**UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS**

**Alumnos**

Salto Cruz Anahí

Torres Hinojosa José Alfredo

**Unidad de Aprendizaje:**Aplicaciones Distribuidas

**Profesor**

Miguel Felix Mata

**Reporte del Proyecto *Red Bittorrent***

**Ciudad de México a 17 de enero de 2024.**

**Índicentenido**

[**Introducción 2**](#_e7ie3avgqwb)

[**Desarrollo 2**](#_ewdkrg9r31si)

[**Pruebas y Resultados 3**](#_2s8eyo1)

[**Observaciones y Conclusiones Finales 5**](#_17dp8vu)

[**Referencias 5**](#_o4arv6e90bw1)

|  |
| --- |
| Introducción |

Descripción de herramientas, bibliotecas, códigos integrados para el proyecto.

**Java RMI (Java Remote Method Invocation):** Java RMI es la biblioteca estándar de Java para la comunicación remota entre objetos en un entorno distribuido. Permite invocar métodos de objetos remotos de manera transparente, como si fueran métodos locales. Puedes utilizar esta herramienta para implementar la comunicación entre el cliente y el servidor en tu proyecto.

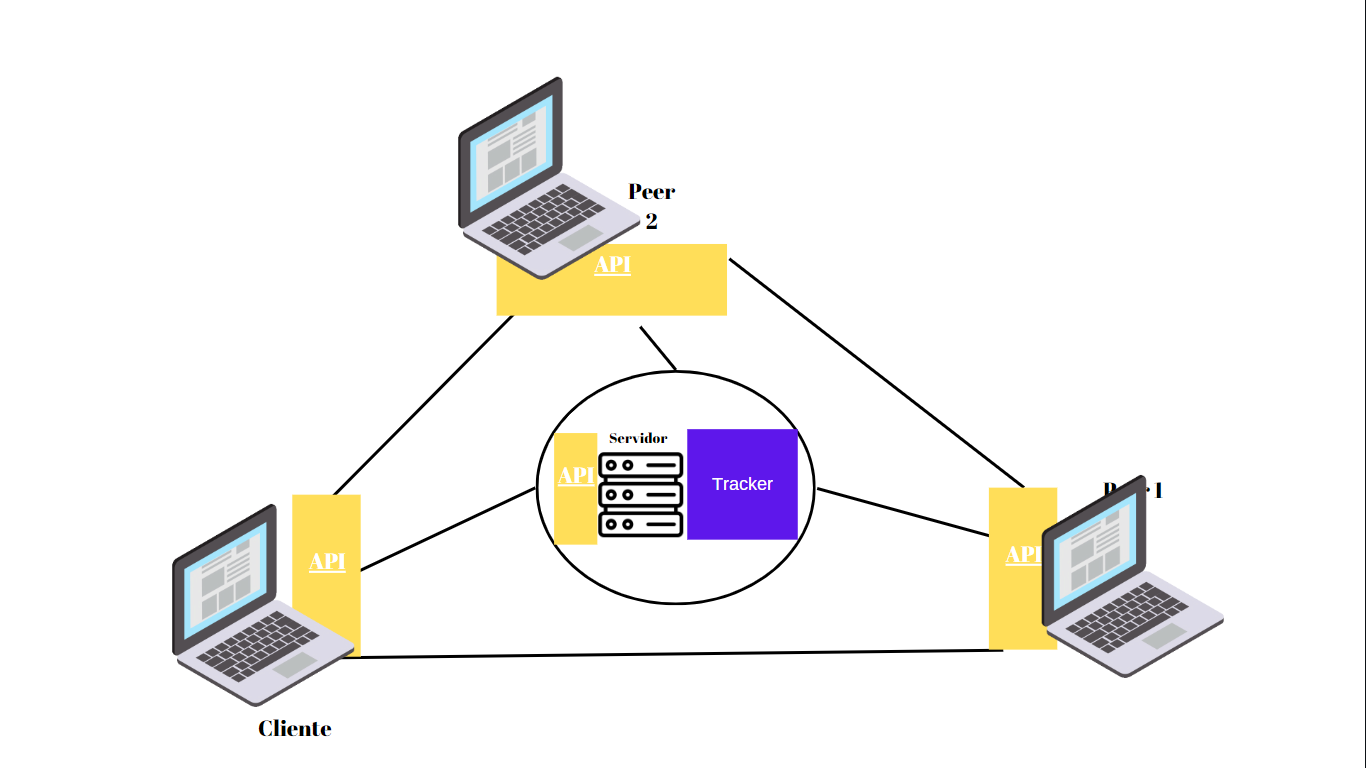
**Java NIO (New Input/Output):** La biblioteca Java NIO proporciona capacidades de E/S no bloqueantes y orientadas a canales. Es útil para construir sistemas de envío de archivos eficientes y escalables. Puedes utilizarla para implementar la transferencia de archivos entre el cliente y el servidor de manera asíncrona.

