Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

APLICACIONES DE BIG DATA PARA ANÁLISIS DE GRAFOS DE TRANSACCIONES DE TOKENS NO FUNGIBLES

TRABAJO FIN DE MÁSTER

Rigoberto Valentín López Fernández

2023

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

APLICACIONES DE BIG DATA PARA ANÁLISIS DE GRAFOS DE TRANSACCIONES DE TOKENS NO FUNGIBLES

Autor
Rigoberto Valentín López Fernández

Tutor
Félix Cuadrado Latasa

Departamento de Ingeniería de Sistemas Telemáticos

2023

Resumen

Los Tokens No Fungibles, o NFTs por sus siglas en inglés, son identificadores digitales únicos que no pueden ser copiados, subdivididos o sustituidos. Existe un comercio online de estos tokens, a menudo con criptomonedas, y suelen estar codificados en contratos inteligentes dentro de una cadena de bloques. Los NFTs se utilizan principalmente para representar tanto objetos digitales como reales, ya que certifican la autenticidad y propiedad del objeto representado.

El mercado de NFTs ha crecido de forma significativa en los últimos años, donde cada vez más objetos son intercambiados de esta forma. Existen colecciones que representan obras de arte, artículos de juegos, objetos coleccionables, utilitarios, entre otros. Las transacciones y actores del mercado de NFTs se pueden modelar como una red o grafo dinámico, donde están registrados los eventos que causan modificaciones en dicho grafo.

En este trabajo se realiza un análisis de las características del mercado de NFTs haciendo uso de datos de transacciones que han ocurrido entre noviembre de 2017 y abril de 2021. Para realizar esta investigación, inicialmente se realiza un estudio de diferentes herramientas de Big Data que permiten modelar grafos y ejecutar análisis sobre estos. En segundo lugar, se realiza una caracterización estadística de los datos disponibles, desglosando los datos en las diferentes categorías de NFTs. A continuación, se diseña un conjunto de algoritmos para obtener propiedades de interés de los comerciantes de NFTs y de los propios tokens. Finalmente, se ejecutan una serie de análisis temporales sobre los datos disponibles, con el objetivo de generar diferentes perspectivas que permitan una mayor comprensión del comportamiento de estos sistemas.

Abstract

Non-Fungible Tokens, or NFTs for short, are unique digital identifiers that cannot be copied, subdivided, or replaced. These tokens are traded online, often with cryptocurrencies, and they are usually encoded in smart contracts within a blockchain. NFTs are primarily used to represent both digital and real objects, as they certify the authenticity and ownership of the represented object.

The NFT market has grown significantly in recent years, with more and more items being traded in this way. There are collections that represent artwork, in-game items, collectibles, utility objects, among others. The transactions and actors of the NFT market can be modeled as a network or dynamic graph, where the events that cause graph updates are registered.

In this work, an analysis of the characteristics of the NFT market is carried out using data from transactions that have occurred between June 2017 and April 2021. Initially, a study of different Big Data tools that allow the modeling and analysis of graphs is carried out. Second, a statistical characterization of the available data is performed, breaking down the data into the different NFT categories. Next, a set of algorithms is designed to derive properties of interest from NFT traders and tokens. Finally, a series of temporary analyzes are carried out on the available data, with the aim of generating different perspectives that allow a better understanding of the behavior of these systems.

Índice general

Índice de figuras

Índice de tablas

Siglas

NFT	Non-Fungible Token
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
RDD	Resilient Distributed Dataset

1. Introducción

Desde el acuñamiento del primer Token No Fungible, "Quantum", en el año 2014, el interés por estos tokens ha ido creciendo. Los NFTs surgen como vía para certificar la proveniencia y propiedad de activos. Son asociados a activos de diferentes categorías como coleccionables, arte, juegos, metaverso y otros artículos.

En los últimos años se han roto récords impresionantes en transacciones de NFTs, por ejemplo, el NFT "The Merge" fue vendido por 91.8 millones de dólares entre el 2 y el 4 de diciembre de 2021. "The Merge" es una obra de arte digital única en el mundo de las NFT porque, en lugar de ser una sola obra, "The Merge" es considerado arte fragmentado. Otro ejemplo es "Everydays – The First 5000 Days", una obra de arte creada por el conocido artista Mike Winklemann y vendida en la casa de subastas Christie's el 11 de marzo de 2021, por 69.3 millones de dólares [1].

En la investigación realizada, se han encontrado relativamente pocos recursos sobre el modelado y análisis del mercado de NFTs a gran escala. En el artículo "Mapping the NFT revolution: market trends, trade networks, and visual features" [2], se realiza una recopilación de datos de transacciones de NFTs, provenientes de la entidad NonFungible Corporation y de las APIs de código libre CryptoKitties sales, Gods-Unchained, Decentraland, and OpenSea. El estudio definido en [2], es el único encontrado hasta el momento donde se haga un análisis del mercado de NFTs teniendo en cuenta un número importante de colecciones de estos tokens.

En este trabajo, se utiliza el conjunto de datos ofrecidos en [2] para realizar un modelado del mercado de NFTs en una plataforma de Big Data. Este conjunto está compuesto por datos relativos a 6.1 millones de transacciones de 4.7 millones de NFTs, entre el 23 de noviembre de 2017 y el 27 de abril de 2021, obtenidos principalmente de las cadenas de bloques Ethereum y WAX.

El propósito de este proyecto es caracterizar los elementos que intervienen en las transacciones de NFTs en diferentes instantes y ventanas de tiempo.

Objetivos

En esta sección se presentan el objetivo general y los objetivos específicos de este trabajo. El cumplimiento de estos objetivos da respuesta al problema de investigación planteado.

Objetivo general

Modelar un conjunto de transacciones representativas del mercado de NFTs a lo largo de un intervalo de tiempo en una plataforma de Big Data.

Objetivos específicos

- Caracterizar el conjunto de datos disponibles estadísticamente.
- Diseñar algoritmos para obtener las propiedades de las entidades que conforman el mercado de NFTs.
- Ingerir el conjunto de datos en una plataforma de Big Data para su posterior análisis.
- Realizar análisis temporales sobre los datos correspondientes a las transacciones de NFTs.

2. Estado del arte

En el presente capítulo se hace una revisión de los conceptos teóricos necesarios para comprender la investigación realizada. Se lleva a cabo también un estudio de las herramientas disponibles para el modelado y análisis de grafos. Por último, se realiza una caracterización del mercado de NFTs.

2.1 Tokens No Fungibles

2.1.1 Historia de los NFT

El concepto de NFT proviene originalmente de un token de Ethereum. Sin embargo, mucho antes de que existiera Ethereum, el concepto que se convirtió en la fuerza impulsora de los NFT se publicó en [3], donde se introdujo el concepto de "Colored Coins" para la cadena de bloques de Bitcoin.

La idea de Colored Coins era describir una clase de métodos para representar y administrar activos del mundo real en la cadena de bloques, para demostrar la propiedad de esos activos; similar a los Bitcoins regulares, pero con un elemento "token" agregado que determina su uso, haciéndolos segregados y únicos.

El 3 de mayo de 2014, el artista digital Kevin McCoy acuñó el primer NFT conocido, “Quantum”, en la cadena de bloques Namecoin. “Quantum” es una imagen digital de un octágono pixelado que hipnóticamente cambia de color y realiza pulsaciones de una manera que recuerda a un pulpo.

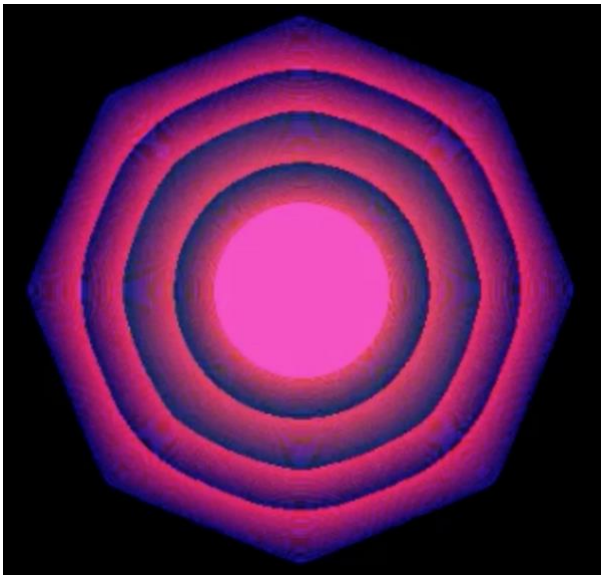

Figura 1. Primera NFT conocida Quantum [4].

En los años siguientes, la plataforma Counterparty (Bitcoin 2.0) se estableció y ganó terreno como una plataforma que permitió la creación de activos digitales. Sin embargo,

la cadena de bloques de Bitcoin nunca tuvo el objetivo de ser empleada como una base de datos para tokens que representan la propiedad de los activos y, por lo tanto, comenzó la migración de los NFTs a la cadena de bloques Ethereum.

El gran cambio de NFTs a Ethereum fue respaldado con la introducción de un conjunto de estándares de tokens, lo que permitió la creación de tokens por parte de los desarrolladores. El estándar de token es una subsidiaria del estándar de contratos inteligentes, incluido para informar a los desarrolladores sobre cómo crear, emitir e implementar nuevos tokens en línea con la tecnología de cadenas de bloques subyacente [5].

John Watkinson y Matt Hall, dos desarrolladores de software, crearon su propia serie generativa de NFTs en la cadena de bloques Ethereum, que denominaron CryptoPunks. Los CryptoPunks son considerados algunos de los primeros NFT creados y fueron ofrecidos originalmente de forma gratuita. El proyecto experimental, limitado a 10.000 piezas, donde no existen dos personajes iguales, se inspiró en la cultura *punk* londinense y el movimiento *cyberpunk*.

Durante el hackathon más grande del mundo para el ecosistema Ethereum, el estudio Axiom Zen, con sede en Vancouver, presentó CryptoKitties. CryptoKitties es un juego virtual basado en la cadena de bloques Ethereum. El juego permite a los jugadores adoptar, criar e intercambiar gatos virtuales, almacenándolos en billeteras criptográficas. Después de su anuncio, no pasó mucho tiempo antes de que el juego se convirtiera en una sensación viral, volviéndose tan popular que CryptoKitties obstruyó la cadena de bloques Ethereum y se comenzaron a obtener grandes ganancias [6].

Tras el gran éxito de CryptoKitties, los NFTs asociados a juegos realmente comenzaron a ganar impulso, atrayendo cada vez más la atención del público. Los proyectos de NFT de juegos y metaverso estuvieron en el centro de atención y el primero en abrir camino en este espacio fue Decentraland (MANA), una plataforma de realidad virtual descentralizada en la cadena de bloques Ethereum [5]. Decentraland es una plataforma de juegos de mundo abierto que permite a los jugadores explorar, jugar, construir, recopilar artículos y más, y todo lo que se encuentre, gane y construya allí, pasa a ser propiedad en la cadena de bloques. En 2018 se crea el estándar *Ethereum Request for Comments* 721 (ERC-721), que define una interfaz estándar que establece cómo crear tokens no fungibles en Ethereum, denominada ERC-721 [7].

El año 2021 se convirtió en el año de los NFTs, puesto que hubo una gran explosión y aumento en la oferta y demanda de estos tokens. Uno de los factores más importantes de este auge fueron los grandes cambios que se produjeron en el mercado del arte y en la industria en general, cuando las prestigiosas casas de subastas Christie's y Sotheby's,

además de llevar las subastas al mundo en línea, comenzaron a vender NFTs de arte. Esto condujo a la venta récord de Christie's de Beeple's Everydays: the First 5000 Days NFT por 69 millones de dólares. Esta venta, hecha por una casa de subastas tan prestigiosa, validó significativamente el mercado de NFTs [5].

Además del aumento en la demanda de NFTs que resultó de la famosa subasta de Christie's, otro efecto colateral fue que otras cadenas de bloques se involucraron y comenzaron sus propias versiones de NFTs. Entre estas cadenas de bloques se encuentran Cardano, Solano, Tezos y Flow. Con estas plataformas más nuevas para NFTs, se establecieron algunos estándares nuevos para garantizar la autenticidad y la singularidad de los activos digitales creados. Cardano, por ejemplo, utiliza un protocolo de prueba de participación en lugar de la prueba de trabajo de Ethereum, que se informa que consume mucha menos energía que las criptomonedas de prueba de trabajo, como Bitcoin y Ethereum [8]. Cardano se presenta como más rápido y flexible que Bitcoin y más seguro y escalable que Ethereum, y puede procesar 257 transacciones/segundo, en comparación con las 15 de Ethereum. Tanto Cardano como Ethereum permiten la ejecución de contratos inteligentes [9].

Hacia finales de 2021, una vez que Facebook cambió su nombre a Meta y pasó al metaverso, el aumento en la demanda de NFTs, especialmente dentro del metaverso, fue notable.

2.2 Teoría de Grafos

El mercado de NFTs puede ser enfocado como una red o grafo muy grande, donde existen dos tipos de nodos: Traders y NFTs. Se establecen conexiones entre Traders cuando se realiza una transacción y relaciones de propiedad entre Traders y los NFTs que compran.

En esta sección, se presentan los conceptos teóricos asociados a grafos, necesarios para comprender el modelado del mercado de NFTs que se realiza en el presente trabajo investigativo.

Los orígenes de la teoría de grafos se remontan al trabajo de Euler sobre el problema de los puentes de Königsberg (1735), que posteriormente condujo al concepto de un gráfico euleriano [10].

2.2.1 Grafo simple

Según [11], un grafo simple G consta de un conjunto finito no vacío V(G) de elementos llamados vértices (o nodos), y un conjunto finito E(G) de distintos pares desordenados de elementos distintos de V(G) llamados aristas. Se denomina V(G) al conjunto de vértices y E(G) al conjunto de aristas de G. Una arista {v, w} se dice que une los vértices

v y w, y generalmente se abrevia como vw. Un grafo simple está compuesto por una sola arista como máximo entre cada par de vértices.

Por otro lado, se define un grafo general, donde no existe un límite en el número de aristas que existe entre los pares de vértices.

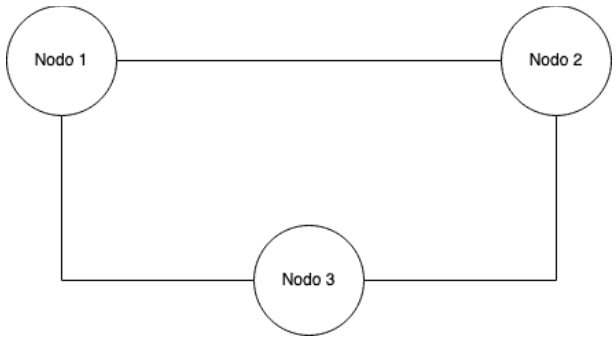

Figura 2. Grafo simple.

2.2.2 Grafo general

Un grafo G consta de un conjunto finito no vacío V(G) de elementos llamados vértices, y una familia finita E(G) de pares no ordenados de elementos (no necesariamente distintos) de V(G) llamadas aristas; el uso de la palabra 'familia' permite la existencia de múltiples aristas. Se denomina V(G) al conjunto de vértices y E(G) a la familia de aristas de G. Una arista {v, w} se dice que une los vértices v y w, y nuevamente se abrevia como vw [11].

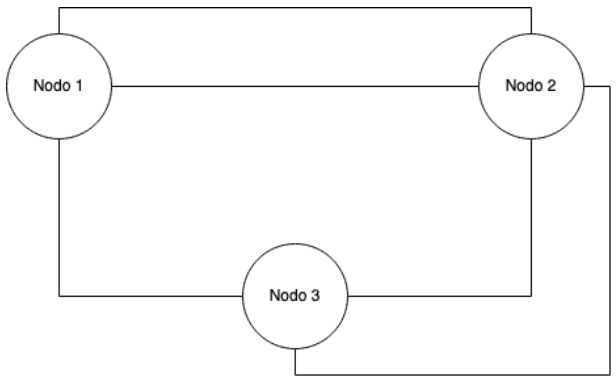

Figura 3. Grafo general.

2.2.3 Grafo dirigido

Un grafo dirigido D consta de un conjunto finito no vacío V(D) de elementos llamados vértices, y una familia finita A(D) de pares ordenados de elementos V(D) llamados arcos. Se denomina V(D) al conjunto de vértices y A(D) a la familia de arcos de D. Un arco (v, w) es usualmente abreviado como vw [11].

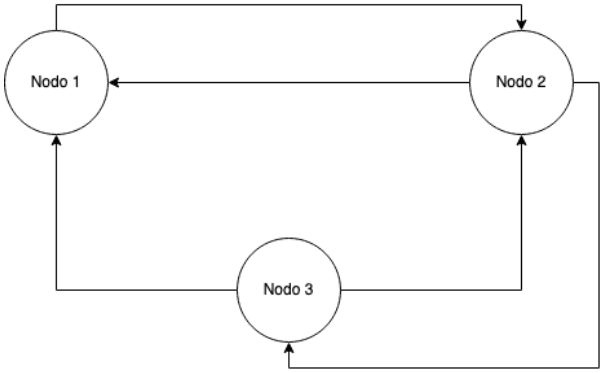

Figura 4. Grafo dirigido.

2.2.4 Grafo ponderado

Un grafo ponderado es aquel donde cada arista tiene un valor numérico asociado, denominado peso. Normalmente, los pesos son valores no negativos.

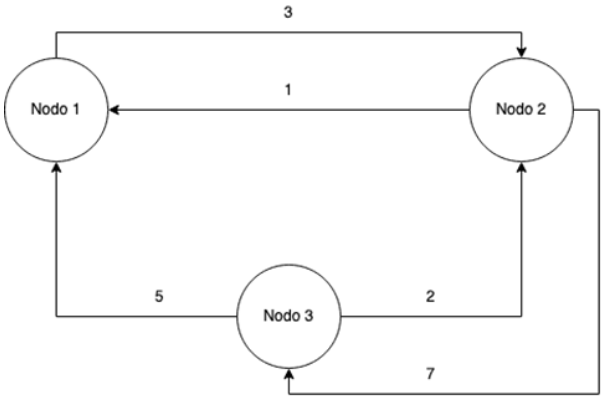

Figura 5. Grafo ponderado dirigido.

2.2.5 Grafo temporal

Un grafo temporal es un tipo de grafo cuyos componentes, es decir, vértices y aristas, pueden cambiar con el tiempo. En diferentes instantes de tiempo, pueden agregarse nuevos nodos, mientras que las conexiones entre los mismos pueden surgir o desaparecer. Las propiedades asociadas a vértices y aristas también pueden variar con la dimensión temporal.

Aunque los grafos estáticos han sido ampliamente estudiados, todavía se está lejos de tener un conjunto concreto de principios estructurales y algorítmicos de los grafos temporales. Muchas propiedades y problemas de grafos se vuelven radicalmente diferentes y generalmente sustancialmente más difíciles cuando se les agrega una dimensión de temporal. Existe un conjunto rico y de rápido crecimiento de sistemas y aplicaciones modernos que se pueden modelar de forma natural a través de grafos

temporales. Esto motiva aún más la necesidad del desarrollo de una extensión de la teoría de grafos [12].

2.2.6 Propiedades de vértices y aristas

A continuación, se presentan las algunas de las definiciones que caracterizan los grafos:

Adyacencia: Dos vértices de un grafo son adyacentes si existe una arista que los unes, mientras que ambos vértices son incidentes con dicha arista. Dos aristas distintas son adyacentes si tienen un vértice común.

Grado: El grado de un vértice indica el número de aristas incidentes que tiene dicho vértice. Si el grafo es dirigido, se puede diferenciar entre grado de entrada y salida, correspondiente a aristas dirigidas al vértice de referencia y a aristas que parten de dicho nodo, respectivamente.

