https://doi.org/10.22201/dgtic.ctud.2023.1.1.13

Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 62 - 71

Diseño de una cola de hilos

Información del reporte:

Licencia Creative Commons

Para citar este reporte técnico:

Talavera Rosales, A. (2023). Diseño de una cola de hilos. *Cuadernos Técnicos Universitarios de la DGTIC, 1* (1), páginas (62 - 71). https://doi.org/10.22201/dgtic.ctud.2023.1.1.13

Alejandro Talavera Rosales

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

badboy@unam.mx

ORCID: 0009-0005-5489-2854

Resumen:

Desde hace décadas los sistemas operativos permiten el desarrollo de programas que aprovechen las capacidades de ejecución de varias tareas a la vez, lo que se conoce como tareas concurrentes. Si se incorpora correctamente esta cualidad a una aplicación, se puede lograr una reducción en el tiempo de respuesta para distintas tareas que se van realizando. Sin embargo, existen circunstancias donde no es posible realizar estas tareas, como es el caso de equipos con recursos computacionales limitados como los celulares de gama baja, en los que el uso de la programación de tareas concurrentes puede comprometer la ejecución de una aplicación, llegando incluso a la aparición de las denominadas *Screen of Death*[1], con la subsecuente falla del equipo. En estos casos se debe limitar el uso de programación concurrente para no exceder las capacidades computacionales, lo que puede llevarse a cabo mediante el uso de un componente diseñado para limitar el número de tareas que se ejecutan a la vez.

Palabra clave:

Threads, Java, Estructuras de datos, hilos de ejecución.

[1] *Screen of Death* o Pantalla de la muerte, es el término técnico que se da a una falla catastrófica que interrumpe la ejecución de un sistema y presenta un mensaje indicando esta condición. Por lo regular, la única ruta de acción es reiniciar el dispositivo. Unas de las más conocidas son las pantallas azules del sistema operativo *Windows*.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 63 - 71

1. HILOS DE EJECUCIÓN

La programación concurrente emplea el concepto de hilos de ejecución, mismo que proviene del término en inglés *Thread*. Un hilo de ejecución es el mecanismo que permite la ejecución de una tarea en un equipo de cómputo. Dicha ejecución se realiza en distintas fases, incluyendo su preparación, la ejecución y manejo de recursos de cómputo, hasta concluir su ejecución. Todo programa requiere al menos un hilo de ejecución.

Si un programa consta de varias tareas, éstas se ejecutan una después de otra dentro del mismo hilo de ejecución. Dicho hilo[2] sigue la ejecución secuenciada de cada tarea hasta que termina. Cuando las tareas pueden ejecutarse de manera independiente, sin depender una de otra, es posible asignar uno o más hilos de ejecución paralelos.

La ejecución concurrente de tareas establece nuevos elementos que deben ser considerados por parte de los desarrolladores a fin de lograr su correcta implementación.

2. DESARROLLO CON HILOS

En el desarrollo de aplicaciones uno de los elementos más relevantes a conseguir es el uso óptimo de los recursos computacionales disponibles. Esto se logra al identificar claramente los procesamientos de información que se llevarán a cabo usando la aplicación, así como tener en cuenta los tiempos de respuesta de las distintas opciones que se implementan. Con esto en mente resulta lógico incluir el uso de las capacidades de concurrencia para su aprovechamiento. Estas capacidades permiten que más de una tarea pueda ser ejecutada a la vez en lugar de la estrategia básica de disponer de una lista de actividades que van ejecutándose una después de otra. Como ejemplo de este escenario se percibe cuando un navegador web carga una página que contiene varias imágenes. Los navegadores actuales van descargando distintas imágenes a la vez y no así una por una.

En principio, integrar el manejo de la concurrencia a una aplicación resulta muy simple mediante el empleo de bibliotecas destinadas para este propósito como lo son el paquete *java.lang.Thread* del lenguaje *Java*, o la biblioteca *thread* de *C++*. La problemática en la que los desarrolladores deberán centrarse será la del uso de los recursos computacionales que la aplicación requiere para su operación.