**Tabla o listado de códigos utilizados en el proyecto.**

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Referencia** |
| Código 1.1  PeerConnection.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Contiene información del peer, así como los métodos implementados de la interfaz TorrentPeer.java. | Código de elaboración propia |
| Código 1.2  PeerImp.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Implementa el servidor del peer y se ejecuta el menú principal de funciones. | Código de elaboración propia |
| Código 1.3  TorrentPeer.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Define los métodos remotos del peer. | Código de elaboración propia |
| Código 1.4  TorrentTrack.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Define los métodos remotos del tracker. | Código de elaboración propia |
| Código 1.5  TrackerConnectionr.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Contiene información del tracker, así como los métodos implementados de la interfaz TorrentTrack.java. | Código de elaboración propia |
| Código 1.6  TrackerImpr.java  Ubicado en la carpeta Proyecto en el enlace: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto> | Ejecuta el Tracker (Servidor). | Código de elaboración propia |

|  |
| --- |
| Desarrollo |

**Diagrama de la arquitectura de funcionamiento del proyecto, enfatizando los elementos distribuidos.**

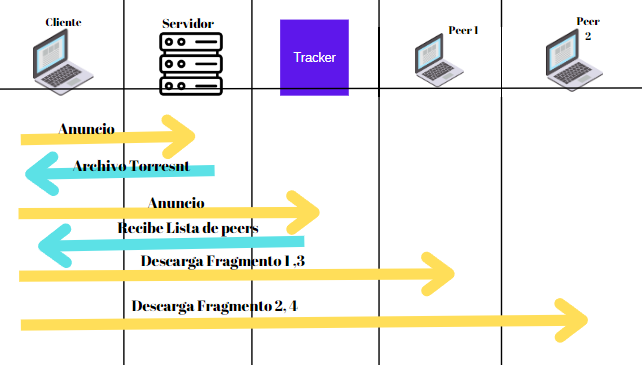
****

Se enfoca en la gestión de peers y trackers, permitiendo la comunicación y el intercambio de datos eficiente entre los participantes de la red. Estos códigos establecen la base para un sistema distribuido que facilita la descarga rápida y confiable de archivos a través de la división y distribución de partes entre los peers.

**Diseño de política de funcionamiento de descargas, transferencias y recuperación de errores**

En la implementación de la red BitTorrent, se sigue una política de atención a las descargas y transferencias de archivos conforme se van solicitando. Esto significa que los peers atienden las solicitudes de partes de archivos en el orden en que son recibidas. Esta política permite que los recursos de la red se utilicen de manera eficiente en una red de pocos nodos y equitativa entre los peers.