Ponderación: Valor numérico asociado a cada arista.

Poder: El poder de un vértice es la suma de los pesos de todas las aristas incidentes a dicho vértice.

En esta investigación, los grafos que se modelan son de tipo general, puesto que puede ocurrir que un Trader compre a otro Trader más de un NFT, por lo que se establecerán varias aristas entre ambos. Estos grafos son también dirigidos, debido a que las relaciones entre nodos de tipo Trader se establecen con origen comprador y destino vendedor. Cada arista tiene un peso correspondiente al total de artículos que un comprador adquirió de un vendedor, por lo que es un grafo ponderado. Se define el poder de un nodo de tipo Trader como el número total de compras y ventas que realiza dicho nodo. Por último, el modelado se hace representando un grafo temporal.

2.3 Herramientas de modelado de grafos

2.3.1 Neo4j

Neo4j es una base de datos de grafos comercial. Una base de datos de grafos, en lugar de tener filas y columnas, tiene nodos, aristas y propiedades. Estos sistemas de almacenamiento constituyen un tipo de bases de datos no relacionales, o NoSQL, lo cual significa que Neo4j cumple las siguientes propiedades:

- No requiere mapeo relacional de objetos ni normalización de datos.
- No tiene un esquema fijo.
- Ofrece estructuras de datos heterogéneas en el mismo dominio.

A diferencia de otras bases de datos no relacionales, Neo4j tiene soporte transaccional e implementa las propiedades ACID.

Modelado de datos

En Neo4J no existe un conjunto de prescripciones impuestas que restrinjan los datos que acepta. Es decir, el modelo de datos está implícito en los datos que almacena [13]. En este sistema se crean entidades, sus relaciones y las propiedades de ambos. En la figura 6 [14], se muestra un ejemplo de grafo modelado en Neo4j de un sistema de correo electrónico.

Las bases de datos de grafos se pueden diferenciar en cuanto a si ofrecen almacenamiento y procesamiento nativo o no. El procesamiento nativo implica que estas bases de datos tienen una estructura de datos subyacente diseñada específicamente para el almacenamiento y gestión de grafos. Estas estructuras se diseñan para maximizar la rapidez de recorrido del grafo ante algoritmos arbitrarios [13]. En la figura 7, se puede observar la estructura definida para Neo4j, donde todos los niveles en la arquitectura están optimizados para el almacenamiento de datos de grafos.

Neo4j tiene capacidad nativa de procesamiento de grafos puesto que utiliza adyacencias sin índices. Esto significa que cada nodo referencia directamente sus nodos adyacentes actuando como micro índices para los nodos cercanos. Este patrón de diseño permite recorrer los grafos de forma rápida y eficiente.

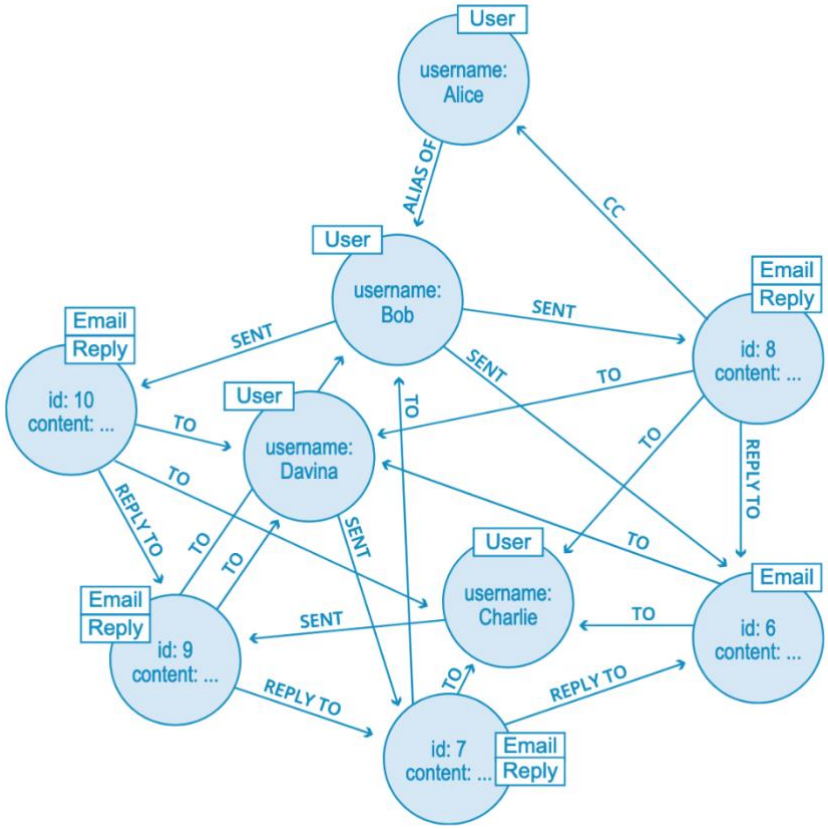

Figura 6. Grafo definido en Neo4j para el modelado de un sistema de correo electrónico [14].

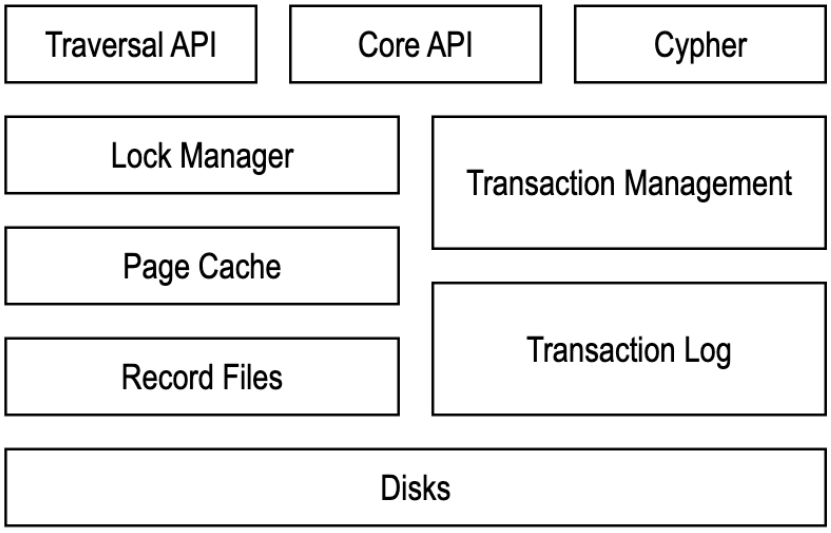

Figura 7. Estructura de Neo4j para el almacenamiento nativo de grafos [14].

Lenguaje de consulta

Neo4j utiliza Cypher como lenguaje de consulta. Cypher es un lenguaje de consulta declarativo que utiliza ASCII-Art para representar patrones gráficos visuales para encontrar o actualizar datos en Neo4j. Está basado en el poder de SQL, pero optimizado específicamente para grafos [15].

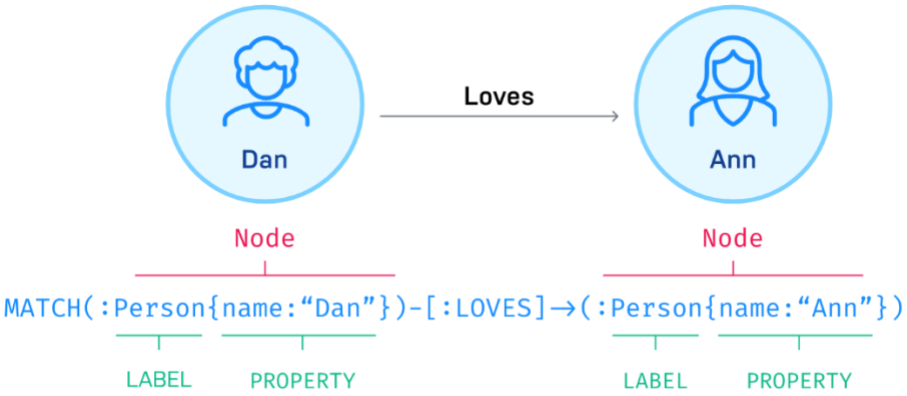

Figura 8. Ejemplo del lenguaje de consulta Cypher en Neo4j [16].

Fuentes de entrada de información

Neo4j permite importar datos de archivos de texto. Además, dispone de un comando para cargar información expuesta en formato JSON a través de una API. También brinda la posibilidad de importar datos provenientes de un sistema de bases de datos relacional.

Neo4j ofrece los siguientes conectores para la interconexión con sistemas externos [17]:

- Conector para Apache Spark

- Conector para Apache Kafka
- Conector para plataformas de Inteligencia de Negocio

Algoritmos embebidos

Neo4j incluye una biblioteca abierta de algoritmos de grafos de alto rendimiento que revelan los patrones y estructuras ocultos en sus datos conectados [18]. Estos algoritmos se pueden clasificar en búsqueda de trayectos, centralidad y detección de comunidad. Se presenta a continuación una lista de algunos de los algoritmos ofrecidos en Neo4j y optimizados para este sistema.

Búsqueda de trayectos

- Búsqueda primero en anchura
- Búsqueda primero en profundidad
- Trayecto más corto

Centralidad

- PageRank
- Grado

Detección de comunidad

- Propagación de etiquetas
- Componentes fuertemente conectados

Análisis temporal

Si bien las bases de datos de grafos están orientadas a gestionar datos orientados a estos elementos, no tienen en cuenta la dimensión temporal. Sin embargo, es común que los elementos en un grafo varíen en el tiempo.

Neo4j no incluye, de forma nativa, una vía para realizar consultas temporales. Esto implica que las consultas para un análisis temporal de las modificaciones de un grafo pueden ser complicadas y de gran verbosidad.

Recientemente, se ha desarrollado un lenguaje de consultas de grafos temporales, T-Cypher. T-Cypher amplía el lenguaje Cypher con construcciones temporales fáciles de utilizar [19]. Este lenguaje puede ser integrado con Neo4j mediante la traducción de consultas T-Cypher a consultas Cypher. Sin embargo, este método no constituye una opción óptima en cuanto a eficiencia, debido a que Neo4j no provee soporte nativo para T-Cypher.

2.3.2 Spark GraphX

Spark GraphX es una de las bibliotecas que se encuentra sobre del *core* de Spark. GraphX extiende la estructura de datos básica de Spark, los *Resilient Distributed Datasets* (RDD), para introducir una abstracción de tipo Grafo: un multigrafo dirigido con propiedades asignadas a vértices y aristas [20].

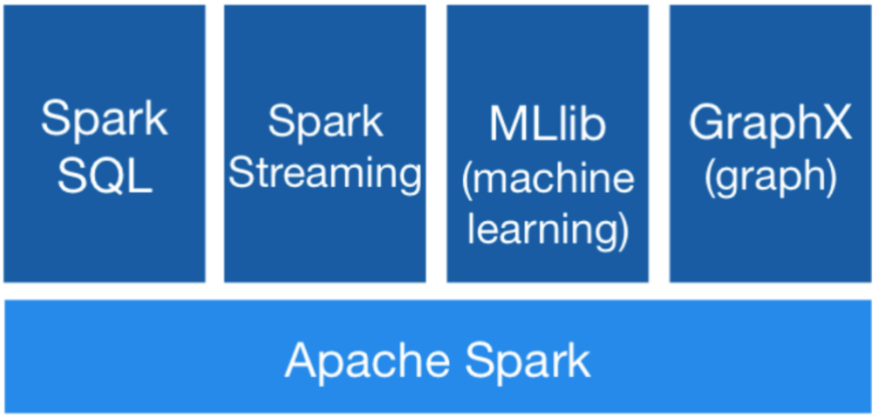

Figura 9. Arquitectura de Apache Spark.

GraphX no es una base de datos de grafos, sino que es un sistema de procesamiento de grafos. Este sistema está optimizado para ejecutar algoritmos sobre todo el grafo de forma paralela a una escala masiva [21]. Si se compara con Neo4j, Spark GraphX puede ser considerado un Sistema de Procesamiento Analítico Online (OLAP), mientras que Neo4j es un ejemplo de un Sistema de Procesamiento Transaccional Online (OLTP).

Modelado de datos

GraphX separa los elementos del grafo siguiendo la lógica "corte de vértice", lo cual implica que las aristas son distribuidas equitativamente entre los nodos que conforman el clúster de Spark. Esto permite balancear los datos del grafo en el clúster.

La información de vértices, aristas y propiedades se encuentra en disco o ya cargadas en memoria organizadas en RDDs. Esta información es luego convertida en un objeto Graph.

En última instancia, GraphX almacena las aristas del grafo en una tabla y los vértices en otra. Esto permite que los algoritmos implementados en GraphX recorran grafos de manera eficiente, a lo largo de las aristas de un vértice a otro, o como tablas de aristas o vértices [21].

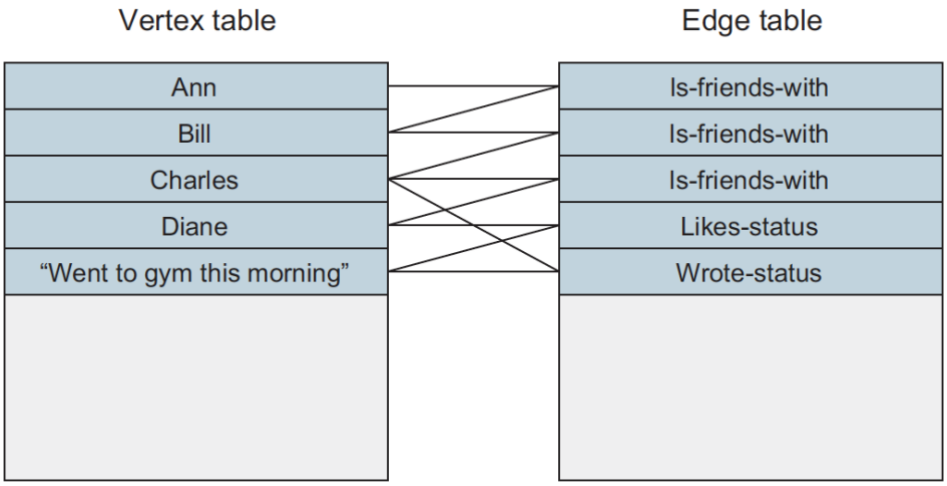

Figura 10. Distribución de vértices y aristas en GraphX [21].

Operadores de grafo

Como se ha explicado anteriormente, GraphX no es una base de datos de grafos. Debido a esto, no cuenta con un lenguaje de consulta. En GraphX se definen un conjunto de operadores de grafos para realizar el procesamiento deseado sobre estos. Los operadores se definen sobre la abstracción Graph que brinda la biblioteca. Se muestra a continuación una lista de algunos operadores:

Información sobre el grafo

- numVertices: Permite obtener el número de vértices
- degrees: Permite obtener el grado de un vértice

Vista del grafo como colecciones

- vertices: Genera una colección con los vértices del grafo
- edges: Genera una colección con las aristas del grafo

Transformación de atributos de vértices y aristas

- mapVertices: Genera un nuevo grafo con las propiedades modificadas de los vértices por la función map definida
- mapEdges: Genera un nuevo grafo con las propiedades modificadas de las aristas por la función map definida

Fuentes de entrada de información

GraphX hace uso del *core* de Spark por lo que GraphX no importa los datos directamente. Como ya se ha mencionado, una vez que Spark ha cargado los datos en memoria, GraphX puede convertirlos en grafos. Spark permite ingerir datos desde

sistemas de archivos (local, Hadoop), desde almacenamiento *cloud* (Cloud Storage, Amazon S3), desde colas de mensajería (Kafka, Kinesis) y desde bases de datos relacionales y NoSQL (MySQL, MongoDB), entre otros.

Escala de datos

Spark es un motor para procesamiento de datos a gran escala. Su capacidad de escalado horizontal permite procesar grandes cantidades de datos.

Algoritmos embebidos

GraphX incluye un conjunto de algoritmos de grafos para simplificar las tareas de análisis. Los algoritmos están contenidos en el paquete org.apache.spark.graphx.lib y se puede acceder a ellos directamente como métodos en Graph a través de GraphOps [20].

Los algoritmos definidos en GraphX son:

- PageRank
- Componentes conectados
- Conteo de triángulos

Análisis temporal

Los sistemas de procesamiento de grafos como GraphX se emplean principalmente en el minado de grafos, mientras que carecen de soporte para un modelo expresivo de grafos como el modelo de grafo con propiedades y un lenguaje de consulta declarativo [22]. Tampoco brindan soporte integrado para grafos temporales y su análisis, por lo que la gestión y el uso de la información temporal tiene que ser gestionada por las aplicaciones [23].

2.3.3 Raphtory

Raphtory es una plataforma de código abierto para análisis en tiempo real de grafos temporales distribuidos, que le permite cargar y procesar grandes conjuntos de datos dinámicos a lo largo del tiempo. Raphtory está financiado en parte por The Alan Turing Institute y desarrollado por Pometry, The QMUL Networks Group y una comunidad de colaboradores independientes [24].

Los grafos temporales son aquellos donde los elementos de grafos, es decir, vértices, aristas y propiedades, varían con el tiempo.

Modelado de datos

La entrada de Raphtory está compuesta por registros que identifican cambios en el grafo. Estos cambios incluyen modificaciones en vértices, aristas y propiedades de

ambos y que suceden en instantes de tiempo específicos. Se almacena un registro histórico de las modificaciones del grafo, incluyendo las propiedades de nodos y aristas.

Raphtory sigue un enfoque centrado en vértice, donde cada vértice tiene información local sobre el grafo, solo conoce sus vecindades. Para complementar esto, los vértices pueden enviar mensajes a sus vecinos [25]. Específicamente, cada nodo conoce:

- Sus actualizaciones históricas y las de sus propiedades, incluyendo el instante de tiempo de cada modificación.
- La historia y propiedades de sus aristas incidentes, tanto de entrada como de salida, ya que Raphtory modela grafos dirigidos.
- Estado algorítmico propio. A los nodos se les puede asignar un estado determinado cuando se ejecuta un algoritmo.

Algoritmos embebidos

Debido al enfoque centrado en vértice, hay algoritmos que necesitan hacer uso de la comunicación entre los vértices. Atendiendo a esto, los algoritmos que se definen en Raphtory se pueden clasificar atendiendo al número de mensajes que necesitan enviar los vértices en:

Cero pasos: No es necesario enviar ningún mensaje, con la información disponible en cada nodo es suficiente.

Un paso: Estos algoritmos requieren el envío de exactamente un mensaje. Por ejemplo, cuando solamente es necesario conocer un valor de estado o propiedad de sus vecinos.

Iterativos: No se sabe a priori cuántas iteraciones serán necesarias. En estos casos se envían mensajes entre los nodos hasta que se cumple una condición determinada o se alcanza el número máximo de iteraciones especificado.

Raphtory ofrece un número importante de algoritmos implementados en los paquetes com.raphtory.algorithms.temporal y com.raphtory.algorithms.temporal [26]. Entre ellos se encuentran:

- Componentes conectados
- Algoritmos de propagación de etiquetas
- Ancestros
- MotifAlpha

Análisis temporal

Una de las principales características de Raphtory es que está diseñado para tener en cuenta grafos dinámicos, donde los elementos cambian en el tiempo. Esto permite analizar los cambios que ocurren en el grafo desde el punto de vista temporal.

