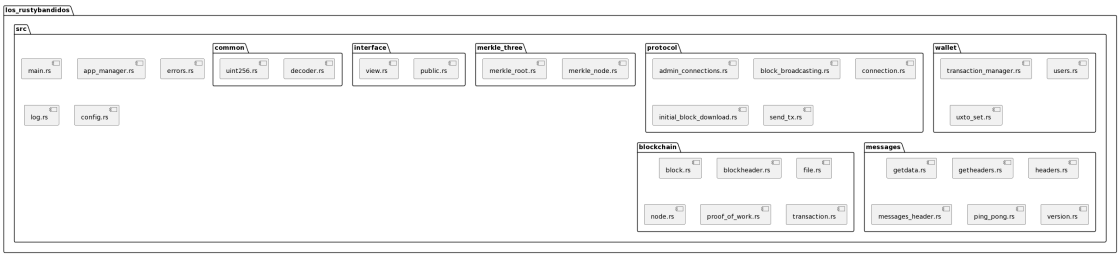

Figura 2: Diagrama de los módulos de la arquitectura

3.2.2. Flujos de operaciones

A continuación mostraremos algunos flujos de la aplicación:

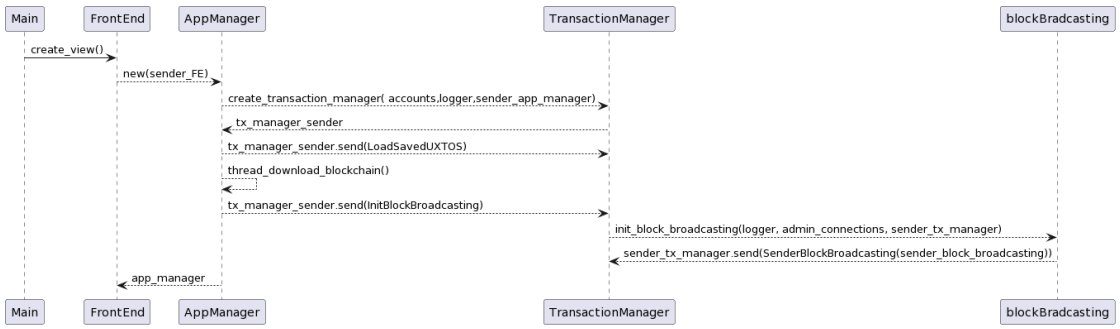

Figura 3: Flujo de llamadas para Iniciar el Nodo.

La figura 3 muestra el proceso de inicio del Nodo y creación de los Managers. Inicia desde el archivo *main.rs*, este hace un llamado al FrontEnd para crear la interfaz gráfica y se crea el App Manager enviandole un sender para que se puedan comunicar y enviar mensajes. El App Manager crea el Transaction Manager y este le devuelve el sender para que estén comunicados.

Posteriormente el App Manager empieza a descargar el resto de la blockchain y le envía un mensaje al Transaction Manager para que inicie el Block Broadcasting. El Transaction Manager inicia el block bradcasting enviando el sender para estar comunicados, finalmente el Transaction Manager recibe el sender del Block Broadcasting.

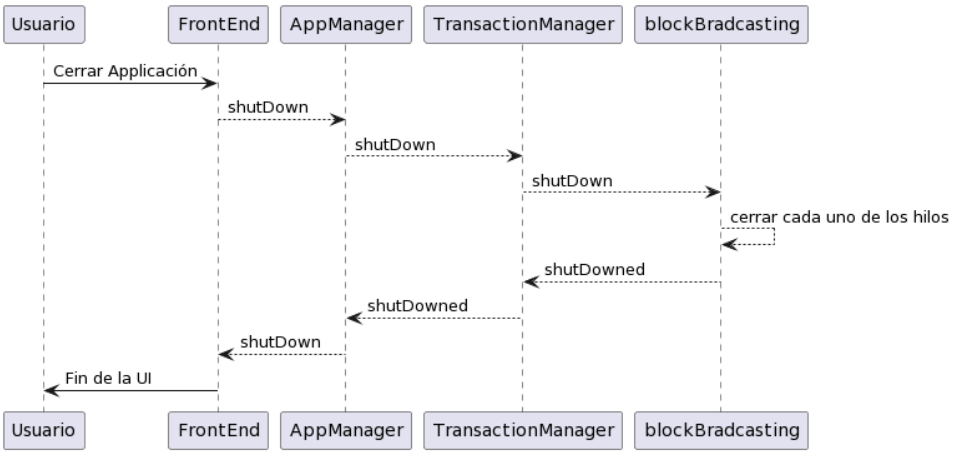

Figura 4: Flujo de llamadas para cerrar el Nodo.

La figura 4 muestra el flujo cuando el usuario cierra la aplicación presionando un botón en la vista, en ese momento se desencadenan llamados a través de todos los componentes para cerrar los hilos, el FrontEnd comunica al App Manager que la aplicación debe cerrarse, el App Manager a su vez informa al Transaction Manager que debe cerrarse y el Transaction Manager avisa al Block Broadcasting, que es responsable de los hilos de las conexiones a los nodos, que se deben cerrar.

Una vez cerradas todas las conexiones, el Block Broadcasting notifica al Transaction Manager que se ha completado el cierre, el Transaction Manager informa al App Manager que se ha cerrado, luego, el App Manager comunica al Front End que la aplicación se está cerrando. Finalmente, la interfaz gráfica se cierra, finalizando así la aplicación.

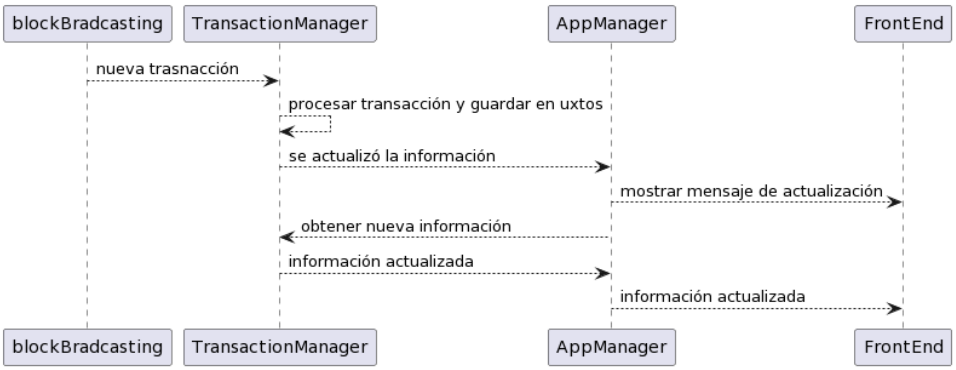

Figura 5: Flujo de llamadas al recibir una nueva transacción.

La figura 5 muestra cuando el Block Broadcasting recibe una nueva transacción desde la red de nodos Bitcoin. La transacción se envía al Transaction Manager para su procesamiento. Posteriormente el Transaction Manager se comunica con el App Manager para notificar la llegada de la transacción. El App Manager solicita los cambios al Transaction Manager y éste envía los cambios al App Manager. Finalmente, el App Manager envía los cambios al Front End, que se encarga de actualizar la interfaz de usuario para reflejar la nueva transacción.

Este flujo garantiza que la nueva transacción sea recibida y procesada correctamente en la aplicación, y que los cambios se reflejen en el Front End para que el usuario pueda ver la transacción actualizada.

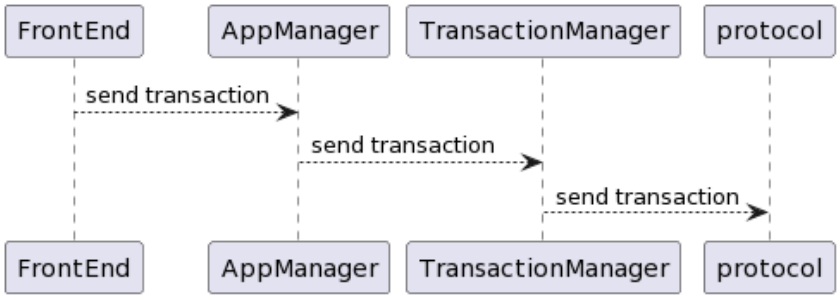

Figura 6: Flujo de llamadas al crear una nueva transacción.

La figura 6 muestra la acción de enviar una transacción, la misma comienza en el Front End. El Front End realiza una llamada al App Manager para solicitar el envío de la transacción. El App Manager, a su vez, invoca al Transaction Manager para gestionar esta solicitud. El Transaction Manager utiliza el protocolo para enviar la transacción a través de los nodos.

4. Conclusiones

El proyecto nodo Bitcoin en lenguaje Rust nos brindó diversos aprendizajes. Durante el desarrollo del proyecto, nos encontramos con dificultades significativas al abordar la implementación de las funciones de serialización en Rust. Estos aspectos específicos del protocolo Bitcoin resultaron desafiantes y consumieron una cantidad considerable de tiempo y esfuerzo. Sin embargo, estas dificultades nos permitieron adquirir un profundo conocimiento de los detalles técnicos de la serialización en el contexto de Bitcoin.

A pesar de las dificultades mencionadas, el proyecto nos dejó enseñanzas sobre los protocolos asociados a Bitcoin. A medida que trabajamos en la implementación del nodo, pudimos profundizar en el estudio de la estructura de las transacciones, la validación de bloques y otros aspectos clave del protocolo.

Otro aspecto destacado del proyecto fue la utilización de actores y la programación concurrente en Rust. Estas técnicas nos permitieron organizar el código de manera efectiva y gestionar las tareas simultáneas de manera eficiente. La programación concurrente fue especialmente útil para mejorar la escalabilidad y la capacidad de respuesta del sistema. La experiencia de trabajar con actores y programación concurrente en Rust nos proporcionó una sólida base para abordar proyectos futuros en entornos de alta concurrencia.