**Diagrama o descripción de conectividad**

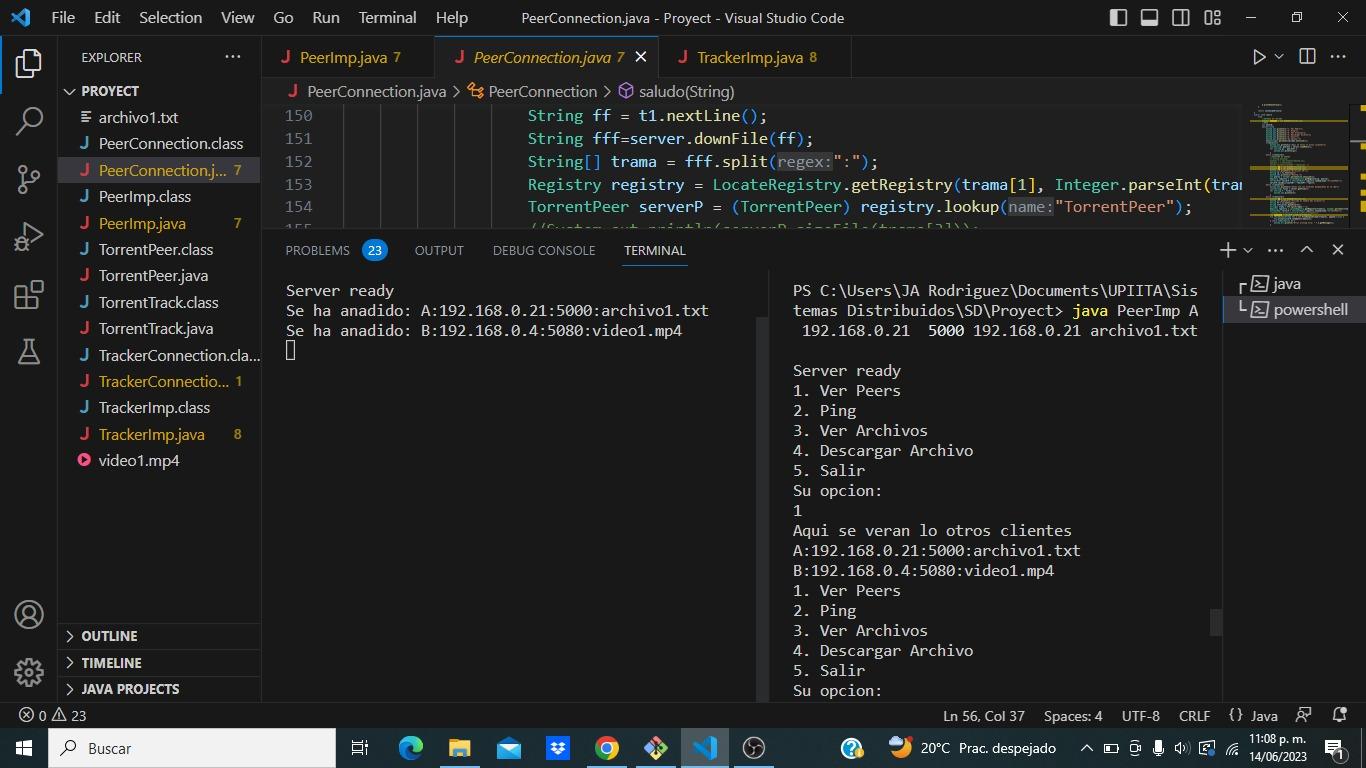


|  |
| --- |
| Pruebas y Resultados |

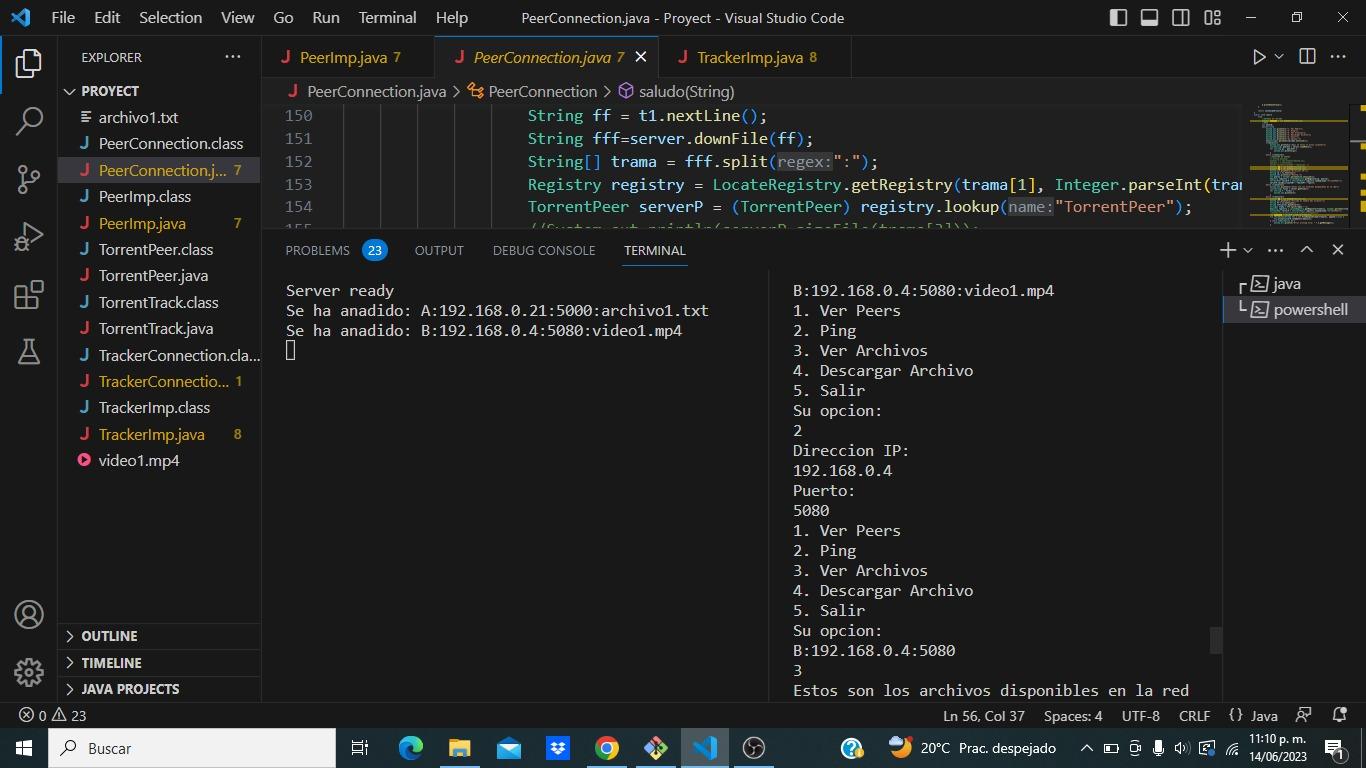
**Lista o tabla con explicación del Producto o funcionalidad no lograda**