Ahora bien, el uso libre y sin planeación de este tipo de programación puede afectar negativamente a una aplicación en lugar de beneficiarla. En el caso de equipos muy limitados de memoria y/o poder de procesamiento, ejecutar varias tareas a la vez puede resultar en problemas de rendimiento notorios tales como la lentitud en la respuesta de un procesamiento o una falla catastrófica que bloquee al propio equipo, lo que además pueda derivar en la pérdida de información, entre otros efectos adversos.

Continuando con el ejemplo de los navegadores, en equipos muy limitados lanzar varias peticiones a la vez para la descarga de distintas imágenes puede resultar en una demanda de recursos que el propio equipo no será capaz atender apropiadamente y, en algunos casos, el sistema operativo del dispositivo al verse excedido por la carga de trabajo que tiene que atender, detendrá su operación de forma abrupta y en varios se presentará una pantalla indicando esta falla también conocida como *Pantalla de la muerte*.

[2] Se usará el término *Hilo* como sinónimo de *Hilo de Ejecución* para simplificar las explicaciones a lo largo de las siguientes secciones.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 64 - 71

3. COLA DE HILOS

Para controlar el número de hilos que se ejecutan de forma concurrente se diseñó un componente de *software*, al cual se ha denominado *Cola de hilos*. El componente incorpora para su operación dos estructuras de datos clásicas: la *Lista* y la *Cola*.

Una lista representa una colección de elementos ordenados, entendiendo en este caso el orden como el lugar o posición que ocupan dichos elementos en la estructura. El objetivo para la *Cola de hilos* será el de contener las tareas que se están ejecutando al mismo tiempo en un momento determinado, en otras palabras, los elementos de esta lista serán hilos.

Con la estructura de datos cola, se aprovecharán sus cualidades de operación donde el primer elemento que se agrega será el primero en salir. El propósito de esta estructura será la de alojar las distintas tareas que se desean ejecutar, previo a su asignación a un hilo.

Con las tareas de cada estructura claramente identificadas, la operación de la *Cola de hilos* será la siguiente:

1. Se indica el número de hilos que se pueden ejecutar a la vez.
2. Se cargarán todas las tareas a ser ejecutadas.
3. Cuando se indique la ejecución de la *Cola de hilos*:
    a. Se obtienen las primeras tareas de la cola, el mismo número que el indicado en el punto 1.
    b. Por cada tarea se crea un hilo para su ejecución y se agregan a la lista.
    c. Se llama a ejecución a cada uno de los hilos en la lista.
4. Cuando un hilo en la lista termina su ejecución:
    a. Se libera de la lista el lugar del hilo que terminó su ejecución.
    b. Si la cola de tareas no está vacía:
        i. Sacar el siguiente elemento de la cola de tareas.
        ii. Crear un nuevo hilo y asociarlo a la tarea que se obtuvo.
        iii. Ocupar el espacio libre en la lista con el nuevo hilo.
        iv. Ejecutar este hilo.
    c. Si la cola de tareas está vacía, termina la ejecución de la *Cola de hilos*.

La operación de la *Cola de hilos* que se ha definido limita el número de tareas que pueden ejecutarse a la vez, permitiendo al mismo tiempo mantener a las demás tareas en espera a su ejecución. Una vez que alguno de los hilos en la lista termine su ejecución, se verificará si no hay más tareas pendientes de ejecutar. En caso de haber una tarea, se asocia a un nuevo hilo para su ejecución. Cuando la cola de tareas no tenga elementos se termina la ejecución de la cola de hilos.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 65 - 71

4. DESARROLLO

Para la implementación de la *Cola de hilos* se eligió el lenguaje de programación *Java* a fin de aprovechar varias de sus cualidades, tales como la independencia de la plataforma, la implementación de las estructuras de datos clásicas y su biblioteca para la programación de hilos.

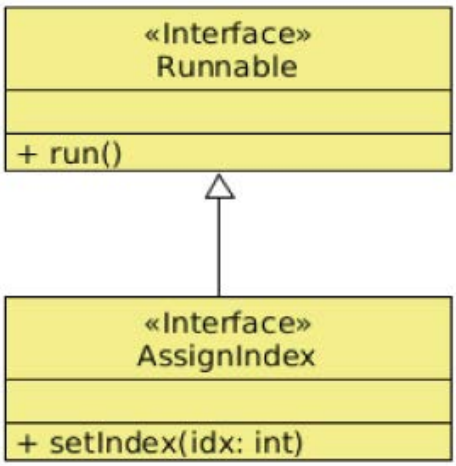
Estos elementos se usaron para el modelado de los distintos componentes de la *Cola de hilos* entre los que se pueden identificar las siguientes: las tareas que se ejecutarán, la terminación de una tarea que se ha ejecutado y la propia cola.