Luego de diseñar un algoritmo (o utilizar alguno de los que están empaquetados en Raphtory), es necesario especificar el componente temporal de interés. Para esto, se parte de un objeto TemporalGraph, que contiene información de un grafo a lo largo del tiempo. Este objeto puede ser filtrado para obtener solamente el rango de tiempo de interés. A continuación, se puede crear una o varias colecciones de perspectivas sobre la línea de tiempo seleccionada previamente. A partir de este punto, es posible aplicar una secuencia de operaciones sobre el grafo que generan una tabla, y una secuencia de operaciones de tabla para obtener un resultado para la consulta. El último paso es guardar los resultados obtenidos [27].

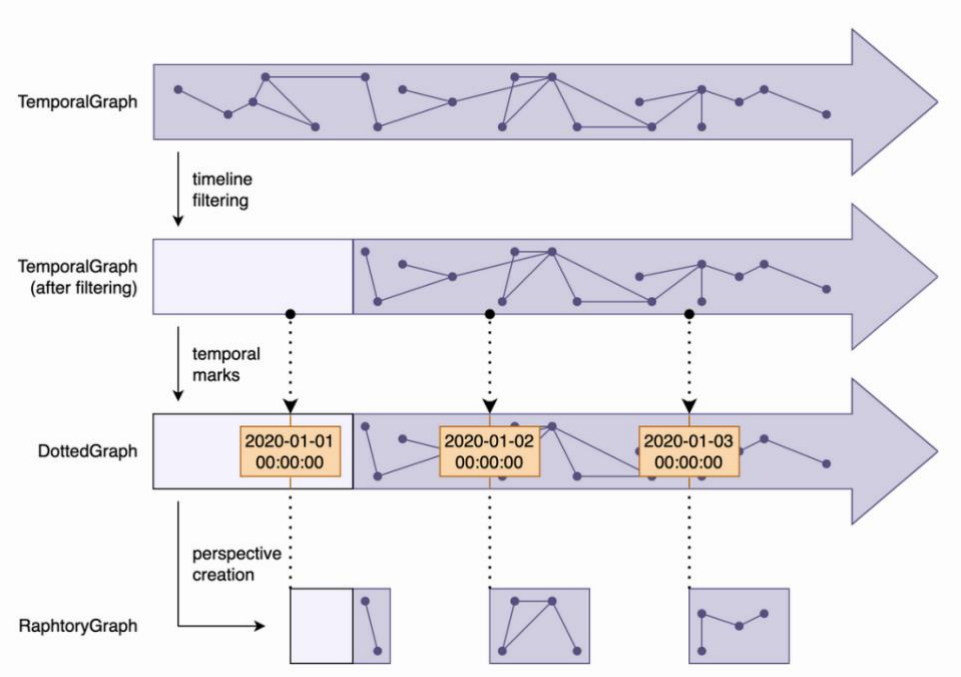

Figura 11. Etapas para crear perspectivas en un grafo temporal en Raphtory [27].

2.4 Selección de la herramienta de modelado de grafos

Como ya se ha explicado anteriormente, el mercado de NFTs está compuesto por Traders que comercian NFTs entre sí. El *dataset* presenta el componente temporal necesario para realizar un análisis de diferentes períodos de tiempo, teniendo en cuenta, además, el comportamiento del sistema en diferentes ventanas temporales. De las herramientas analizadas, Raphtory constituye la mejor opción para este caso por su compatibilidad para realizar análisis temporales de forma nativa.

En el siguiente capítulo se realiza una descripción de la configuración de Raphtory para el modelado y análisis del mercado de NFTs.

3. Modelado del mercado de NFTs

3.1 Descripción del *dataset*

En este trabajo se analiza un conjunto de datos provisto en [2]. Este *dataset* incluye solo transacciones que representan compras de NFT, cuyas propiedades cambian después de esa transacción. Se excluye del análisis cualquier transacción que represente la acuñación de NFTs u ofertas durante una subasta. Los datos de la cadena de bloques de Etherum para las colecciones SuperRare, Makersplace, Knownorigin, Cryptopunks y Asyncart fueron compartidos por NonFungible Corporation, una empresa que rastrea los datos históricos de ventas de NFT para generar valoraciones de estos Tokens No Fungibles. Se descargaron otros datos de la cadena de bloques de Ethereum de cuatro API de código abierto: CryptoKitties sales, Gods-Unchained, Decentraland y OpenSea. Se ha tenido en cuenta también la cadena de bloques WAX, mediante el seguimiento de transacciones en la API de Atomic [2].

3.1.1 Preprocesamiento del *dataset*

El *dataset* de transacciones de NFTs, disponible en [2], tiene un gran número de líneas en blanco que separan los datos de las transacciones. Esto representa un problema para el procesamiento de los registros y la construcción del grafo, debido a que Raphtory lee línea a línea y descompone las mismas para obtener los campos que conforman una transacción. Por otro lado, los campos en el *dataset* están separados por el carácter “,”, pero existen, además, campos de texto que incluyen el mismo carácter. Esto causa que la descomposición de las líneas, en los campos de interés, no tenga una longitud fija, lo cual puede provocar un problema al acceder a los campos mediante índices. La solución a este segundo problema se explica en la descripción del constructor del grafo (GraphBuilder).

Para solucionar el primero de estos problemas se ha utilizado el comando “awk 'BEGIN{RS="\n[[:space:]]*\n";ORS=" "}1' <<filename>>”, que realiza el siguiente procedimiento, asumiendo que la línea en blanco es la n:

- Borra línea n
- Anexa la línea n + 1 a la línea n – 1

De esta forma, se eliminan las líneas en blanco y se reestablece la estructura correcta de las transacciones con problemas.

3.2 Configuración de Raphtory

A continuación, se muestra un diagrama de componentes de Raphtory:

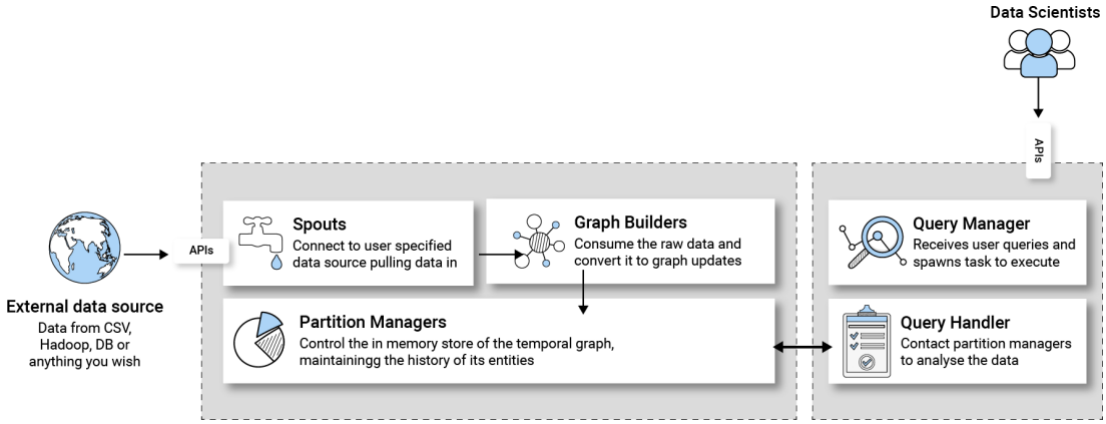

Figura 12. Componentes de ingesta de Raphtory [28].

Raphtory necesita 4 elementos para ejecutar un análisis temporal sobre un grafo: Spout, Graph Builder, Raphtory Graph y algoritmos.

Spout

Se utiliza para leer la fuente de datos, en este trabajo un archivo CSV. El spout está compuesto fundamentalmente por dos métodos: setupDataSource y generateData.

```
override def setupDataSource(): Unit = {
  fileQueue++= scala.io.Source.fromFile(directory + "/" + filename)
                  .getLines.drop(1)
}
```

setupDataSource extrae los datos y los inserta en una cola. En este caso se desecha la primera línea ya que corresponde a la descripción de los campos del *dataset*.

```
override def generateData(): Option[String] = {
    if(fileQueue isEmpty) {
        dataSourceComplete()
        None
    }
    else
        Some(fileQueue.dequeue())
}
```

generateData envía cada línea de datos desde la cola al constructor del grafo, hasta que la cola se vacía. Cada línea debe contener la información necesaria para crear un vértice, arista o ambos, así como sus propiedades.

GraphBuilder

Esta clase se encarga de obtener los campos para crear los vértices y aristas y de la definición de estos. GraphBuilder obtiene las líneas de la cola creada en el Spout. Estas líneas son procesadas y se extraen los campos necesarios para la construcción del grafo. A continuación, se muestra la configuración del GraphBuilder para la generación del grafo de Traders y NFTs:

```
val dtformat = new java.text.SimpleDateFormat("yyyy-MM-dd HH:m:ss")

override def parseTuple(tuple: String): Unit = {
  val fields = tuple
                .split(",(?=([^\\\"]*\\\"[^\\\"]*\\\")*[^\\\"]*$)")
                .map(_.trim).toList

  val nft = fields(1)
  val nftID = assignID(nft)
  val seller = fields(3)
  val sellerID = assignID(seller)
  val sellerUsername = fields(4)
  val buyer = fields(5)
  val buyerID = assignID(buyer)
  val buyerUsername = fields(6)
  val priceCrypto = fields(11)
  val crypto = fields(12)
  val priceUSD = fields(13) match {
      case "" => "0"
      case x => x
  }
  val category = fields.last
  val collection = fields(fields.length - 2)
  val timestamp  = dtformat.parse(fields(fields.length - 5)).getTime

  // Add NFT node
  addVertex(timestamp, nftID, Properties(
    StringProperty("seller", seller),
    LongProperty("sellerID", sellerID),
    LongProperty("buyerID", buyerID),
    ImmutableProperty("nft", nft),
    StringProperty("priceUSD", priceUSD),
    StringProperty("crypto", crypto),
    StringProperty("priceCrypto", priceCrypto),
    ImmutableProperty("collection", collection),
    ImmutableProperty("category", category),
  ), Type("NFT"))

  // Add Trader vertex for seller
  addVertex(timestamp, sellerID, Properties(
    StringProperty("nft", nft),
    ImmutableProperty("username", sellerUsername),
    ImmutableProperty("address", seller)
  ), Type("Trader"))
```

```
  // Add Trader vertex for buyer
  addVertex(timestamp, buyerID, Properties(
    StringProperty("nft", nft),
    ImmutableProperty("username", sellerUsername),
    ImmutableProperty("address", seller)
  ), Type("Trader"))

  // Add NFT Transaction edge
  addEdge(timestamp, buyerID, sellerID, Type("NFT Transaction"))

  // Add NFT Ownership edge for buyer
  addEdge(timestamp, buyerID, nftID, Type("NFT Ownership"))

  // Remove NFT Ownership edge for seller (if exists)
  deleteEdge(timestamp, sellerID, nftID)

}
```

El primer paso en el GraphBuilder consiste en separar cada en los diferentes campos. Como ya se ha explicado en la sección de preprocesamiento del *dataset*, hay campos que pueden incluir el carácter de separación, por lo que no se puede separar los campos empleando una coma, sino que hay que emplear la expresión regular mostrada. Se puede asignar los atributos "nft", "seller", "sellerUsername", "buyer", "buyerUsername", "priceCrypto", "crypto", "priceUSD" mediante los índices 1, 3, 4, 5, 6, 11, 12, 13, respectivamente.

Cada vértice debe incluir un identificador único cuando se crea, además las aristas utilizan estos para definir los nodos que unen. Raphtory brinda un método "assignID" que permite obtener estos identificadores, como se muestra en la sección de código anterior.

Para obtener el resto de los campos de interés se indexa desde el final de la lista de campos. De esta forma, el primer campo es la categoría, el segundo la colección del NFT y el quinto la estampa de tiempo de la transacción en cuestión.

Una vez extraídos todos los campos necesarios, se definen los vértices y aristas. Para cada transacción, se ejecutan los siguientes pasos:

1. Crear un vértice de tipo NFT.
    a. Propiedades: "seller", "sellerID", "buyerID", "nft", "priceUSD", "crypto", "priceCrypto", "collection", "category"
2. Crear dos vértices de tipo Trader (comprador y vendedor)
    a. Propiedades: "address", "username", "nft"
3. Crear una arista de Trader comprador a Trader vendedor (NFT Transaction)
4. Crear una arista de Trader comprador al NFT (NFT Ownership)

5. Eliminar la arista del Trader vendedor al NFT

Con este modelo, se define el grafo mediante la interacción de vértices de dos tipos: Trader y NFT.

RaphtoryGraph

Esta clase recibe los objetos Spout y GraphBuilder y devuelve un objeto que permite ejecutar análisis temporales sobre el grafo.

Algoritmo

Finalmente, es necesario un algoritmo para ejecutar un análisis sobre el grafo. Es posible realizar dos tipos de análisis temporales: PointQuery y RangeQuery.

En este trabajo se han definido un conjunto de algoritmos que se describen en la siguiente sección.

A continuación, se muestra unos datos de ejemplo de una versión simplificada del *dataset*:

Tabla 1. Ejemplo con campos reducidos de los datos de transacciones.

Tiempo	ID NFT	Vendedor	Comprador	Precio	Colección
t1	NFT 1	Trader B	Trader A	2	Cryptokittie
t2	NFT 2	Trader A	Trader C	4	Alien

A partir de los datos de la tabla anterior, Raphtory construye el siguiente grafo:

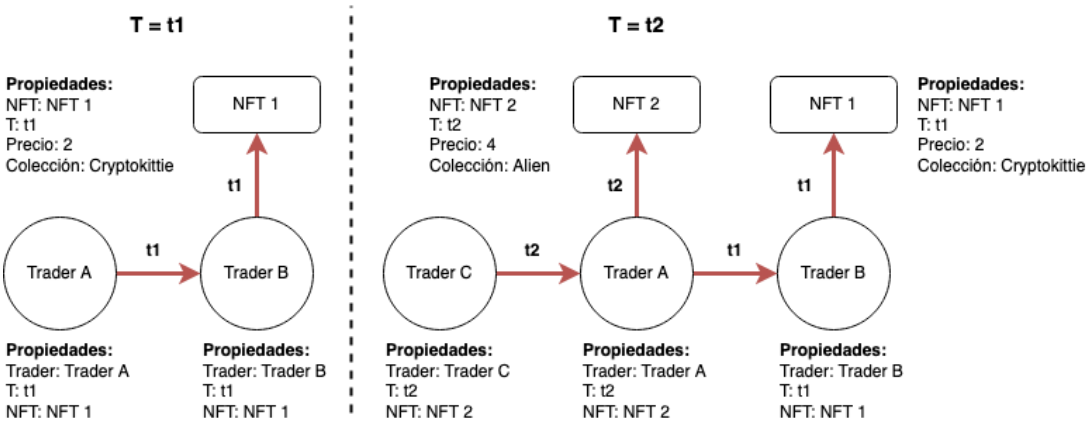

Figura 13. Grafo construido por Raphtory con los datos de la tabla 1.

3.3 Algoritmos

3.3.1 Grado de los Vértices

El grado de un vértice está determinado por el número de aristas conectados a dicho vértice. En un grafo dirigido, se puede diferenciar entre grado de entrada y grado de salida, de acuerdo al sentido de las aristas.

Con el objetivo de determinar el grado de los nodos del conjunto de datos, se define un algoritmo que permite obtener la métrica descrita, teniendo en cuenta solamente las aristas conectadas a nodos de tipo NFT, las conectadas solamente a nodos de tipo Trader o ambos. El grado define, por un lado, el número de transacciones de compra y venta, para las aristas de tipo "NFT Transaction", mientras que por otro establece el número de NFTs que posee un trader, para las aristas "NFT Ownership".

El algoritmo diseñado requiere dos pasos para calcular el grado, excepto cuando se quiere calcular para los dos tipos de aristas ("NFT Transaction" y "NFT Ownership") en conjunto, que solamente es necesario un paso. A continuación, se describen los tres casos.

Caso 1: Grado de los vértices teniendo en cuenta tanto las aristas de tipo "NFT Transaction" como las aristas de tipo "NFT Ownership".

```
vertex.setState("inDegree", vertex.getInNeighbours().size)
vertex.setState("outDegree", vertex.getOutNeighbours().size)
vertex.setState("totalDegree", vertex.getAllNeighbours().size)
```

En este caso, solamente es necesario un paso debido a que Raphtory brinda un método para acceder a los vecinos de entrada, salida y totales de cada vértice. Antes de obtener los datos finales se filtran los vértices de tipo NFT.

La figura 14 muestra un grafo de ejemplo en el que se establecen las relaciones entre 3 vértices de tipo Trader y 3 vértices de tipo NFT. En este ejemplo, en t = 0 "Trader B" poseía los NFTs 1 y 2, "Trader A" era dueño del "NFT 3", mientras que "Trader C" no tenía ninguno. En t = 1, "Trader A" compra a "Trader B" el "NFT 1", por lo que se establecen las aristas representadas con este instante de tiempo. Por último, en t = 2, "Trader C" compra a "Trader A" el "NFT 3". Las aristas en color rojo son las que tiene en cuenta el algoritmo para el cálculo del grado de los vértices Trader, y son, en este caso, todas.

Para el grafo siguiente, los grados quedarían de la siguiente forma, en el instante t = 2:

Tabla 2. Grados de los vértices de tipo Trader representados en la figura siguiente.

Vértice	Grado Entrada	Grado Salida	Grado Total
Trader A	1	2	3
Trader B	1	1	2

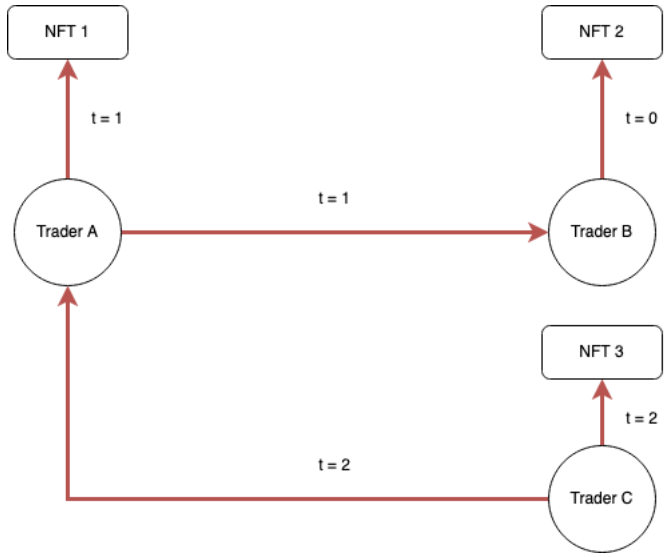

Figura 14. Grafo con 3 nodos de tipo Trader y 3 nodos de tipo NFT. Se tienen en cuenta todas las aristas para el cálculo de los grados.

Caso 2: Grado de los vértices teniendo en cuenta solamente las aristas de tipo "NFT Transaction"

- Paso 1

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
    vertex.messageAllIngoingNeighbors(0)
    vertex.messageAllOutgoingNeighbors(1)
}
```

En este paso, los vértices de tipo Trader envían el valor 0 a todos los vecinos de entrada, mientras que el valor 1 es enviado a todos los vecinos de salida. Con el envío de estos mensajes se consigue que los nodos puedan calcular el número de vecinos de entrada y salida, siguiendo la lógica descrita en el paso 2.

En la figura siguiente se puede observar el mismo grafo que en la figura anterior. Sin embargo, en este caso solamente se tienen en cuenta las aristas que enlazan vértices de tipo Trader (en rojo). En el paso 1, el nodo "Trader A" envía un 0 al nodo "Trader C" y

un 1 al nodo "Trader B". Por otra parte, "Trader B" envía un 0 a "Trader A" y, por último, "Trader C" envía un 1 a "Trader A".

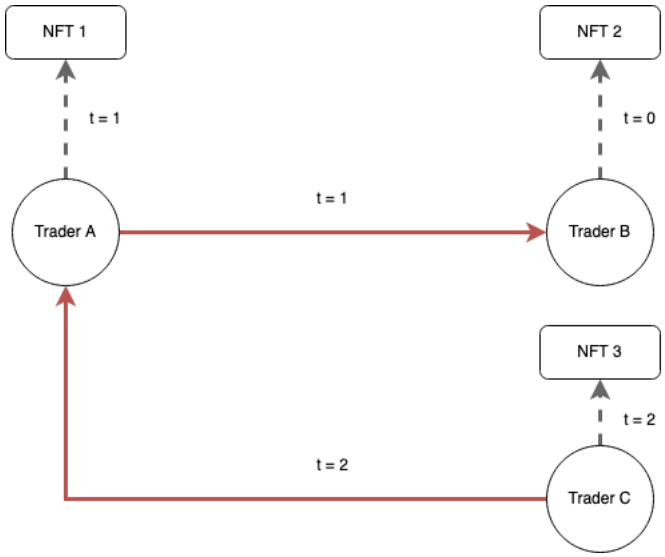

Figura 15. Grafo con 3 nodos de tipo Trader y 3 nodos de tipo NFT. Se tienen en cuenta solamente las aristas de tipo NFT Transaction, en color rojo.

- Paso 2

```
val queueMap: Map[Int, Int] = vertex.messageQueue[Int]
          .groupBy(mess => mess).map(tup => (tup._1, tup._2.length))

val inDegree = queueMap.getOrElse(1, 0)
val outDegree = queueMap.getOrElse(0, 0)

vertex.setState("inDegree", inDegree)
vertex.setState("outDegree", outDegree)
vertex.setState("totalDegree", inDegree + outDegree)
```

En esta etapa, se accede a la cola de mensajes de cada vértice para procesar los valores recibidos en los mensajes del paso anterior. Inicialmente, se crea un mapa o diccionario cuya clave puede ser 0 ó 1, y el valor es el número total de 0 ó 1 recibidos, respectivamente. En el paso 1, los vértices envían un 0 a los vecinos de entrada, esto quiere decir que cuando un vértice recibe un 0 en el segundo paso, agrega este valor a su grado de salida. Cuando se envía un 1 en el paso inicial, el vértice que recibe este valor lo agrega al total del grado de entrada. Por último, se guarda el grado de entrada, salida y total en campos de estado de cada vértice.

Siguiendo el ejemplo de la figura 15, cuando "Trader A" recibe el mensaje con un 0 proveniente de "Trader B", se establece que "Trader A" es un vecino de entrada para un nodo (en este caso "Trader B"), por lo que "Trader A" incrementa su grado de salida en

una unidad. Al revisar su cola de mensajes y encontrar un 1 (el enviado por "Trader C"), se determina que "Trader A" es un vecino de salida, por lo que se incrementa su grado de entrada también en una unidad. De forma similar al caso de "Trader A", "Trader B" recibe un 1 e incrementa el grado de entrada y "Trader C" recibe un 0 e incrementa su grado de salida. En la tabla siguiente se muestra la distribución de los grados para los 3 nodos.

Tabla 3. Grados de los vértices de tipo Trader representados en la figura 15. Se tienen en cuenta solamente las aristas de tipo NFT Transaction, en t = 2.

Vértice	Grado Entrada	Grado Salida	Grado Total
Trader A	1	1	2
Trader B	1	0	1
Trader C	0	1	1

Caso 3: Grado de los vértices teniendo en cuenta solamente las aristas de tipo "NFT Ownership"

- Paso 1

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") == "nft")
    vertex.messageAllIngoingNeighbors(0)
```

El caso 3 se puede considerar un subtipo del caso 2 debido a que el procedimiento casi es el mismo. La única diferencia consiste en que las aristas entre los nodos de tipo Trader y los nodos de tipo NFT siempre están dirigidos del primero al segundo. Como consecuencia de esto, y de que no se tienen en cuenta las aristas de tipo NFT Transaction, solo los nodos NFT envían un mensaje, y siempre a los vecinos de entrada, los respectivos traders que son sus propietarios.

En la imagen siguiente se define nuevamente el grafo de ejemplo y se muestran en rojo las aristas que se tienen en cuenta para este caso, las de tipo NFT Ownership. Como solamente existe una dirección de las aristas de este tipo, el mensaje a enviar es indiferente, se ha utilizado un 0. Cada uno de los nodos NFT: "NFT 1", "NFT 2" y "NFT 3" envía un 0 a "Trader 1", "Trader 2" y "Trader 3", respectivamente.

- Paso 2

```
val inDegree = 0
val outDegree = vertex.messageQueue[Int].size
vertex.setState("inDegree", inDegree)
vertex.setState("outDegree", outDegree)
vertex.setState("totalDegree", inDegree + outDegree)
```

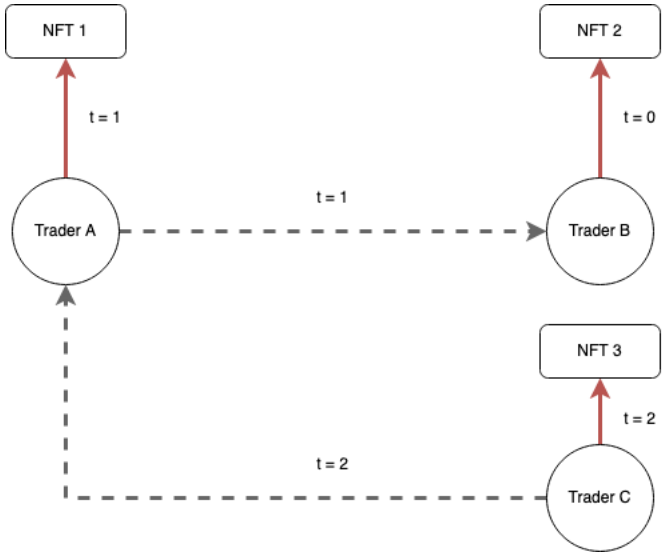

Figura 16. Grafo con 3 nodos de tipo Trader y 3 nodos de tipo NFT. Se tienen en cuenta solamente las aristas de tipo NFT Ownership, en color rojo.

En el segundo paso, se establece el grado de entrada a 0 y, a continuación, como los traders solamente reciben 0, el valor del grado de salida es igual al número total de mensajes recibidos. Por último, se guarda en campos de estado el valor de los grados.

En el ejemplo de la figura 16, cada trader recibe un 0 de su NFT correspondiente. El grado de salida de cada vértice, teniendo en cuenta solamente las aristas de tipo NFT Ownership, es una medida de la cantidad de NFT que posee cada trader.

Tabla 4. Grados de los vértices de tipo Trader representados en la figura 16. Se tienen en cuenta solamente las aristas de tipo NFT Ownership.

Vértice	Grado Entrada	Grado Salida	Grado Total
Trader A	0	1	1
Trader B	0	1	1
Trader C	0	1	1

Selección y Filtrado

Luego de terminar el análisis de todos los casos descritos, se realiza una selección y filtrado de los nodos de tipo Trader, dejando fuera los nodos de tipo NFT.

Selección

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
    Row(vertex.getPropertyOrElse("address", vertex.ID()),
        vertex.getStateOrElse("inDegree", 0),
```

```
        vertex.getStateOrElse("outDegree", 0),
        vertex.getStateOrElse("totalDegree", 0))
} else Row()
```

Se crea una fila (Row) con la propiedad "address", "inDegree", "outDegree" y "totalDegree" para cada vértice de tipo Trader. Para los vértices de tipo NFT se crea una fila en blanco que es filtrada en el siguiente paso.

Filtrado

```
row => row.getValues().length > 1
```

La salida obtenida para este algoritmo contendría las siguientes filas, para el ejemplo descrito, en el instante t = 2:

Caso 1:

A, 1, 2, 3

B, 1, 1, 2

C, 0, 2, 2

Caso 2:

A, 1, 1, 2

B, 1, 0, 1

C, 0, 1, 1

Caso 3:

A, 0, 1, 1

B, 0, 1, 1

C, 0, 1, 1

3.3.2 Peso de las Aristas

Cada arista tiene un peso correspondiente al número total de NFTs que un comprador ha comprado a un vendedor. Con el objetivo de determinar esta métrica, al crear el grafo mediante la clase NFTBuilder se ha añadido el campo "seller" a cada NFT, el cual especifica el vendedor de la NFT en cada transacción. El campo "seller" cada vez que Raphtory recibe una actualización del grafo, es modificado con el último vendedor. Raphtory mantiene un registro histórico de los valores asignados a esta propiedad.

El algoritmo implementado consta de dos pasos:

- Paso 1

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") == "nft")
{
    val seller = vertex.getPropertyOrElse("seller","unknown_seller")
```

```
    vertex.messageAllIngoingNeighbors(seller)
}
```

En este paso, los vértices de tipo NFT envían su propiedad "seller" a los nodos Trader que son sus propietarios.

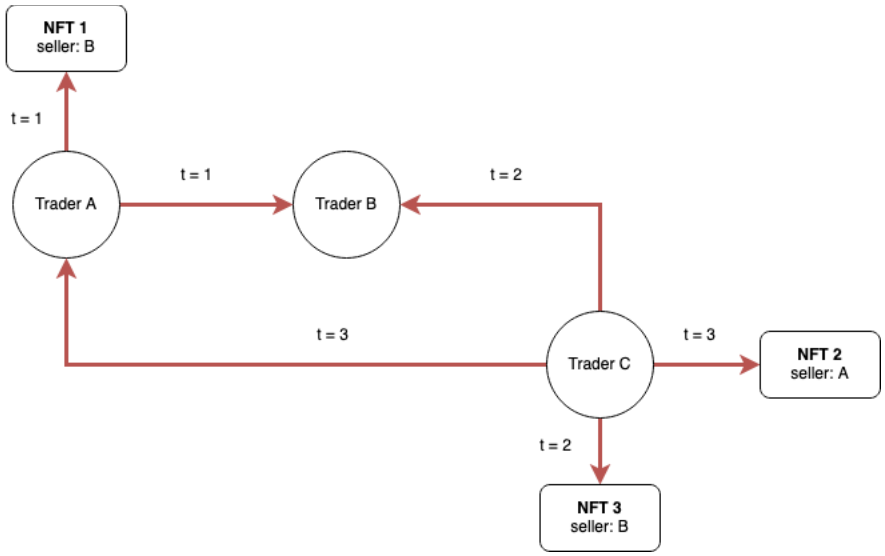

Figura 17. Grafo con 3 nodos de tipo Trader y 3 nodos de tipo NFT. Se muestra el vendedor de cada NFT.

De acuerdo al ejemplo de la figura anterior, "NFT 1" envía a "Trader A" su vendedor, que en este caso es "Trader B". De forma similar, "NFT 2" envía a "Trader C" su vendedor, "Trader A", mientras que "NFT 3" envía "Trader B" a "Trader C".

- Paso 2

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
    val queueList: List[(String, Int)] = vertex.messageQueue[String]
        .groupBy(el => el).map(e => (e._1, e._2.length)).toList

    Row(vertex.getPropertyOrElse("address", vertex.ID()), queueList)
} else Row()
```

En el paso 2 se procesa los mensajes enviados por los nodos NFT a los nodos Trader. La cola de mensajes está compuesta por una lista de los vendedores de todos los NFTs que posee cada vértice de tipo Trader. Se transforman estos mensajes para agruparlos por vendedor, por lo que se obtiene una lista compuesta por tuplas (Vendedor, Número de NFTs vendidas por vendedor). Se conforma una fila con el valor de la propiedad "address" y la lista de tuplas. En este paso también se realiza la selección de los vértices de tipo Trader, de igual forma que en el caso del algoritmo de cálculo del grado.

En cuanto al grafo de ejemplo de la figura 17, luego de aplicar la transformación a la cola de mensajes, "Trader A" obtiene la lista List("Trader B", 1), mientras que "Trader C" obtiene List(("Trader A", 1), ("Trader B", 1)), correspondiente a "NFT 2" y "NFT 3", respectivamente.

Explosión de filas

Para los casos como los de "Trader C", que tienen dos o más aristas de distintos vendedores, es necesario utilizar una "explosión" de las filas obtenidas en el paso 2. Por último, se realiza un filtrado igual al descrito en el algoritmo anterior.

```
explode(row =>
          if (row.getValues().length > 1) {
              row.get(1).asInstanceOf[List[(String, Int)]]
                  .map(expl => Row(row(0), expl._1, expl._2))
          }
          else List(Row())
      )
```

Con este procedimiento se consigue descomponer una fila compuesta por varios vendedores en una fila por vendedor. Se puede entender la aplicación de este método al observar a continuación las filas de salida para "Trader C".

El archivo de salida, con los pesos para las aristas, contendría las filas:
A, B, 1
C, A, 1
C, B, 1

3.3.3 Poder de los Traders

El poder de los nodos de tipo Trader se define como el número total de compras y ventas que ha hecho cada trader. No basta con saber la cantidad de NFTs que posee un determinado trader en un momento dado, sino que es necesaria una visión histórica de sus transacciones. Para obtener esta visión, se utilizan los registros históricos de modificaciones de los campos "seller", "sellerID", "buyer" y "buyerID", que mantiene Raphtory y que actualiza con cada cambio en el grafo. Este algoritmo también se utiliza para obtener información sobre la cantidad de dinero que un trader ha ganado o gastado en el comercio de NFTs, así como sus días de actividad y el detalle de colección/categoría de cada NFT con el que ha interactuado.

Al igual que en los dos casos anteriores, este algoritmo requiere dos pasos:

- Paso 1

```
case class TraderInfo(toNode: String, nTransaction: Int, priceUSD:
                      BigDecimal)
case class CollectionCategory(toNode: String, collection: String,
                              category: String)

if (vertex.getPropertyOrElse("address", otherwise = "nft") == "nft")
{
  val buyerList: List[Long] = vertex.getPropertyHistory("buyerID")
                .get.asInstanceOf[List[(Long, Long)]]
                .map {tuple => tuple._2}

  val sellerList: List[Long] = vertex.getPropertyHistory("sellerID")
                .get.asInstanceOf[List[(Long, Long)]]
                .map {tuple => tuple._2}

  val priceUSD: List[BigDecimal] = vertex
                                  .getPropertyHistory("priceUSD")
                                  .get
                                  .asInstanceOf[List[(Long, String)]]
                                  .map {tuple => tuple._2}
                                  .map(x => BigDecimal(x))

  val collection = vertex.getPropertyOrElse("collection", "unknown")
  val category = vertex.getPropertyOrElse("category", "unknown")

  (buyerList zip priceUSD).foreach { tuple =>
     vertex.messageNeighbour(tuple._1, TraderInfo("buyer", 1,
                                                  tuple._2))
     vertex.messageNeighbour(tuple._1, CollectionCategory("buyer",
                                              collection, category))
   }

   (sellerList zip priceUSD).foreach { tuple =>
      vertex.messageNeighbour(tuple._1, TraderInfo("seller", 1,
                                                   tuple._2))
      vertex.messageNeighbour(tuple._1, CollectionCategory("seller",
                                              collection, category))
   }
```

```
}

if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
  vertex.setState("purchased", 0)
  vertex.setState("sold", 0)
  vertex.setState("strength", 0)
  vertex.setState("amountSpent", 0)
  vertex.setState("amountGained", 0)
  vertex.setState("boughtNFTInfo", "")
  vertex.setState("soldNFTInfo", "")

  val activityDays = vertex.getPropertyHistory("nft").get
              .asInstanceOf[List[(Long, String)]]
              .map(tuple => tuple._1)
              .map(datetime => TimeUnit.DAYS.convert(datetime,
                  TimeUnit.MILLISECONDS)).distinct.length

  vertex.setState("activityDays", activityDays)
}
```

Inicialmente, para cada nodo de tipo NFT, se crean las listas de compradores y vendedores que ha tenido dicho nodo. Para esto se utiliza el método getPropertyHistory, disponible para los vértices, para obtener una lista con los cambios históricos de los campos "buyerID" y "sellerID". A continuación, se crea una variable que almacena los registros de todos los precios de venta que ha tenido el activo, y dos variables adicionales que guardan la colección y categoría a la que pertenece el NFT. Posteriormente, se envían dos mensajes a cada comprador y vendedor con el precio compra/venta, la colección y categoría de cada NFT que han comprado y vendido, respectivamente. Esta información va codificada en las *case classes* mostradas al inicio del fragmento de código, TraderInfo y CollectionCategory.

En el caso de los nodos de tipo Trader, simplemente se inicializan las propiedades de interés a 0 y se obtiene el número de días de actividad haciendo uso de la propiedad "nft" añadida a cada nodo Trader en el GraphBuilder.

En este estudio, los datos principales usados de la salida de este algoritmo son el poder de los traders y los días de actividad. Debido a esto, el ejemplo mostrado a continuación solamente tiene en cuenta dichos parámetros.

En la imagen siguiente, se muestra la disposición de un grafo con transacciones en dos instantes de tiempo diferentes, t = 1 y t = 2. En el instante de tiempo t = 1, "Trader

A" compra a "Trader B" los NFTs 1, 2 y 3 (vértices y aristas en azul). En un instante de tiempo t = 2, "Trader C" compra los NFTs 2 y 3 a "Trader A" (nodos y aristas en amarillo).

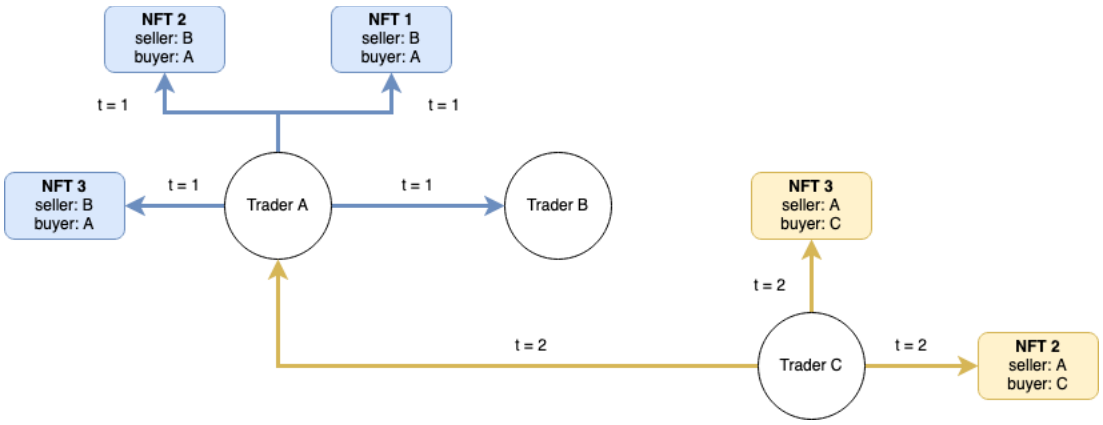

Figura 18. Grafo con 3 nodos de tipo Trader y 3 nodos de tipo NFT en dos instantes de tiempo diferente. Los vértices y aristas coloreados en azul identifican transacciones ocurridas en tiempo t = m, mientras que los coloreados en amarillo son transacciones que ocurren en t = n, donde m sucede después que n.

Las listas "buyerList" y "sellerList" de los nodos NFT del grafo serían las siguientes:

NFT 1

buyerList: List("Trader A")

sellerList: List("Trader B")

NFT 2

buyerList: List("Trader A", "Trader C")

sellerList: List("Trader B", "Trader A")

NFT 3

buyerList: List("Trader A", "Trader C")

sellerList: List("Trader B", "Trader A")

Luego de obtener las listas de vendedores y compradores, los tres NFTs del grafo envían a todos los nodos de sus dos diccionarios correspondientes un mensaje con el precio de venta asociado con cada uno de los traders. Esto se realiza mediante la función *zip*, y funciona correctamente porque se mantiene el orden tanto en las listas de

compra/venta como en el precio asociado a cada operación. Desde el punto de vista de los días de actividad, suponiendo que t=1 y t=2 no ocurren en el mismo día, "Trader B" y "Trader C" tienen un solo día de actividad, mientras que "Trader A" tiene 2. Si ambos instantes temporales pertenecen a un día, los 3 traders tendrán 1 día de actividad.

- Paso 2

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
  val (traderInfo: List[TraderInfo],
   collCatInfo: List[CollectionCategory]) = vertex.messageQueue[Any]
            .partition {
              case _: TraderInfo => true
              case _ => false
            }

  if (traderInfo.nonEmpty) {
    val queueMapTraderInfo = traderInfo
              .groupBy(traderInfo => traderInfo.toNode)
              .map(tup => (tup._1,
                tup._2.foldLeft(0)(_ + _.nTransaction),
                tup._2.foldLeft(BigDecimal(0))(_ + _.priceUSD))
              ).iterator.map(c => c._1 -> (c._2, c._3)).toMap

    val amountSpent = queueMapTraderInfo.getOrElse("buyer", 0)
            match {
              case 0 => 0
              case x => x.asInstanceOf[(Int, BigDecimal)]._2
            }

    val amountGained = queueMapTraderInfo.getOrElse("seller", 0)
            match {
              case 0 => 0
              case x => x.asInstanceOf[(Int, BigDecimal)]._2
            }

    val purchased = queueMapTraderInfo.getOrElse("buyer", 0)
            match {
              case 0 => 0
              case x => x.asInstanceOf[(Int, Double)]._1
            }

    val sold = queueMapTraderInfo.getOrElse("seller", 0)
```

```
            match {
              case 0 => 0
              case x => x.asInstanceOf[(Int, Double)]._1
            }

  vertex.setState("purchased", purchased)
  vertex.setState("amountSpent", amountSpent)
  vertex.setState("sold", sold)
  vertex.setState("amountGained", amountGained)
  vertex.setState("strength", purchased + sold)
}

if (collCatInfo.nonEmpty) {
  val queueMapCollCategory = collCatInfo
            .groupBy(collcat => collcat.toNode)
            .map(tuple1 => (tuple1._1,
              tuple1._2
                .groupBy(tuple2 => (tuple2.collection,
                                    tuple2.category))
                .map(tuple3 => (tuple3._1, tuple3._2.length))
            ))
            .map(tuple4 => (tuple4._1, tuple4._2
              .map(tuple5 => (tuple5._1._1, tuple5._1._2,
                              tuple5._2))))

  val boughtNFTInfo = queueMapCollCategory.getOrElse("buyer", "")
            match {
              case "" => ""
              case x:  List[(String, String, Int)] =>
                x.mkString(";")
                  .replace("_", "")
                  .replace(",", "_")
            }

  val soldNFTInfo = queueMapCollCategory.getOrElse("seller", "")
            match {
              case "" => ""
              case x:  List[(String, String, Int)] =>
                x.mkString(";")
                  .replace("_", "")
                  .replace(",", "_")
```

```
            }

    vertex.setState("boughtNFTInfo", boughtNFTInfo)
    vertex.setState("soldNFTInfo", soldNFTInfo) } }
```

En el segundo paso, lo primero que se hace es analizar la cola de mensajes de los nodos de tipo Trader para llevar a cabo el proceso necesario, de acuerdo a si el mensaje recibido contiene un objeto de tipo TraderInfo o CollectionCategory. De cada NFT se recibe siempre 2 mensajes, uno para cada objeto.

Para cada nodo Trader, se agrupan en dos listas los objetos recibidos, según su tipo. La lista de objetos TraderInfo contiene la información relativa al número de NFTs vendidos y comprados (poder), así como el total de dinero invertido. Por otro lado, la lista de objetos CollectionCategory agrupa por colección y categoría el número de NFTs que el nodo Trader ha comprado y vendido.

Siguiendo con el ejemplo de la figura anterior, "Trader A" recibe como comprador los mensajes de los 3 NFTs, y como vendedor los mensajes de "NFT 2" y "NFT 3". En el caso de "Trader B", recibe mensajes de los 3 NFTs como vendedor. Por último, "Trader C", recibe los mensajes como comprador de "NFT 2" y "NFT 3".

Al analizar los mensajes que contienen objetos TraderInfo, es calculado el poder, que es igual a "purchased+sold", según lo mostrado en el fragmento de código anterior. De acuerdo al ejemplo de la figura 18, el poder de "Trader A" es 5, el de "Trader B" 3 y el de "Trader C" 2.

Como se ha mencionado anteriormente, no se muestra el detalle de las colecciones y categorías, pero se sigue un procedimiento similar al de la obtención del poder y los precios de venta.

Selección y Filtrado

Al completar la ejecución del algoritmo, se seleccionan solamente los nodos de tipo Trader y se definen las filas formadas por las propiedades "address", "amountSpent", "amountGained", "purchased", "sold", "strength", "activityDays", "boughtNFTInfo" y "soldNFTInfo". El filtrado realizado en este caso es igual al descrito en el algoritmo anterior.

Selección

```
if (vertex.getPropertyOrElse("address", otherwise = "nft") != "nft")
{
  Row(vertex.getPropertyOrElse("address", vertex.ID()),
```

```
      vertex.getStateOrElse("amountSpent", -1),
      vertex.getStateOrElse("amountGained", -1),
      vertex.getStateOrElse("purchased", -1),
      vertex.getStateOrElse("sold", -1),
      vertex.getStateOrElse("strength", -1),
      vertex.getStateOrElse("activityDays", -1),
      vertex.getStateOrElse("boughtNFTInfo", ""),
      vertex.getStateOrElse("soldNFTInfo", "")
     )
} else Row()
```

Luego del filtrado de los vértices de interés, el archivo de salida para el ejemplo descrito contendría las siguientes filas:

- A, 12, 5, 3, 2, 5, 2, (Terra_Utility_1);(Alien_Games_1);(Ultra_Art_1), (Alien_Games_1) ,(Ultra_Art_1)
- B, 1, 23, 0, 3, 3, 1, , (Terra_Utility_1);(Alien_Games_1);(Ultra_Art_1)
- C, 7, 6, 2, 0, 2, 1, (Alien_Games_1);(Ultra_Art_1),

Como no se ha dado información de precios de venta ni colección o categoría en este ejemplo, se asumen estos valores. Para que haya correspondencia con los datos del ejemplo, "NFT 1" se asigna a la colección Terra, categoría Utility, "NFT 2" a la colección Alien, categoría Games y "NFT 3" a la colección Ultra, categoría Arte. Este análisis correspondería a una ventana temporal que incluyera los dos instantes de tiempo.

En el capítulo siguiente se presentan los resultados obtenidos tras realizar el análisis sobre el *dataset* y ejecutar los algoritmos presentados en esta sección.

4. Evaluación de resultados

En este capítulo se presenta inicialmente una caracterización estadística del mercado de NFTs. A continuación, se discuten los resultados obtenidos luego de ejecutar los algoritmos presentados en la sección 3.3 en Raphtory.

4.1 Descripción general de los datos

El conjunto de datos utilizado contiene alrededor de 6,1 millones de transacciones de 4,3 millones de NFTs. Se registran transacciones llevadas a cabo entre el 23 de noviembre de 2017 y el 27 de abril de 2021, de NFTs obtenidos principalmente de las cadenas de bloques Ethereum y WAX. Estas transacciones las realizan un total de 532728 traders diferentes.

4.2 Caracterización de las colecciones de NFTs

Los NFTs que comparten una o varias características se agrupan en colecciones. En el conjunto de datos se identifican transacciones asociadas a 4621 colecciones diferentes. Estas colecciones pueden ser a su vez agrupadas en 6 categorías: Arte, Coleccionables, Juegos, Metaverso, Otros y Utilidades. En la siguiente figura se muestran las 5 colecciones más grandes en términos de número de NFTs diferentes, por categoría:

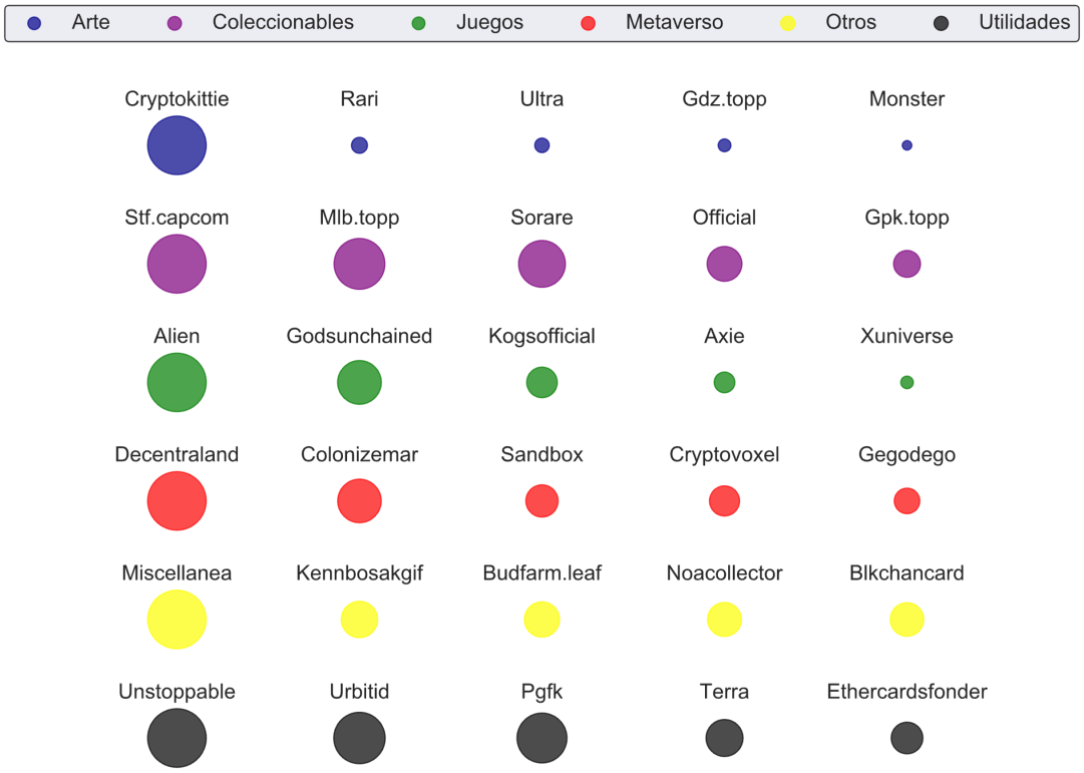

Figura 19. Cinco colecciones más grandes de acuerdo al número de NFTs que contienen. El tamaño de los círculos es proporcional al número de activos en cada colección.

La figura anterior muestra que en la categoría Arte hay bastante disparidad en el número de activos en las colecciones mostradas, conteniendo la colección Cryptokittie el 82,6 % del total de activos de las 5 colecciones.

Esta imagen muestra algunas diferencias con respecto al análisis llevado a cabo en [2]. En primer lugar, la proporción de las colecciones de la categoría Arte varía significativamente, mientras que las colecciones con menor número de NFTs de dicha categoría difieren, siendo Monster en este estudio y SuperRare en [2]. En la categoría Metaverso, las colecciones Sandbox y Cryptovoxel intercambian posiciones entre los dos análisis. Estas diferencias se pueden deber, sin tener seguridad, a que el *dataset* publicado y utilizado en este trabajo no es exactamente igual al empleado para generar los resultados obtenidos en [2].

4.3 Distribución de NFTs y transacciones por categoría

A continuación, se muestra la distribución de NFTs y transacciones por categoría:

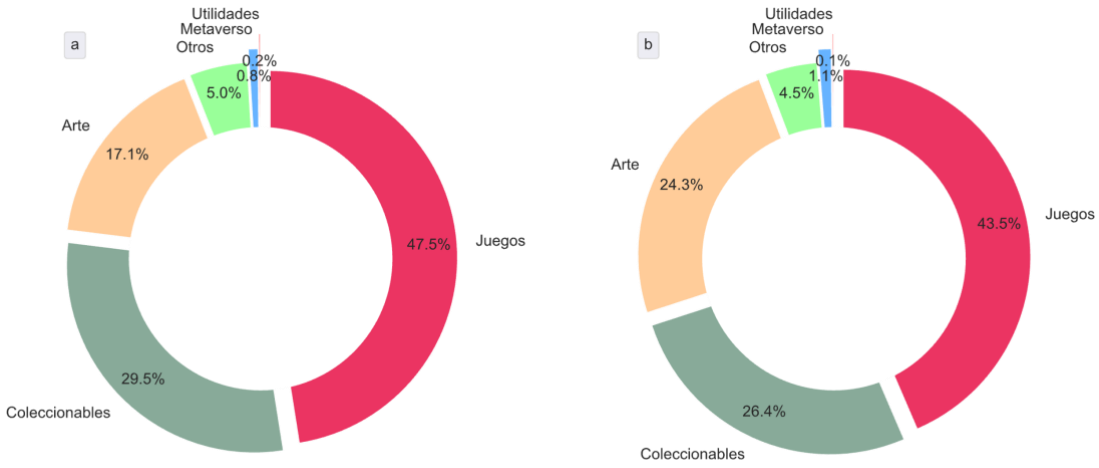

Figura 20. a) Distribución de NFTs por categoría. b) Distribución de transacciones por categoría.

En ambos casos de la figura anterior la proporción es igual: tanto el número de NFTs como la cantidad de transacciones están encabezados por la categoría Juegos, seguida por Coleccionables y en último lugar Utilidades.

4.4 Distribución de NFTs y transacciones por criptomoneda

En el conjunto de datos hay transacciones asociadas a 160 criptomonedas diferentes. Sin embargo, tanto para el número de NFTs como para las transacciones, la suma de las 157 criptomonedas con menor representación equivale al 7% y 10%, respectivamente, de la tercera criptomoneda más frecuente. Debido a esto, se agrupan estas 157 criptomonedas en la categoría Otras.

Cuando se analiza la distribución de NFTs y transacciones, de acuerdo al tipo de criptomoneda, se obtiene el siguiente gráfico:

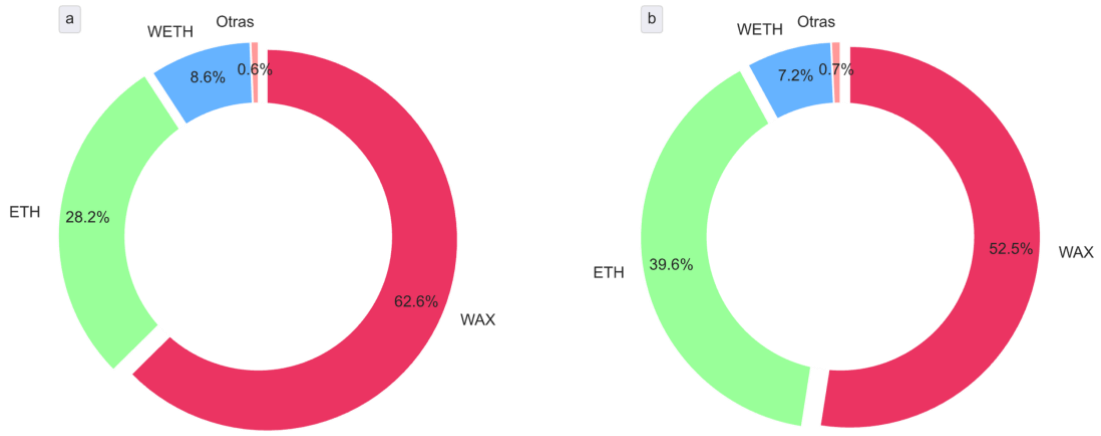

Figura 21. a) Distribución de NFTs por criptomoneda. b) Distribución de transacciones por criptomoneda.

La figura anterior muestra que el 90 % de los activos pertenecen a las criptomonedas WAX y ETH. De forma similar, el 92 % de las transacciones corresponden a NFTs de WAX y ETH.

4.5 Volumen diario en USD

Con el objetivo de analizar temporalmente las dimensiones del mercado de NFTs de acuerdo a la cantidad de dinero intercambiado, se obtiene el volumen diario intercambiado en USD:

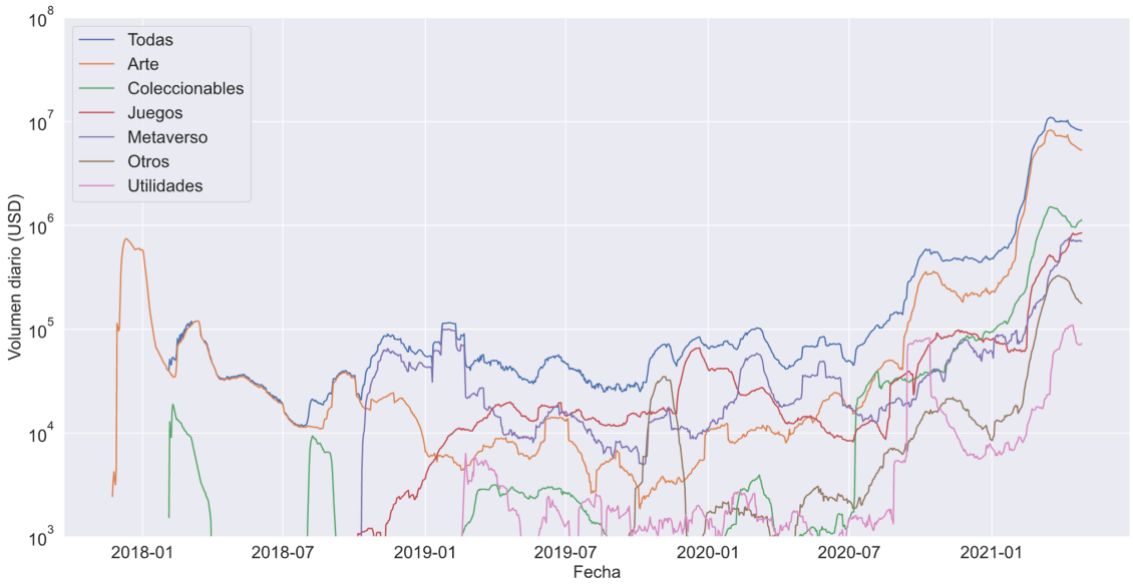

Figura 22. Volumen diario en USD calculado mediante una ventana rodante de 30 días. No se muestran los valores por debajo de 1000 USD.

De acuerdo al gráfico anterior, hasta comienzos de 2018 solamente la categoría Arte supera los 1000 USD diarios. En la semana del 3 al 10 de diciembre de 2017 se comercia una media de 1,54 millones de USD en NFTs de esta categoría. No fue hasta octubre de

2018 donde el volumen asociado a la categoría Arte perdió la primera posición, superado por NFTs de la categoría Metaverso, con una media de 75351 USD en la primera semana luego del cambio de posición. En enero de 2019 el volumen asociado a Juegos también supera al de la categoría Arte y en marzo pasa a ser la categoría con mayor volumen. En agosto de 2020 ocurre un incremento muy fuerte del volumen de todas las categorías, siendo el volumen medio entre septiembre y diciembre de 2020 de 476017 USD. A finales de enero de 2021 surge un pico aún mayor que el anterior, el volumen medio del 1 enero al 27 de abril de 2021 es de 6,4 millones de USD aproximadamente.

4.6 Número de transacciones diarias

La siguiente figura muestra el número de transacciones diarias:

Figura 23. Número de transacciones diarias calculado mediante una ventana rodante de 30 días.

Al comparar las figuras 22 y 23, existe una correspondencia entre los picos de volumen y transacciones. Este comportamiento es particularmente visible en los ascensos de finales de 2017 y a partir de mitad de septiembre de 2020 en adelante. Sin embargo, el pico de transacciones de finales de 2019 no tiene una correspondencia tan marcada, esto se debe a que la media hallada del volumen en la ventana de 30 días suavizó significativamente este ascenso. El número diario máximo de transacciones de los años 2017 a 2021 es 18048, 9798, 26185, 14437 y 154279, respectivamente.

4.7 Contribución al total de transacciones por categoría

Seguidamente, se muestra una serie temporal de la contribución al total de transacciones por cada categoría:

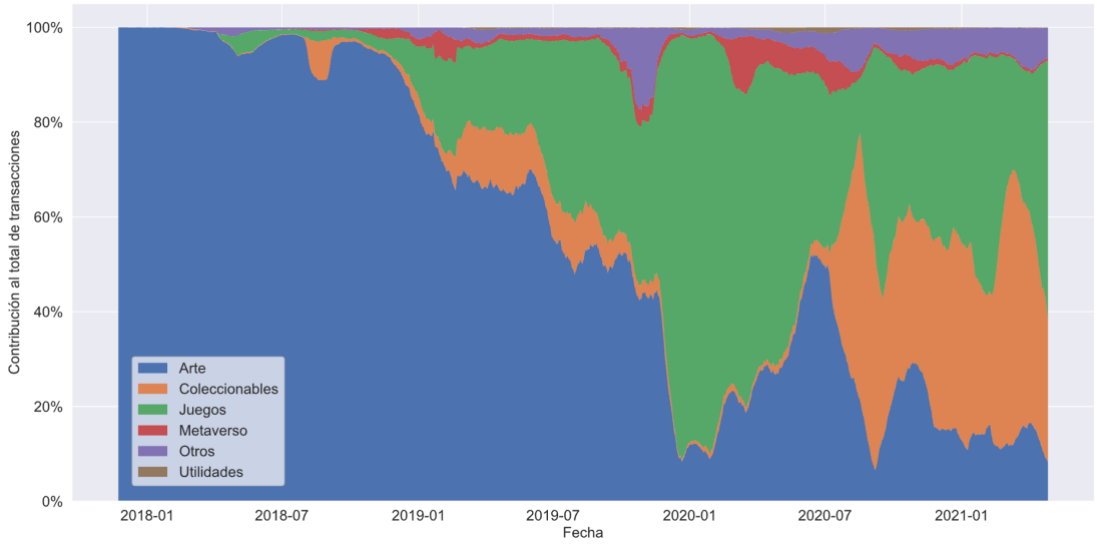

Figura 24. Contribución diaria al total de transacciones, por categoría. Calculado con una ventana rodante de 30 días.

La figura anterior muestra que las transacciones asociadas a activos de la categoría Arte fueron predominantes hasta el fin de la primera mitad de 2019 aproximadamente. Desde finales de 2018 el comercio de NFTs pertenecientes a la categoría Juegos fue en incremento, hasta superar a las transacciones de tipo Arte. En la mitad de 2020 se dispara el comercio de NFTs de tipo Coleccionables, siendo predominantes junto a las categorías Arte y Juegos. Las transacciones de NFTs de tipo Metaverso, Otros y Utilidades son las de menor contribución.

4.8 Contribución al volumen intercambiado en USD

Se lleva a cabo un análisis de la contribución al volumen total de dólares americanos intercambiados diariamente en el mercado de NFTs. La figura siguiente muestra que, hasta el tercer trimestre de 2018, la categoría Arte es en la que mayor cantidad de dólares americanos se intercambian. A partir de este momento, ocurre un aumento de volumen de la categoría Metaverso. Desde el segundo trimestre de 2019, el volumen está dominado por la categoría Juegos, hasta que en la segunda mitad de 2020 la categoría Arte retoma la primera posición.

Si se analizan en conjunto las figuras 24 y 25, se puede notar que, a partir de 2020, el número de transacciones asociadas a la categoría Arte no es predominante mientras que el mayor porcentaje en USD intercambiado sí corresponde a esta categoría. Este comportamiento permite concluir que los NFTs de la categoría Arte tienen un valor de

venta medio mayor a los del resto de categorías. Este mismo comportamiento ocurre en el pico de finales de 2018 para NFTs de la categoría Metaverso.

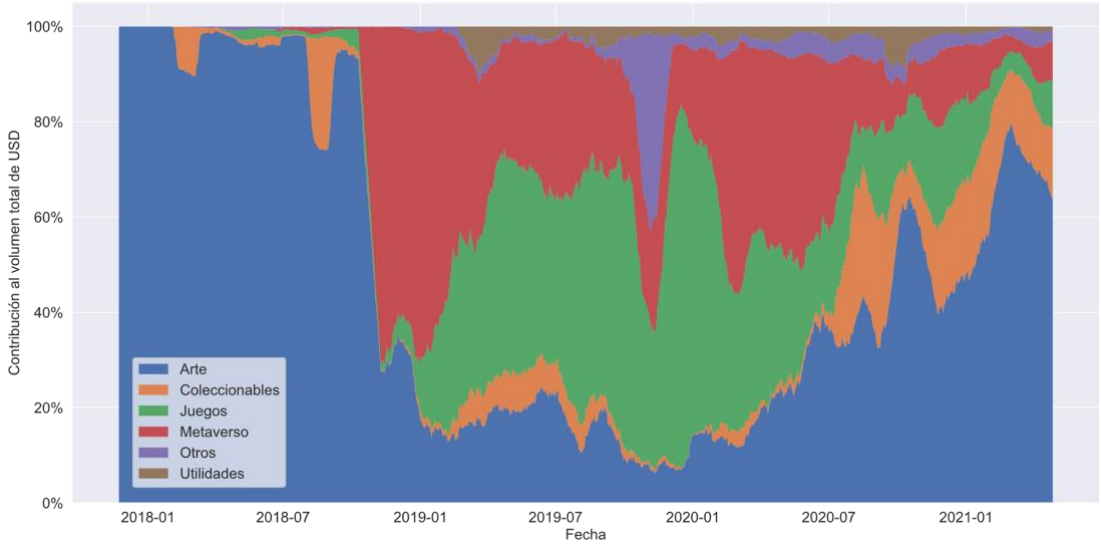

Figura 25. Contribución diaria al volumen total de USD intercambiado, por categoría. Calculado con una ventana rodante de 30 días.

4.9 Función de densidad acumulada del precio de los activos

Con el objetivo de observar la distribución del precio de venta en USD de los NFTs se calcula la función de densidad acumulada. La figura 26a muestra esta distribución para todos los activos, teniendo en cuenta todos los datos del *dataset*. El 75% de las transacciones tienen un valor de venta igual o mayor a 23 céntimos de USD, mientras que el percentil 50 para la totalidad de activos equivale a 1,43 USD. El 25% de las ventas iguala o supera los 13,8 dólares, el 5% alcanza los 247,1 USD y el 1% equivale a transacciones por valor igual o superior a 1840,7 dólares. Solo el 0,18% de las operaciones alcanzan los 10000 USD, el 0,01% los 100000 USD y el 0,002% el millón de dólares.

La figura 26b evidencia la distribución del precio de venta de todas las transacciones, por categoría. La figura indica que las categorías Arte, Metaverso y Juegos son las que mayor precio alcanzan en comparación con el resto. El 1% de las transacciones igualan o superan los 5201 USD, 11082 USD y 329 USD para las categorías Arte, Metaverso y Juegos, respectivamente. El número de activos en estas categorías es diferente, el 1% corresponde a 7697, 371 y 21426 NFTs, respectivamente. En la categoría Arte hay 10 transacciones que superan el millón de dólares, en el caso de Metaverso son 2 y el valor máximo para Juegos es 514974 USD.

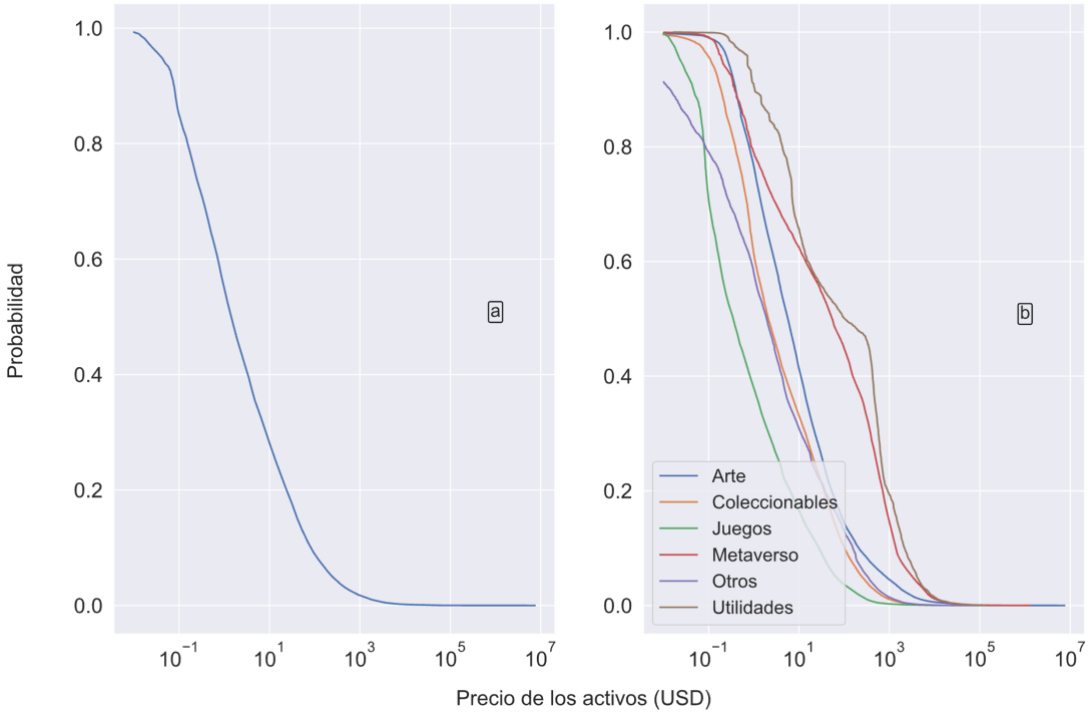

Figura 26. Función de densidad acumulada del precio de compraventa de NFTs en USD. a) Distribución del precio de todos los activos. b) Distribución del precio por categoría.

4.10 Poder de los traders en relación con otros traders

Utilizando el algoritmo para calcular el grado de los vértices (traders), descrito en la sección 3.3.1, se analiza el poder de los traders en relación con otros traders. Al emplear el caso 2 del algoritmo, se identifica el poder de un trader como la cantidad total de traders que tiene como compradores y vendedores.

De acuerdo a los valores de probabilidad encontrados y mostrados en la figura siguiente, el 31% de los traders solamente están asociados a otro trader, ya sea como comprador o vendedor. El 56,7% de los mismos tiene grado menor a 3 y el 5% iguala o supera los 36 traders asociados. Solo el 1% de los traders tiene poder superior a 168.

La figura 28 contiene el máximo, la media y la mediana del poder de los traders, calculado en ventanas de 1 día. Al comparar esta figura con las figuras 22 y 23, se identifica que el pico máximo de poder de traders a finales de 2017, en la figura 28, se corresponde con el incremento en el volumen y el número de transacciones en la misma fecha. Existe un comportamiento similar para el pico de finales de 2018 y los del año 2021. Solamente hay dos valores de mediana para el análisis diario realizado. De los 1252 días que incluye el *dataset*, 1214 días tiene mediana 1 para el poder de los traders, mientras que los 38 días restantes la mediana es 2.

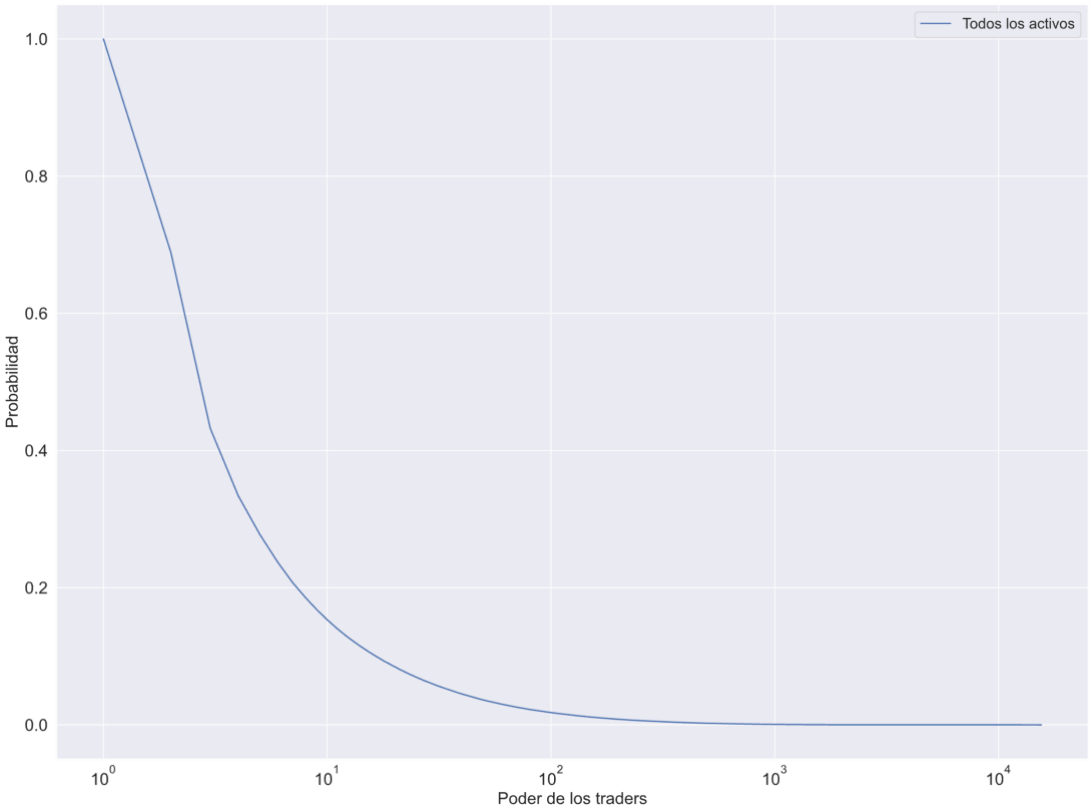

Figura 27. Función de densidad acumulada del poder de los traders en relación con otros traders.

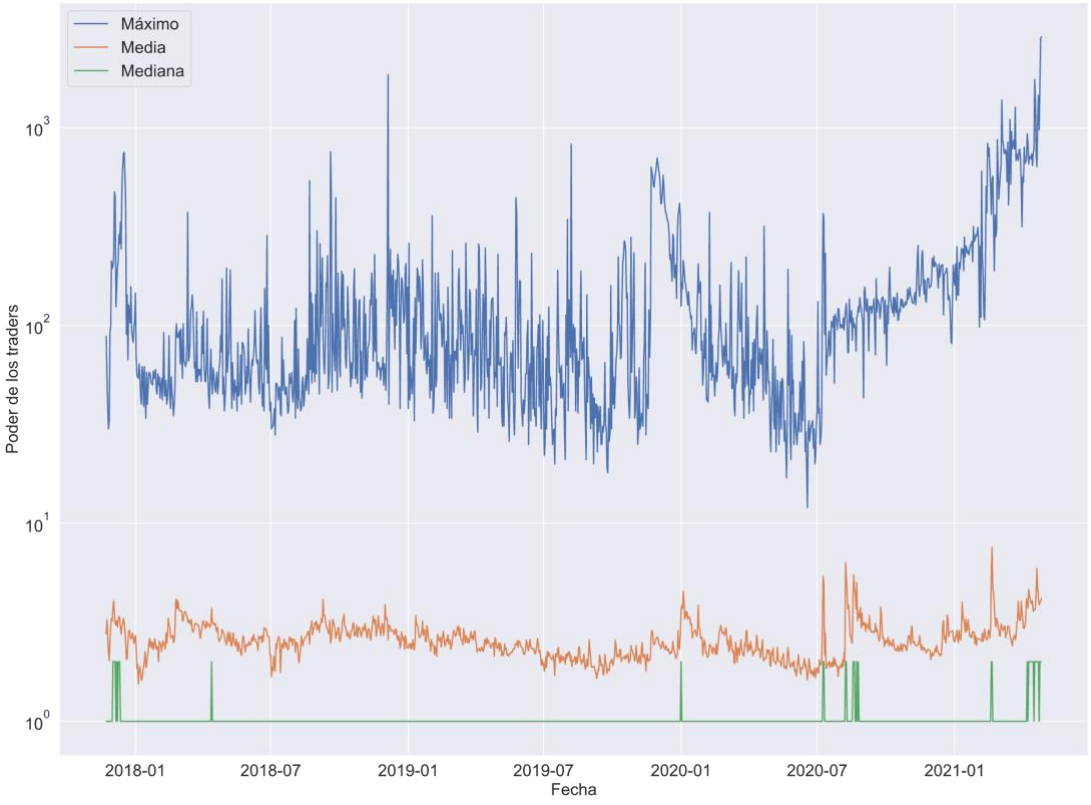

Figura 28. Máximo, media y mediana del poder de los traders en relación con otros traders, calculado con ventanas de 1 día.

4.11 Peso de los enlaces

El algoritmo descrito en 3.3.2 permite caracterizar las relaciones entre parejas de traders. Al calcular el peso de los enlaces, es posible conocer la distribución que existe del número de NFTs intercambiadas entre dos traders. A continuación, se presenta la función de probabilidad acumulada del peso de los enlaces, para todo el período analizado:

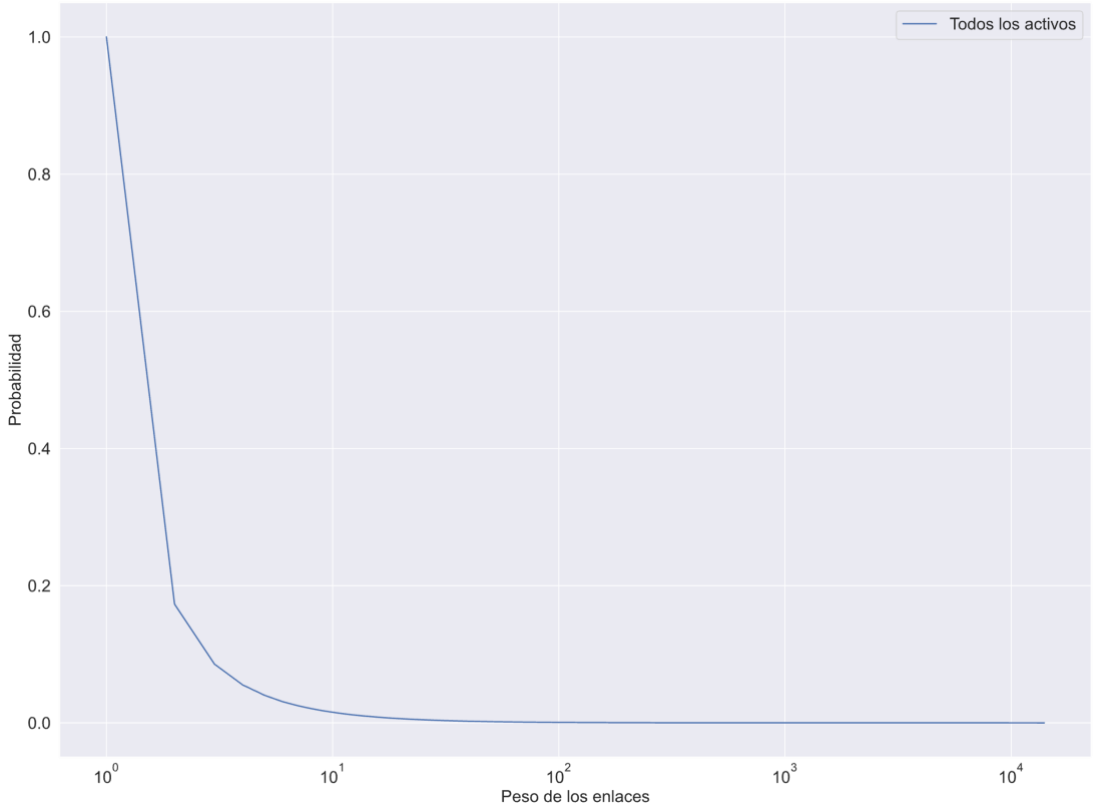

Figura 29. Función de densidad acumulada del peso de los enlaces.

El 82,7% de las parejas de traders intercambian solamente 1 NFT, el 1,55% iguala o supera los 10 NFTs, el 0,07% corresponde a 1000 NFTs o más y el 0,003% supera el millar de activos. Estos resultados indican que, en la gran mayoría de los casos, los traders diversifican sus operaciones entre múltiples compradores/vendedores, es decir, hay muy pocas parejas que intercambian un número significativo de NFTs.

La siguiente figura presenta una serie temporal del peso máximo, medio y la mediana de los enlaces, calculado con ventanas de 1 día. Similar al caso del poder de los traders, existe una correspondencia entre los picos de la figura 30 y las de volumen diario y número de transacciones. Se identifica que los picos máximos del peso de los enlaces a finales de 2017, finales de 2018 y durante 2021, en la figura siguiente, se corresponden con el incremento en el volumen y el número de transacciones en la misma fecha. El ascenso de finales de 2019 concuerda con un pico de transacciones de NFTs de la

categoría Juegos, visible también en el volumen. Durante los 1252 días que incluye el *dataset*, la mediana del peso de los enlaces es uno.

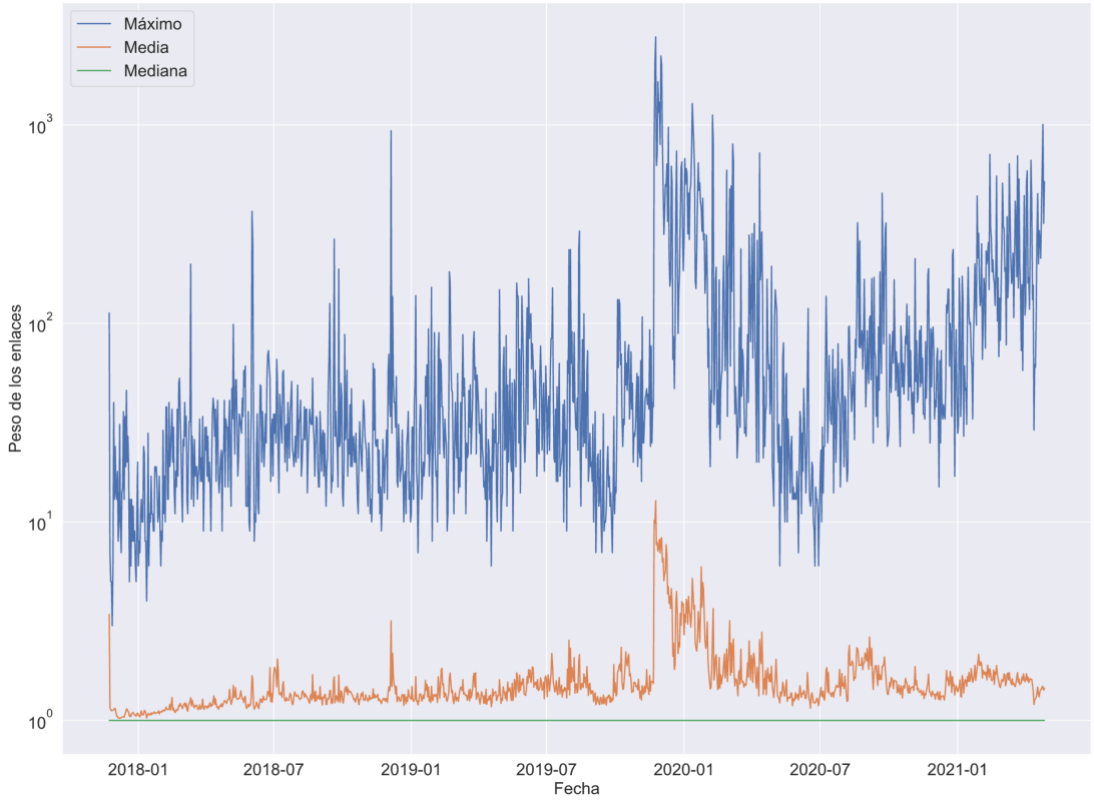

Figura 30. Máximo, media y mediana del peso de los enlaces, calculado con ventanas de 1 día.

4.12 Poder de los traders en relación con los NFTs

Como se ha mencionado en la descripción del algoritmo en 3.3.3, el poder de los traders en relación a los NFTs indica el número total de NFTs vendidos y comprados por cada trader. La figura 31 muestra la distribución de esta variable en todo el *dataset*.

El 26% de los traders solamente ha comprado o vendido 1 NFT, mientras que el 51,2% ha intercambiado menos de 3 NFTs. El 18,6% de los traders tiene un poder igual o superior a 10, el 2,8% supera los 100 NFTs y el 0,3% tiene más de 1000.

En la figura 32 se presentan series temporales con el máximo, media y mediana del poder de los traders, calculadas con ventanas de 1 día. Al comparar la figura 32 con las figuras 22 y 25, se observa una correspondencia del pico de finales de 2018 con el ascenso en volumen de NFTs de la categoría Metaverso. Existe un comportamiento similar en el pico de finales de 2019, coincidente con un ascenso de volumen y transacciones en la categoría Juegos. La tendencia de aumento en 2021 también concuerda con el incremento en número de transacciones y volumen en las mismas fechas. De los 1252 días, en 1056 días la mediana es 1, en 193 días la mediana es 2, mientras que es 1,5 en 1 día y 3 en dos días. La mediana 3 coincide con los picos de finales de 2017 y 2019.

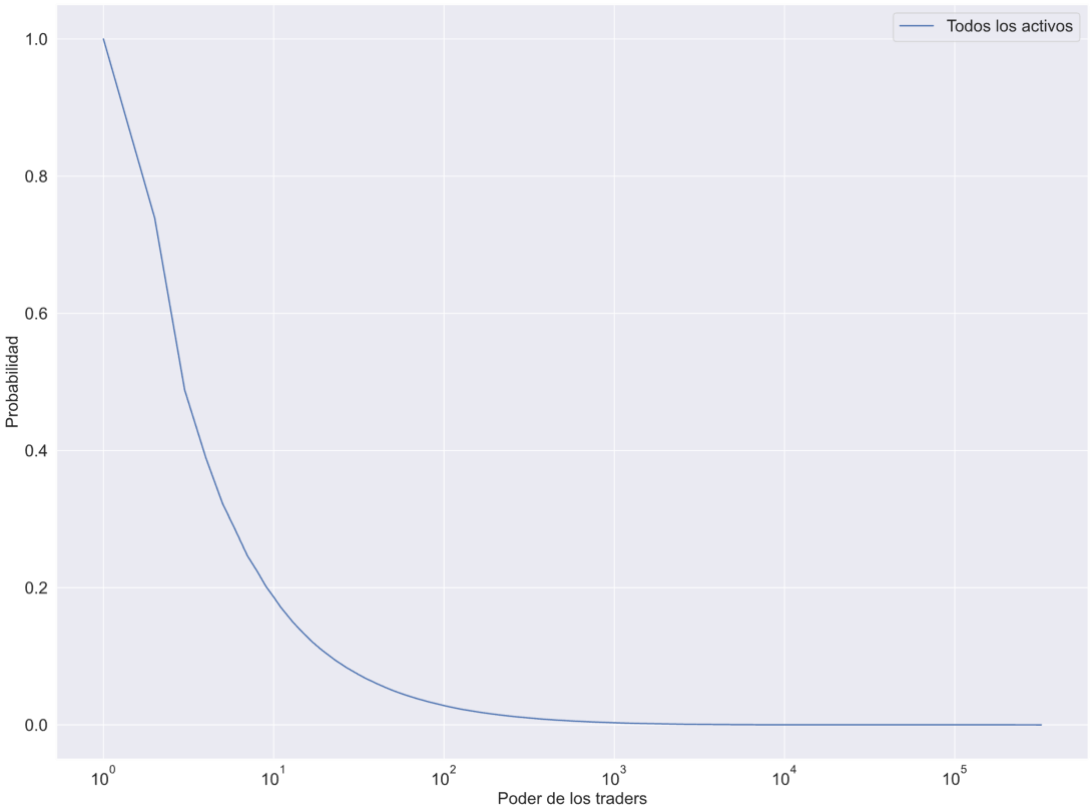

Figura 31. Función de densidad acumulada del poder de los traders en relación a los NFTs.

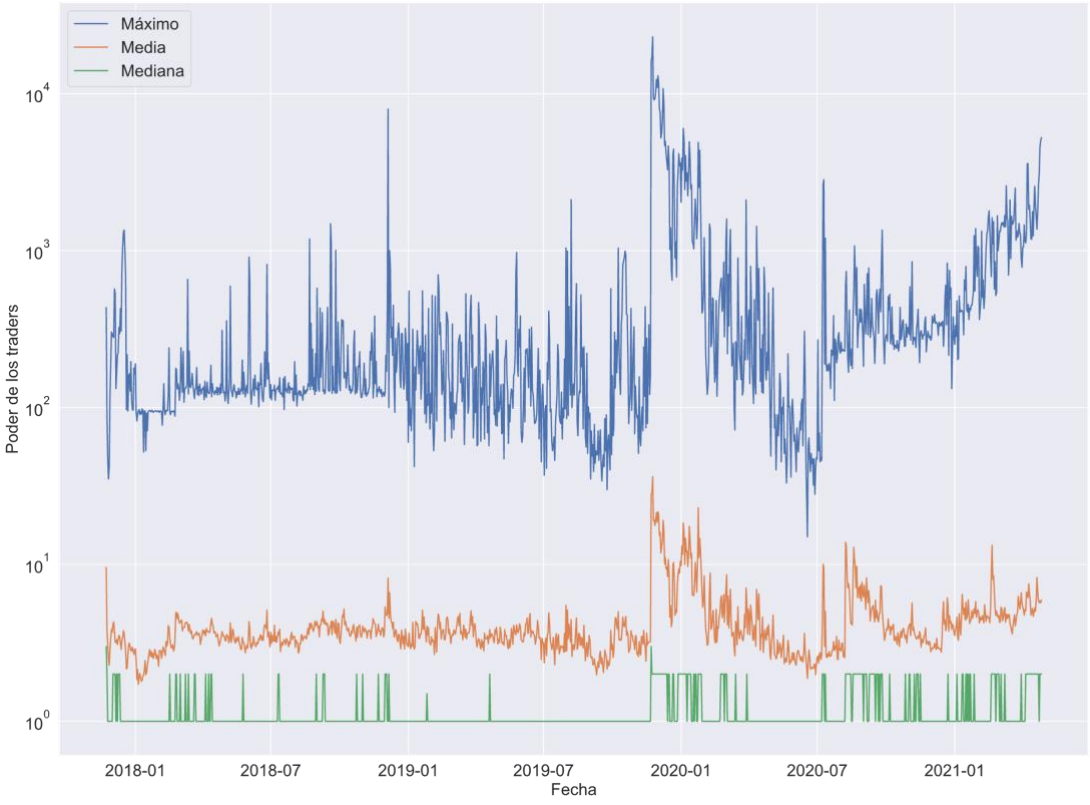

Figura 32. Máximo, media y mediana del poder de los traders, calculado con ventanas de 1 día.

4.12.1 Poder de los traders en relación con los días de actividad

Con el objetivo de analizar la relación existente entre el poder de los traders y el número de días de actividad se analiza la distribución de esta segunda variable:

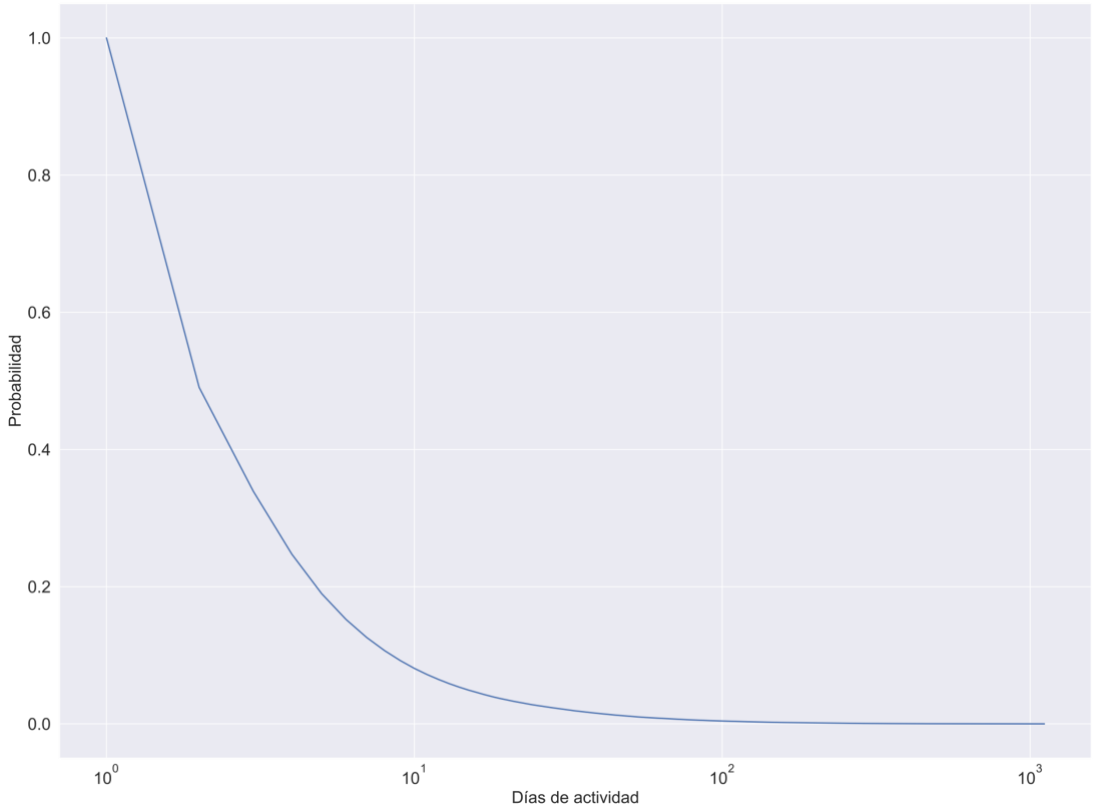

Figura 33. Función de densidad acumulada de los días de actividad.

El 51% de los traders en el *dataset* solamente comerciaron 1 día, el 8% tiene una actividad de 10 o más días, mientras que el 0,4% igualan o superan los 100 días de operaciones y solo el 0,0008% superan los 1000 días de actividad. El número máximo de días que un trader está activo es 1112.

A continuación, se presenta una gráfica que muestra la relación entre las dos variables. La figura muestra que hay una relación lineal para casi todo el rango de días de actividad. A partir de los 417 días hay un cambio de tendencia y luego de los 462 días vuelve a aumentar el poder con respecto a la actividad, aunque de una forma menos lineal. En general la tendencia es que los traders que dedican más tiempo al intercambio en el mercado de NFTs tienen mayor poder.

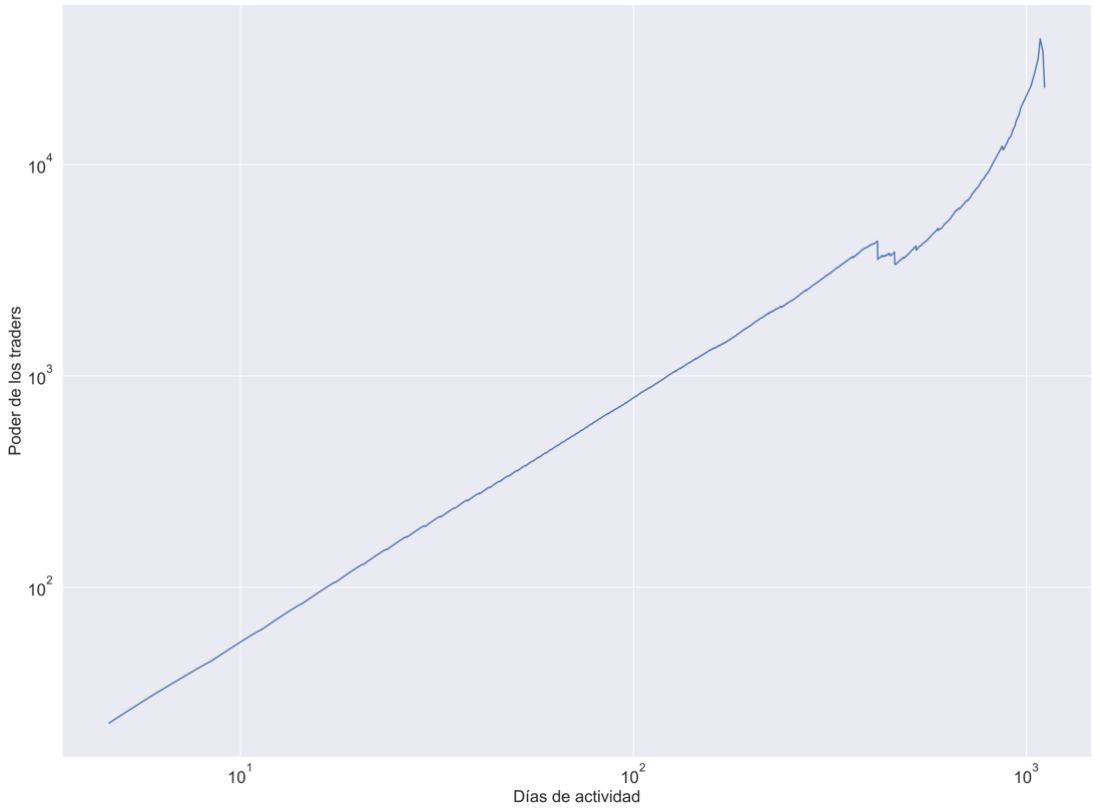

Figura 34. Poder de los traders en relación a los días de actividad.

En este capítulo se han presentado los resultados obtenidos al analizar el mercado de NFTs. Se ha caracterizado dicho mercado desde un punto de vista general, observando la distribución de NFTs por categorías y criptomonedas, así como el volumen intercambiado en USD y el número de transacciones. Se ha estudiado también las contribuciones de cada categoría al volumen y transacciones totales, al igual que la distribución del precio de los activos. Por otro lado, se han empleado los algoritmos presentados en el capítulo anterior para conocer la forma en la que interactúan los traders en el mercado. Este análisis ha permitido comprender la distribución del número de NFTs que los traders poseen, el número de traders con los que cada trader comercia, el número de NFTs que se intercambia entre las parejas de traders y la relación entre el poder y los días de actividad de los actores del mercado.

5. Conclusiones y líneas futuras de investigación

En este capítulo se presenta un resumen de los resultados obtenidos en este trabajo de investigación. Se realiza también un resumen de los métodos y herramientas empleados para la obtención de los resultados. De acuerdo a los resultados obtenidos, ha sido posible satisfacer los objetivos planteados al inicio de la investigación, modelando satisfactoriamente el mercado de NFTs a lo largo de un intervalo de tiempo en una plataforma de Big Data.

5.1 Conclusiones

El mercado de NFTs, relativamente nuevo, ha experimentado un gran crecimiento en 2021. En este trabajo se ha realizado un modelado de dicho mercado a partir de un *dataset* de transacciones de NFTs, que contiene datos desde noviembre de 2017 hasta abril de 2021.

El mercado de NFTs se ha modelado como un grafo temporal, donde los vértices pueden ser de tipo Trader o NFT. Cuando ocurre una transacción, se establece un enlace entre la pareja de traders participantes (relación Trader-Trader), y a la vez se crea una arista entre el nodo Trader comprador y los NFTs adquiridos (relación Trader-NFT).

Las propiedades asociadas a traders, NFTs y transacciones varían con la dimensión temporal, por lo que se ha analizado diversas herramientas de modelado de grafos para verificar su idoneidad para realizar análisis temporales sobre los mismos. Entre las herramientas estudiadas se encuentran Neo4j, Spark GraphX y Raphtory. Se ha seleccionado Raphtory para realizar el presente estudio debido a su capacidad para ejecutar análisis en tiempo real de grafos temporales distribuidos, que le permite cargar y procesar grandes conjuntos de datos dinámicos a lo largo del tiempo.

Raphtory ha brindado la flexibilidad necesaria para la implementación de algoritmos que caracterizan el grafo temporal y sus propiedades. Teniendo en cuenta el enfoque centrado en vértice que emplea esta herramienta, se han desarrollado 3 algoritmos que hacen uso de la comunicación entre los vértices de tipo Trader y NFT. Uno de los algoritmos permite hallar el grado de los vértices de tipo Trader, teniendo en cuenta solamente sus vecinos de tipo Trader, los de tipo NFT, o ambos. Este algoritmo se ha utilizado para analizar la distribución del poder de los traders con respecto a otros traders. El segundo algoritmo implementado calcula el peso de las aristas que se establecen entre nodos de tipo Trader al ocurrir las transacciones. Este algoritmo ha sido empleado para comprobar la distribución del peso de los enlaces entre parejas de traders, teniendo en cuenta diferentes ventanas temporales. Por último, se ha creado un algoritmo que ofrece, el poder de traders con respecto al número de NFTs que poseen. Por otro lado, también permite obtener, entre otros parámetros, el número de días de actividad de los traders.

Se ha realizado un análisis estadístico del mercado de NFTs utilizando Spark y Pandas, para estudiar aquellas variables que no son propias del grafo modelado. Se han encontrado transacciones asociadas a un total de 4621 colecciones diferentes y se ha observado que en la categoría Arte, el número de NFTs en las 5 colecciones más grande presenta la mayor disparidad, conteniendo la colección Cryptokittie el 82,6 % del total de activos de las 5 colecciones.

En cuanto a la distribución de NFTs y transacciones por categoría, el primero y segundo lugar para ambas variables corresponden a las categorías Juegos y Coleccionables, respectivamente. Atendiendo al número de NFTs y transacciones por criptomoneda, en ambos casos la mayor cantidad pertenece a WAX, seguida por Ethereum.

Al comprobar el volumen diario intercambiado en el mercado en USD, se ha encontrado un aumento significativo a partir de julio de 2020, con otro pico a partir de enero de 2021 hasta el final de los datos disponibles. El volumen medio del 1 enero al 27 de abril de 2021 es de 6,4 millones de USD aproximadamente. En el caso del número de transacciones diarias el comportamiento es muy similar, incrementándose notablemente en 2021. El número máximo de transacciones pasa de las decenas de miles en el período de 2017 a 2020, hasta casi 155000 en 2021, de enero a abril.

El cálculo de la contribución al número de transacciones por categoría ha indicado que las categorías más significativas han sido Arte, Juegos y Coleccionables. Por otro lado, las mayores contribuciones al volumen en USD han estado lideradas por las categorías Arte, Juegos y Metaverso.

Se ha comprobado que el precio de venta de los NFTs es relativamente bajo, donde el 25% de las ventas iguala o supera los 13,8 USD, el 5% alcanza los 247,1 USD y el 1% equivale a transacciones por valor igual o superior a 1840,7 dólares. De acuerdo al análisis por categoría, las categorías Arte, Metaverso y Juegos son las que tienen las transacciones de mayor valor. En la categoría Arte hay 10 transacciones que superan el millón de dólares, en el caso de Metaverso son 2 y el valor máximo para Juegos es 514974 USD.

Al analizar los resultados obtenidos de la ejecución de los algoritmos diseñados en Raphtory, se ha estudiado las distribuciones del poder de los traders en relación con otros traders, el peso de los enlaces y el poder de los traders en relación con los NFTs, así como la relación entre el poder y los días de actividad. Se ha determinado que el 31% de los traders solamente están asociados a otro trader, mientras que solo el 1% de los traders tiene poder superior a 168. En cuanto al peso de los enlaces, el 83% de las parejas de traders intercambian solamente 1 NFT, entre tanto, el 1,55% de las mismas comercian

10 o más de estos activos. Al estudiar el poder de los traders en relación con el número de NFTs que han intercambiado, se ha encontrado que el 26% de los traders solamente ha comprado o vendido 1 NFT, el 18,6% de los traders tiene un poder igual o superior a 10, el 2,8% supera los 100 NFTs y el 0,3% tiene más de 1000. Se encuentra una correspondencia entre el comportamiento diario de los resultados de salida de los 3 algoritmos y el comportamiento en cuanto a número de transacciones y volumen intercambiado en el mercado de NFTs.

El análisis de la distribución de la actividad de los traders indica que el 51% de los traders en el *dataset* solamente comerciaron 1 día, mientras que el 0,4% igualan o superan los 100 días de operaciones. Se ha encontrado una relación mayormente lineal entre la actividad y el poder de los traders, donde los traders que participan en el mercado por más tiempo tienen mayor poder.

5.2 Líneas futuras de investigación

Se proponen las siguientes líneas de investigación con el objetivo de ampliar el entendimiento del mercado de NFTs:

- Ejecutar el análisis realizado para un *dataset* con datos actualizados.
- Realizar estudios adicionales del mercado: análisis de la distribución del número de veces que los NFTs son vendidos, porcentaje de transacciones que los traders realizan en sus colecciones primaria y secundaria.

Bibliografía

[1] Gupta, M.S. (2022, February 1). *The Most Expensive NFT artworks ever sold*. https://www.prestigeonline.com/my/pursuits/wealth/most-expensive-nfts-sold-till-date/

[2] Nadini, M., Alessandretti, L., Di Giacinto, F., Martino, M., Aiello, L. M., & Baronchelli, A. (2021). Mapping the NFT revolution: market trends, trade networks, and visual features. *Scientific reports*, *11*(1), 1-11.

[3] Rosenfeld, M. (2012). Overview of colored coins. *White paper, bitcoil. co. il*, *41*, 94.

[4] The Art Newspaper. (2022, February 5). *Sotheby's and artist Kevin McCoy sued over sale of early NFT*. https://www.theartnewspaper.com/2022/02/04/sothebys-kevin-mccoy-lawsuit-quantum-nft

[5] Ante, L. (2021). The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum. *Available at SSRN 3861106*.

[6] Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*.

[7] Entriken, W., Shirley, D., Evans, J. and Sachs, N., 2018. Erc-721 non-fungible token standard. *Ethereum Foundation*.

[8] Ji, Q., Bouri, E., Lau, C.K.M. and Roubaud, D., 2019. Dynamic connectedness and integration in cryptocurrency markets. International Review of Financial Analysis, 63, pp.257-272.

[9] Meynkhard, A., 2020, October. Effect of bitcoin volatility on altcoins pricing. In *Proceedings of the Computational Methods in Systems and Software* (pp. 652-664). Springer, Cham.

[10] Biggs, N., Lloyd, E. K., & Wilson, R. J. (1986). Graph Theory, 1736-1936. Oxford University Press.

[11] Gross, J. L., & Yellen, J. (2003). *Handbook of graph theory*. CRC press.

[12] Michail, O., 2016. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, *12*(4), pp.239-280.

[13] Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., & Partner, J. (2015). *Neo4j in action* (Vol. 22). Shelter Island: Manning.

[14] Sasaki, B. M., Chao, J., & Howard, R. (2018). Graph databases for beginners. *Neo4j*.

[15] Neo4j. (n.d.). *Getting Started with Neo4j*. Retrieved June 15, 2022, from https://neo4j.com/developer/get-started/

[16] Neo4j. (n.d.). *Neo4j Cypher Query Language*. Retrieved June 15, 2022, from https://neo4j.com/product/cypher-graph-query-language/?ref=product

[17] Neo4j. (n.d.). *Neo4j Graph Data Platform*. Retrieved June 15, 2022, from https://neo4j.com/product/

[18] Hodler, A.E. (2018, April 23). *Graph Algorithms in Neo4j: 15 Different Graph Algorithms & What They Do*. https://neo4j.com/blog/graph-algorithms-neo4j-15-different-graph-algorithms-and-what-they-do/

[19] Institut National de Recherche en Sciences et Technologies du Numérique. (n.d.). *T-Cypher: A Temporal Graph Query Language*. Retrieved June 16, 2022, from https://project.inria.fr/tcypher/

[20] Apache Spark. (n.d.). *GraphX Programming Guide*. Retrieved June 16, 2022, from https://spark.apache.org/docs/latest/graphx-programming-guide.html

[21] Malak, M., & East, R. (2016). *Spark GraphX in action*. Simon and Schuster.

[22] Junghanns, M., Petermann, A., Neumann, M., & Rahm, E. (2017). Management and analysis of big graph data: current systems and open challenges. In *Handbook of big data technologies* (pp. 457-505). Springer, Cham.

[23] Rost, C., Gomez, K., Täschner, M., Fritzsche, P., Schons, L., Christ, L., ... & Rahm, E. (2022). Distributed temporal graph analytics with GRADOOP. *The VLDB journal*, *31*(2), 375-401.

[24] Raphtory. (n.d.). *Raphtory Temporal Graph Analysis*. Retrieved June 15, 2022, from https://raphtory.readthedocs.io/en/master/

[25] Raphtory. (n.d.). *Analysis in Raphtory*. Retrieved June 15, 2022, from https://raphtory.readthedocs.io/en/master/Analysis/analysis-explained.html

[26] Raphtory. (n.d.). *Algorithms*. Retrieved June 15, 2022, from https://raphtory.readthedocs.io/en/master/_autodoc/com/raphtory/algorithms/index.html

[27] Raphtory. (n.d.). *Running Queries*. Retrieved June 15, 2022, from https://raphtory.readthedocs.io/en/master/Analysis/queries.html

[28] Raphtory. (n.d.). *Raphtory Overview*. Retrieved January 25, 2023, from https://www.raphtory.com/about

[29] Kugler, L. (2021). Non-fungible tokens and the future of art. *Communications of the ACM*, *64*(9), 19-20.

[30] Valeonti, F., Bikakis, A., Terras, M., Speed, C., Hudson-Smith, A., & Chalkias, K. (2021). Crypto collectibles, museum funding and OpenGLAM: challenges, opportunities and the potential of Non-Fungible Tokens (NFTs). *Applied Sciences*, *11*(21), 9931.

[31] Dowling, M. (2022). Fertile LAND: Pricing non-fungible tokens. *Finance Research Letters*, *44*, 102096.

[32] Steer, B., Cuadrado, F., & Clegg, R. (2020). Raphtory: Streaming analysis of distributed temporal graphs. *Future Generation Computer Systems*, *102*, 453-464.

[33] Steer, B. A. (2021). *Raphtory: modelling, maintenance and analysis of distributed temporal graphs* (Doctoral dissertation, Queen Mary University of London).

[34] Waudby, J., Steer, B. A., Prat-Pérez, A., & Szárnyas, G. (2020, June). Supporting Dynamic Graphs and Temporal Entity Deletions in the LDBC Social Network Benchmark's Data Generator. In *Proceedings of the 3rd Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)* (pp. 1-8).

[35] Xin, R. S., Gonzalez, J. E., Franklin, M. J., & Stoica, I. (2013, June). Graphx: A resilient distributed graph system on spark. In *First international workshop on graph data management experiences and systems* (pp. 1-6).

[36] Gombos, G., Rácz, G., & Kiss, A. (2016, August). Spar (k) ql: SPARQL evaluation method on Spark GraphX. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW) (pp. 188-193). IEEE.*

[37] *Arnold, N. A., Steer, B., Hafnaoui, I., Parada G, H. A., Mondragón, R. J., Cuadrado, F., & Clegg*, R. G. (2021). Moving with the Times: Investigating the Alt-Right Network Gab with Temporal Interaction Graphs. *Proceedings of the ACM on Human-Computer Interaction*, *5*(CSCW2), 1-17.

ANEXO A: ASPECTOS ÉTICOS, ECONÓMICOS, SOCIALES Y AMBIENTALES

A.1 INTRODUCCIÓN

El presente trabajo investigativo es parte del sector tecnológico de sistemas telemáticos y constituye el Trabajo de Fin de Máster correspondiente al Máster Universitario en Redes y Servicios Telemáticos de la Universidad Politécnica de Madrid (UPM). Se presenta como objetivo aumentar el conocimiento relativo al mercado de NFTs mediante el análisis y caracterización del mismo, que no ha sido investigado exhaustivamente. Existen varios grupos de interés asociados al desarrollo de este proyecto. Por un lado, se encuentra el Departamento de Ingeniería de Sistemas Telemáticos y el grupo de investigación *Real-Time Systems and Architecture of Telematic Services* (STRAST), pertenecientes a la Escuela Técnica Superior de Ingeniería de Telecomunicación de la UPM, desde donde ha sido publicado el tema de investigación tratado. Por otro lado, se encuentra la comunidad científica con interés en los temas tratados y la sociedad en general, estando el mercado de NFTs al alcance de muchas personas.

A.2 DESCRIPCIÓN DE IMPACTOS RELEVANTES RELACIONADOS CON EL PROYECTO

Se han identificado varios aspectos relevantes relacionados con el proyecto desarrollado. Al analizar los resultados obtenidos, se ha determinado que este trabajo tiene un impacto positivo, tanto social como económicamente. Desde el punto de vista social, permite aumentar el acceso a la información, en este caso sobre grafos dinámicos, herramientas de modelado de grafos y el mercado de NFT. Por otro lado, contribuye al incremento de la educación a través de una investigación científica realizada en una universidad de prestigio internacional. Al extender la información relativa al mercado de NFTs, brinda la posibilidad de que un mayor número de personas entren al mercado.

La aportación de conocimiento y el acceso a nueva información es de interés para la universidad y los grupos de investigación relacionados, puesto que permite consolidar su actividad investigadora. Desde un punto de vista más general, la sociedad también se ve beneficiada con la generación de conocimiento. En este caso, al brindar información sobre las características y comportamiento del mercado en cuestión, puede haber una mejora de ingresos para nuevos traders potenciales y la adquisición de activos de interés.

A.3 IMPACTO SOCIAL

La democratización del conocimiento apunta a revalorizar la práctica de la investigación científica y tecnológica en vinculación con los objetivos del desarrollo social, orientando la promoción de nuevas investigaciones y la aplicación del conocimiento científico y tecnológico disponible a la resolución de los desafíos que plantea la producción de bienes y servicios, así como las problemáticas socialmente relevantes.

Tras analizar el estado del arte al comenzar el presenta proyecto de investigación, se ha determinado que el estudio del mercado de NFTs es aún limitado. Bajo el paraguas de generación de conocimiento que brindan las universidades, en este caso la UPM, se ha desarrollado este trabajo con la idea de incrementar el conocimiento y acceso a información relativa a los NFTs. El análisis realizado brinda información sobre el surgimiento y desarrollo de estos activos, así como su distribución en colecciones y características de los actores que componen el mercado.

Aunque no es sencillo determinar el alcance que puede tener este proyecto y su posible divulgación, puede ser un punto de apoyo para nuevas investigaciones y para la inclusión de personas interesadas en el mercado de NFTs.

A.4 CONCLUSIONES

Como se ha comentado anteriormente, el mayor impacto de este proyecto de investigación recae en la categoría social. Es posible que se pueda obtener también un beneficio económico al lograr que más personas se interesen en el mercado de NFTs. Desde el punto de vista ético, al ser un análisis de un sistema basado en datos que no contienen información personal, se debe tener en cuenta el posible impacto sobre la propiedad intelectual. En este trabajo se ha referenciado la bibliografía consultada, por lo que no existe un impacto negativo en esta categoría. Atendiendo a la categoría ambiental, no se han utilizado recursos de cómputo por el tiempo suficiente como para generar un impacto notable.

ANEXO B: PRESUPUESTO ECONÓMICO

La tabla que se muestra a continuación contiene el presupuesto estimado para el desarrollo del presenta proyecto investigativo:

	horas	Precio/hora	TOTAL
COSTE DE MANO DE OBRA (coste directo)	300	15 €	**4.500 €**

COSTE DE RECURSOS MATERIALES (coste directo)

	Precio de compra	Uso en meses	Amortización en años	TOTAL
Ordenador personal (Software incluido)	1.300,00 €	4	5	86,67 €
Servidor Laboratorio DIT (PowerEdge R650 Server)	8.950,00 €	4	5	596,67 €

TOTAL	**683,33 €**

GASTOS GENERALES (costes indirectos)	13%	sobre CD	**673,83 €**
BENEFICIO INDUSTRIAL	6%	sobre CD+CI	**351,43 €**

SUBTOTAL PRESUPUESTO		**6.208,60 €**
IVA APLICABLE	21%	**1.303,81 €**

TOTAL PRESUPUESTO	**7.512,40 €**

El coste de mano de obra para el desarrollo de este proyecto se ha llevado a cabo teniendo en cuenta el número de horas requeridas para completar el mismo. Considerando que el salario bruto de un ingeniero de IT ronda los 2500 €, el total de mano de obra equivale a 4500 €. Este trabajo se ha desarrollado entre mayo de 2022 y enero de 2023 en períodos no continuos. Se estima que el tiempo de uso de los recursos materiales ha sido 4 meses y que dichos recursos se amortizan en 5 años. Se ha empleado en este trabajo un ordenador personal valorado en 1300 €, incluido

el software necesario. Se estima que, con el uso dado a este recurso y el período de amortización considerado, el coste directo es de 86,67 €. Además, para el análisis realizado se ha utilizado un servidor del Departamento de Ingeniería de Sistemas Telemáticos. Este servidor es el modelo PowerEdge R650 Server de Dell y tiene las siguientes especificaciones:

- Procesador: Intel Xeon Silver 4314, 2,4 GHz, 16 núcleos/32 subprocesos
- RAM: GB de memoria RDIMM, 3200 MT/s, bloque doble
- Almacenamiento: 2 SSD SATA de lectura intensiva 480 GB 6 Gb/s
- Adaptador de red: 4 puertos de 1 GbE

Se estima que el coste de este servidor es de 8950 €, estableciendo 5 años de amortización, y un tiempo de uso de 4 meses, el coste directo equivale a 596,67 €. Se considera un coste indirecto del 13% sobre los costes directos, asociado al gasto energético correspondiente al desarrollo del proyecto, lo cual corresponde a 673,83 € de gastos generales. Para esta estimación, se ha tenido en cuenta que el servidor Dell tiene un consumo de 1100 W, que ha sido utilizado alrededor de 300 horas a alto rendimiento y un precio medio de 0,21 €/kWh. El porcentaje de beneficio industrial se considera en 6%, como generalmente es fijado, lo cual equivale a 351,43 €. Tras sumar los costes se obtiene un presupuesto de 6208,60 €, que al aplicarle el IVA del 21%, da el presupuesto total del proyecto, 7512,40 €.