|  |  |  |
| --- | --- | --- |
| **Producto** | **No se logró funcionalidad** | **Motivo o Circunstancia** |
| Conectividad con servidor AWS. | Instalar el servidor en una máquina virtual de AWS. | No se pudo configurar las instancias, para funcionar con un servidor AWS. |

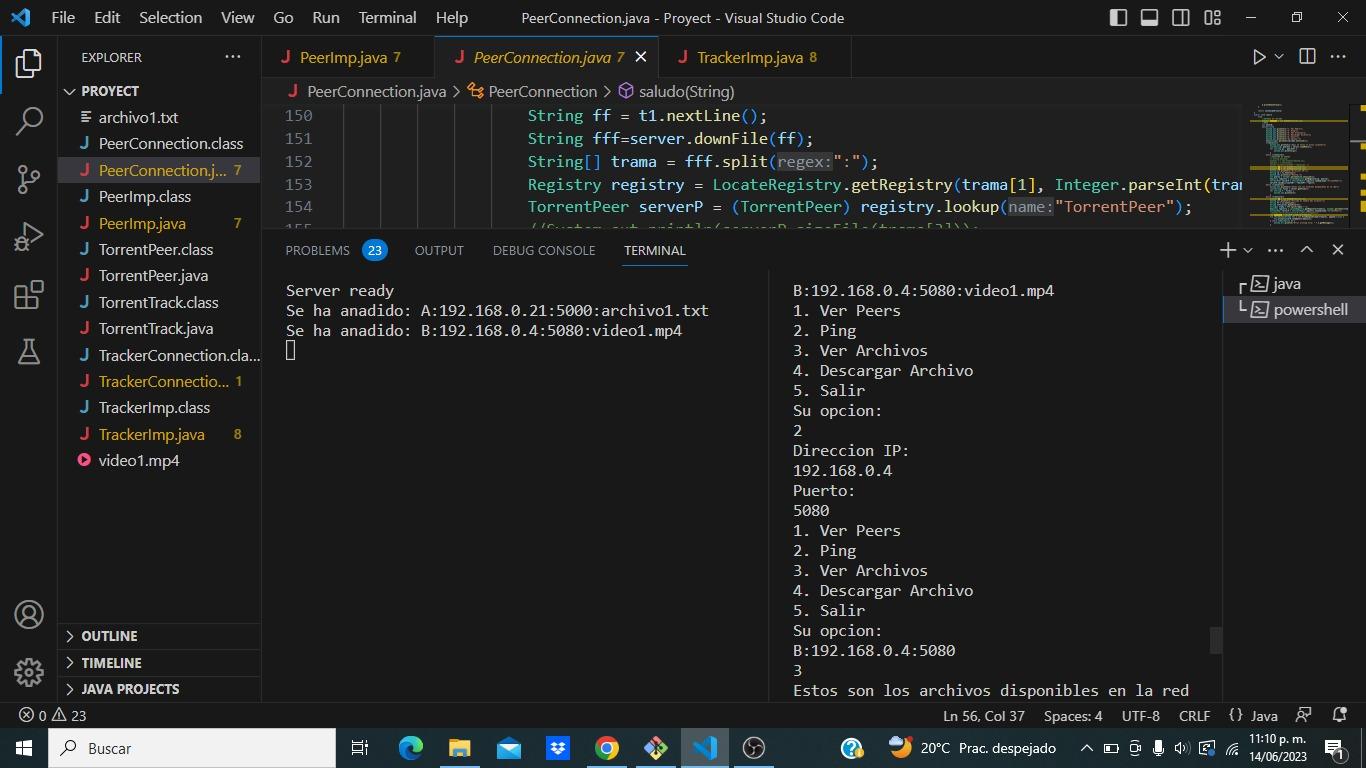
**Imágenes de la ejecución del proyecto con los pasos más relevantes y de los experimentos y resultados obtenidos , explicando en texto los detalles**