4.1 TAREAS A EJECUTAR

Se configuró la interfaz *AssignIndex* a fin de representar las tareas que se ejecutarán en la *Cola de hilos*. Esta interfaz hereda su funcionalidad de la interfaz *Runner* por lo que en términos muy simples cualquier tarea (definida en un clase de *Java*) solamente deberá incluir obligatoriamente los métodos *run()* y *setIndex()*; el primero, como el método que ejecutará el hilo al que finalmente se asocie, mientras que el segundo es para asignar la posición en la lista de hilos que están ejecutándose. La figura 1 muestra el diseño de la interfaz.

Figura 1

Diseño de la interfaz AssignIndex



Nota. La interfaz *Runnable* es parte de los paquetes estándar de *Java* (*Oracle Java Documentation 2022* y *Oracle Java Documentation, 2023a*). Para más información de su definición se recomienda al lector consultar la documentación oficial de *Oracle*: https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html

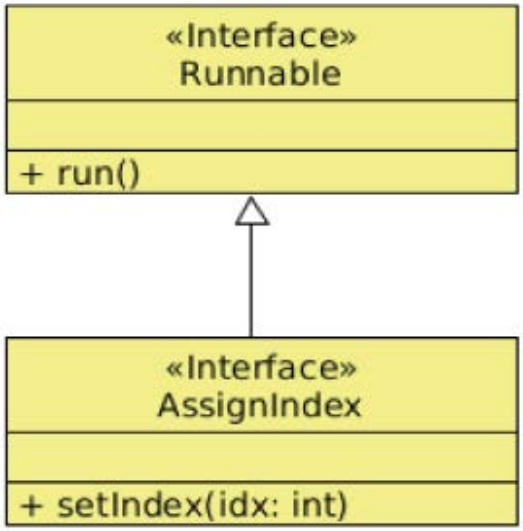
4.2 TERMINACIÓN DE UNA TAREA

La interfaz *Callback* define un solo método denominado *callback(int)*, el cual será llamado cuando una tarea concluye ya que se debe indicar a la *Cola de hilos* sobre este acontecimiento con el propósito de llevar a cabo las acciones para verificar si hay tareas por procesar, o bien, concluir con la ejecución de la *Cola de hilos*. La figura 2 muestra el diseño de la interfaz.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 66 - 71

Figura 2

Diseño de la interfaz Callback



4.3 COLA DE HILOS

La clase *ThreadQueue* representa la propia *Cola de hilos*, misma que para su operación implementa las interfaces *Queue*, del lenguaje *Java*, y *Callback* que se definió.

ThreadQueue consta de tres atributos: *queue*, *activeThreadsList* y *totalActiveThreads*, donde:

- **queue:** es la cola de tareas a ejecutar.
- **activeThreadsList:** la lista de hilos ejecutándose.
- **totalActiveThreads:** número de tareas que se pueden ejecutar a la vez.

Asimismo, cuenta con los siguientes métodos:

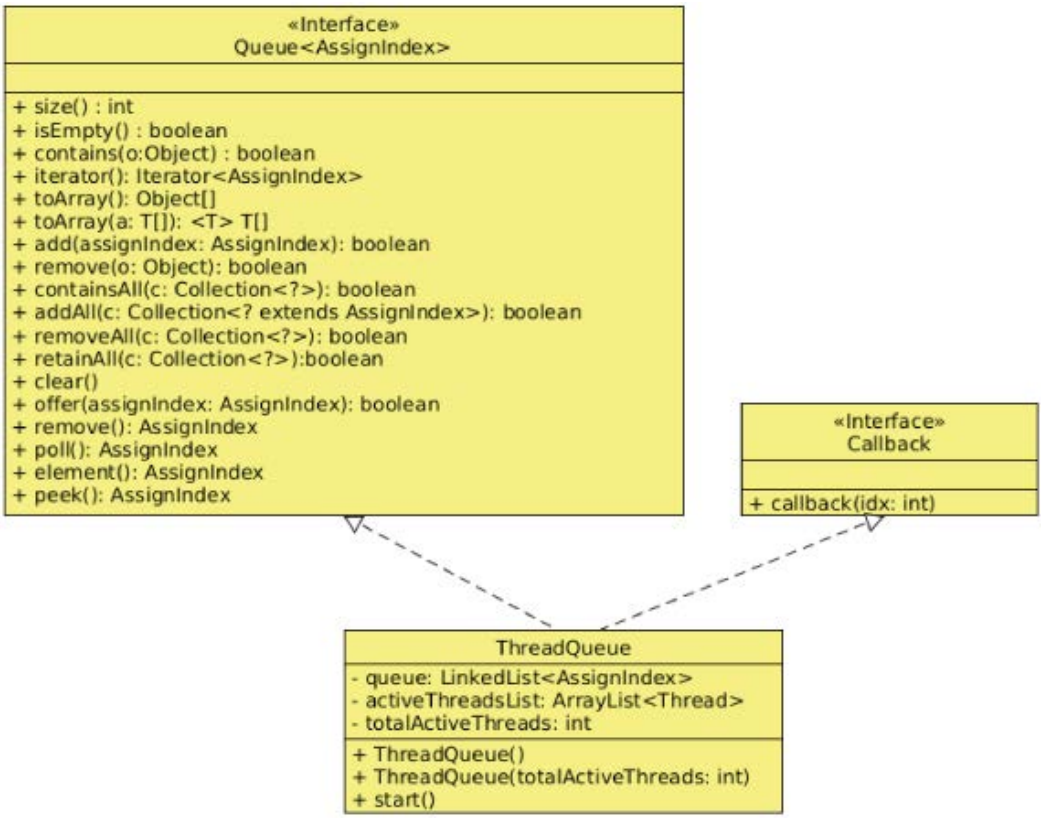
- **ThreadQueue():** constructor por omisión.
- **ThreadQueue(int):** constructor que indica el número de hilos activos.
- **start():** inicia la ejecución de la *Cola de hilos*.
- **callback(int):** método que cada tarea ejecuta cuando concluye su propia ejecución.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 67 - 71

La figura 3 muestra el diseño completo de la clase *ThreadQueue*.

Figura 3

Diseño de la clase ThreadQueue



Nota. La interfaz *Queue* es parte de los paquetes estándar de *Java* (Friesen, J. 2015, Liang, Y., 2021 y *Oracle Java Documentation. 2023b*). Para más información de su definición se recomienda al lector consultar la documentación oficial de *Oracle*: https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

5. CREANDO TAREAS PARA LA COLA DE HILOS

Para que una tarea pueda agregarse y posteriormente ejecutarse dentro del control de la *Cola de hilos* como se ha mencionado anteriormente, es necesario que implemente la interfaz *AssignIndex*. En otras palabras, la clase que represente una tarea deberá incluir el método *run()* que tendrá las operaciones que se ejecutarán mediante un hilo que se le asignará dentro de la *Cola de hilos*. El segundo método que debe programarse es *setIndex()*, que será invocado cuando la tarea pase a la lista de hilos que se están ejecutando, entonces el *ThreadQueue* asociado indicará con este método el lugar que ocupará para su control.

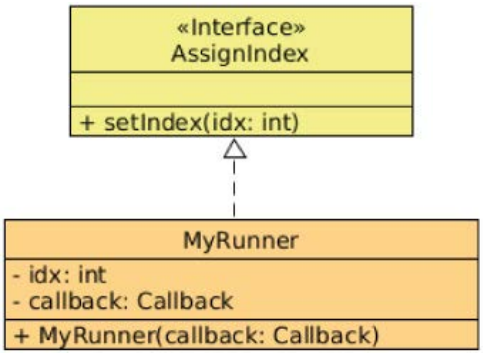
Cuando una tarea concluye su ejecución, deberá invocar al método *callback()* de la *Cola de hilos*, por lo que se debe tener una referencia desde la tarea. El parámetro que se envía a este método deberá ser la misma posición asignada cuando la tarea se agregó a la lista de hilos en ejecución.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 68 - 71

El diseño de una tarea básica puede apreciarse en la figura 4.

Figura 4

Diseño de una tarea



La clase *MyRunner* puede ser programada para la ejecución de una tarea particular permitiendo que su diseño atienda las especificaciones de operación que se definan en su contexto de operación.

Como resultado del diseño una misma *Cola de hilos* puede ejecutar tareas muy distintas como resultado de la relación de funcionalidad mediante el uso de interfaces del lenguaje *Java*.

Retomando el ejemplo del navegador, para tener una tarea de descarga de imágenes se debe crear una clase para este propósito. Al igual que en el caso de *MyRunner* la nueva clase debe indicar la implementación de la interfaz *AssignIndex*, permitiendo la asignación de su posición en la *Cola de hilos* y al terminar la descarga indicar dicha posición.

6. USO DE LA COLA DE HILOS

Para hacer uso de la *Cola de hilos* solo se requiere tres pasos: crear un *ThreadQueue* a la programación, agregar las tareas que se quieren ejecutar al objeto *ThreadQueue* y finalmente iniciar la ejecución de la cola. La figura 5 ejemplifica estas tres acciones.

Figura 5

Diseño de una tarea

```
1  ThreadQueue threadQueue = new ThreadQueue(5);
2
3  threadQueue.add(new MyRunner(threadQueue));
4  threadQueue.add(new MyRunner(threadQueue));
5  threadQueue.add(new MyRunner(threadQueue));
6  threadQueue.add(new MyRunner(threadQueue));
7
8  threadQueue.start();
```

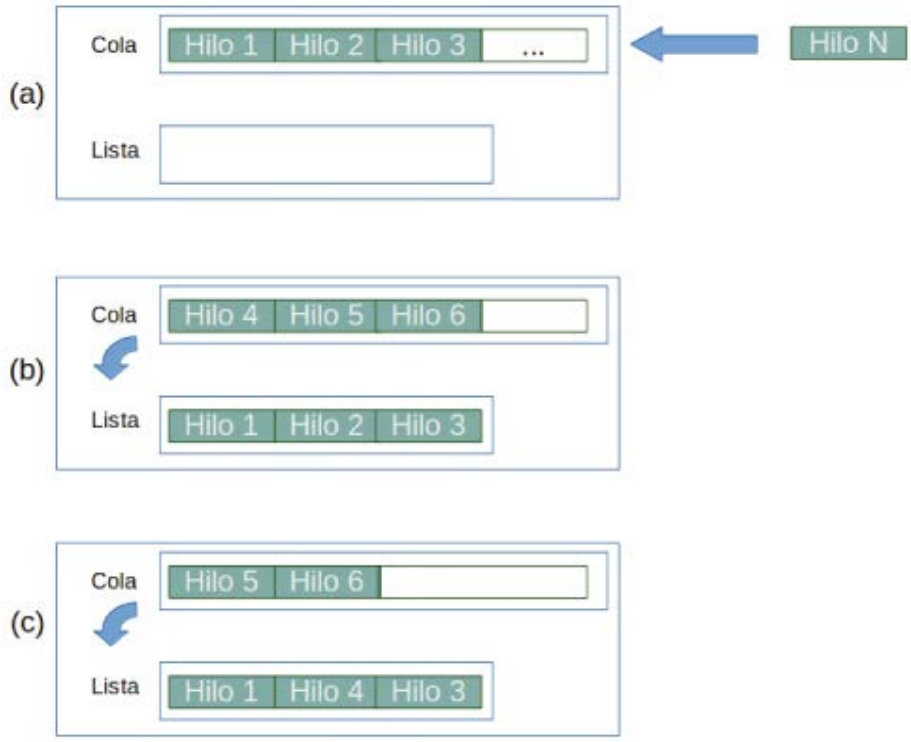
https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 69 - 71

En la línea 1 se indica la creación de una *Cola de hilos* que en este caso tendrá a lo más cinco hilos ejecutándose a la vez. Las líneas 3 a 6 muestran la carga de tareas, en este caso mediante la creación de objetos *MyRunner* según su especificación. La línea 8 se llama a la ejecución de la *Cola de hilos*, misma que comenzará la ejecución de las tareas indicadas y conforme vayan concluyendo estas tareas se irán ejecutando las pendientes.

La Figura 6 resume la operación de la *Cola de hilos* que se ha diseñado. En primer lugar los hilos que se van a ejecutar se cargan en la cola interna (a). Una vez completada la carga, los primeros hilos se sacan de la cola interna y pasan a la lista donde éstos comienzan su ejecución (b). Cuando una tarea en la lista completa su ejecución, su lugar será ocupado por el siguiente hilo en la cola interna (c), repitiendo esta acción hasta que ya no haya más hilos en espera de ejecución.

Figura 6

Operación de la Cola de hilos



7. CONCLUSIONES

El uso de la *Cola de hilos* que se diseñó previene la demanda excesiva de recursos de cómputo en aplicaciones donde existen diversas tareas ejecutándose en paralelo. Su uso en una aplicación proporciona diversos beneficios tales como evitar el bloqueo de la aplicación, controlar la cantidad de tareas concurrentes que se ejecutan, impedir la lentitud del equipo o prevenir la interrupción catastrófica de equipos con recursos limitados.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 70 - 71

El diseño presentado se puede mejorar según el contexto donde se necesite, por ejemplo, en lugar de una cola simple se puede complementar su programación para implementar una cola con prioridad donde las tareas se ejecutarán con base en un criterio de precedencia cada vez que una nueva tarea se ejecute; o bien, extender la funcionalidad incluyendo métodos para pausar la cola o incluir mejoras sobre el proceso de excepciones.

Con el uso de esta nueva estructura se alcanzan varios beneficios ya que contribuye a la ejecución ordenada de tareas en un equipo de cómputo, el número de hilos que se pueden ejecutar al mismo tiempo se limita desde la construcción de la *Cola de hilos* y conforme éstas van concluyendo su ejecución, su lugar será ocupado por una nueva hasta que no haya más.

REFERENCIAS BIBLIOGRÁFICAS

Liang, Y. (2021). *Introduction to Java™ Programming and Data Structures*. (12a ed.). USA: Pearson

Friesen, J. (2015). *Java Threads and the Concurrency Utilities*. (1a. ed.). USA: Apress

Oracle Java Documentation (2022). *Concurrency*. Recuperado el 21 de octubre de 2023, de https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Oracle Java Documentation (2023a). *Interface Runnable*. Recuperado el 21 de octubre de 2023, de https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html

Oracle Java Documentation (2023b). *Interface Queue*. Recuperado el 21 de octubre de 2023, de https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

ANEXO A

Una implementación libre en el lenguaje de programación *Java* de la *Cola de hilos* presentada, se ubica en el repositorio institucional de la DGTIC, la cual puede accederse en la siguiente dirección:

https://kwira-kaab.unam.mx/badboy/threadqueue

Este desarrollo cuenta con los siguientes archivos:

- *App.java*, archivo principal de ejecución.
- *AssignIndex.java*, interfaz para las tareas que se ejecutarán.
- *Callback.java*, interfaz de la llamada de vuelta para la conclusión de una tarea.
- *ThreadQueue.java*, implementación de la *Cola de hilos*.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.13 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 71 - 71

Se incluyen además dos ejemplos de implementación de la interfaz *AssignIndex,* las cuales son:

- *MyRunner.java*, muestra una tarea de consumo variable de tiempo.
- *DownloadRunner.java*, muestra una tarea destinada a la descarga de imágenes.

Para la programación del componente se usaron las siguientes tecnologías:

- *Eclipse 2023-09 (4.29.0).*
- *JDK 20.*

La clase *DownloadRunner* usa la biblioteca *Apache Commons IO™* para simplificar la descarga de archivos que se encuentran disponibles desde algún sitio Web usando su *URL* mediante la clase *FileUtils*.

Existen otras alternativas a esta tecnología como es el uso del paquete *Java NOI*, para la construcción de componentes a bajo nivel, o bien, se pueden emplear otras bibliotecas como el caso de *AsyncHttpClient (AHC)* de *Netty*.