Tema 9

Hilos

NSOperationQueue

- Un **thread** o hilo es una tarea que puede ejecutarse de forma paralela o concurrente a otra. A la hora de implementar operaciones costosas que deben ejecutarse de manera simultanea es necesario el uso de threads.

- El uso de threads supone un problema a la hora de crear una solución escalable ya que hay que decidir cuantos hilos crear según la situación, y esto puede cambiar de manera dinámica. Además, es necesario crear y mantener dichos hilos.

- Para solucionar estos problemas, se puede hacer uso de las colas de operaciones, implementadas por el objeto **NSOperationQueue**.

NSOperationQueue

- La clase **NSOperationQueue** se encarga de regular la ejecución del conjunto de operaciones que se ha puesto en cola.

- A diferencia de otras colas, que se ejecutan siguiendo un mecanismo FIFO (First in, first out), NSOperationQueue tiene en cuenta otros factores, como por ejemplo, la dependencia entre operaciones.

- Las operaciones que se encolan son instancias de la clase **NSOperation**, y se realiza mediante el método **addOperation:**.

NSOperation

- La clase **NSOperation** encapsula el código que debe ejecutar.
- Normalmente no se hace uso de la clase NSOperation, en su lugar se emplean subclases, ya sean propias o ya existentes como **NSInvocationOperation** o **NSBlockOperation.**
- Mediante los métodos **addDependency:** y **removeDependency:** se pueden añadir dependencias entre operaciones, de forma que unas se ejecuten únicamente al terminar otras.

NSOperation

- Por otro lado, podemos definir un bloque para que se ejecute al finalizar la operación. Es posible realizar esto mediante la propiedad **setCompletionBlock:**.

- Para priorizar operaciones en una misma cola, se emplea **setQueuePriority:**, y podemos acceder a su prioridad mediante **queuePriority.**

NSInvocationOperation

- **NSInvocationOperation** es una subclase de NSOperation.
- Se emplea mediante el uso de un selector.
- Para la creación se utiliza el método **initWithTarget:selector:object:** mediante el cual indicamos el selector y en caso de ser necesario un objeto.
- Al igual que con los objetos NSOperation se añade a la cola mediante **addOperation:**.

NSBlockOperation

- **NSBlockOperation** es una subclase de NSOperation.
- A diferencia de NSInvocationOperation se hace uso de bloques, por lo que el código no es necesario tenerlo en un método separado.
- Para su creación se emplea el método **blockOperationWithBlock:**.
- Es posible añadir nuevos bloques a un objeto NSBlockOperation ya creado haciendo uso del método **addExecutionBlock:**, de esta forma, se ejecutan de forma concurrente los distintos bloques.
- Al igual que con NSInvocationOperation debemos añadir los objetos NSBlockOperation a la cola mediante **addOperation:**.

Subclases de NSOperation

- Para crear una subclase de NSOperation tenemos dos opciones, que la operación sea **concurrente** o **no concurrente**.

- Para las operaciones no concurrentes basta con implementar el método **main**.

- Para las operaciones concurrentes es posible implementar los métodos **start**, **isConcurrent**, **isExecuting** y **isFinished.**

Subclases de NSOperation

- Para hacer uso de estas subclases mediante las NSOperationQueue es más sencillo emplear operaciones no concurrentes, de forma que simplemente debemos implementar el método main y añadir la operación a la cola.

- Si en cambio queremos ejecutar las operaciones de manera manual, suele ser aconsejable emplear operaciones concurrentes aunque es necesario monitorizar el estado e informar de cambios mediante notificaciones.

Subclases de NSOperation

- Tanto el método **main** como el método **start** son los encargados de realizar la tarea.

- El método **isConcurrent** indica si la operación se realiza de forma asíncrona o no.

- **isExecuting** informa del estado de la operación, es decir, si se está ejecutando o no.

- Por último, **isFinished** indica si la operación ha terminado de ejecutarse.