****

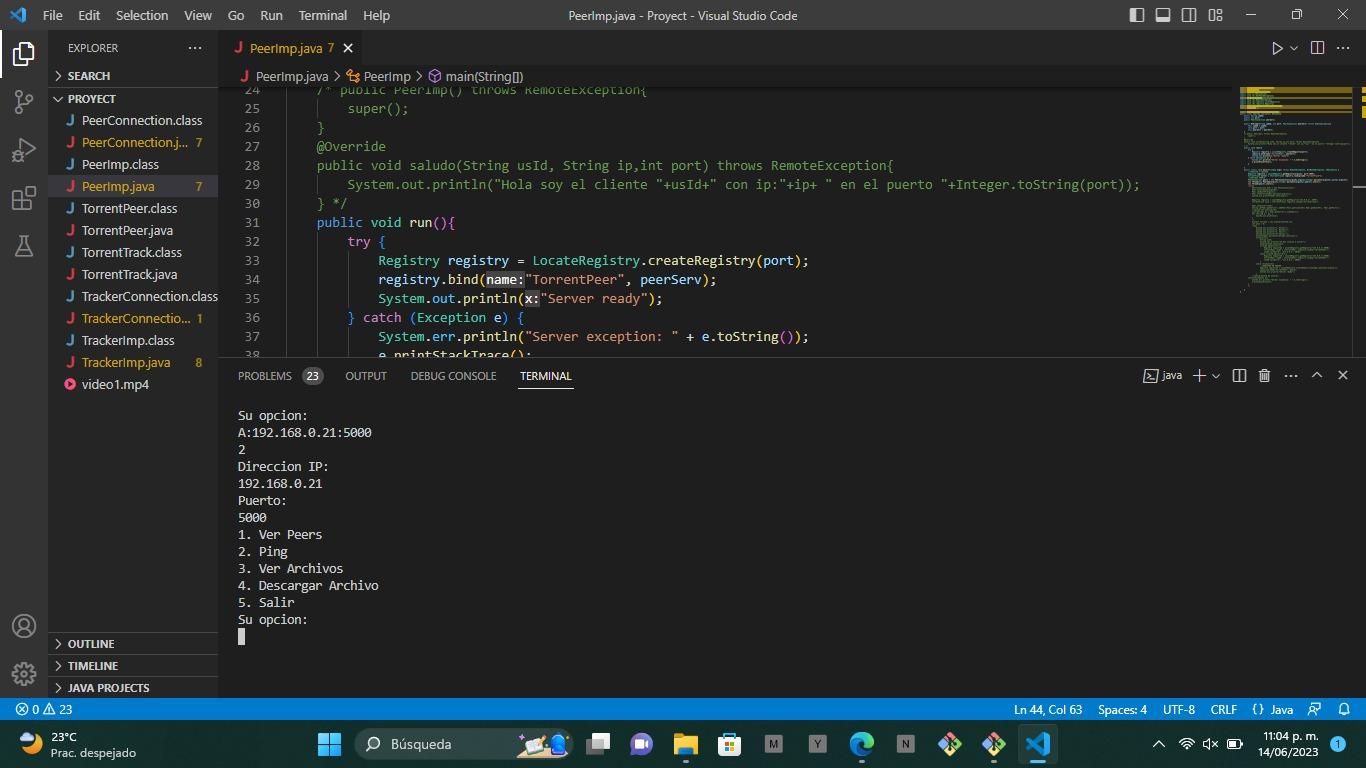
Ejecución: Opción 1 del menú

****

Ejecución: Opción 2 del menú

****

Ejecución: Opción 3, 4 ,5 del menú

****

Ejecución: Conexión con otro equipo

|  |
| --- |
| Observaciones y CONCLUSIONES |

Se puede concluir que:

La configuración de una red P2P, a pesar generar una comunicación más efectiva entre miembros de la red, y de no complicarse en cuestiones de lógica, el trabajo que representa en cuestión de kernel y la necesidad de implementar más recursos que en otros tipos de configuración, nos hace apegarnos más a las redes centralizadas, sin embargo no porque una red P2P implique más trabajo, significa que no se utilice, todo lo contrario, es el futuro y algunos protocolos como BitTorrent son ejemplo de la evolución de los sistemas distribuidos, permitiendo que otros sistemas descentralizados emerjan en la actualidad.

|  |
| --- |
| Referencias |

Repositorio en Github: <https://github.com/anahisc/Proyecto_AppsDistribuidas/tree/main/Proyecto>

# Vídeo: