Si un message al procesarse

* Devuelve un error que no logro conectarse a la base de datos
* O dio un timeout en la base de datos
* O aborto en medio del proceso de actualizar la base de datos

Entonces

* Siempre es como que se proceso el mensaje por message queue
* Lo recibio y lo proceso, deseamos que lo vuelva a procesar
* Para ello enviamos un Nuevo mensaje con la misma informacion, se va a cola y se vuelve a intentar

In order to make sure a message is never lost, RabbitMQ supports message acknowledgments. An ack(nowledgement) is sent back from the consumer to tell RabbitMQ that a particular message has been received, processed and that RabbitMQ is free to delete it.

If a consumer dies (its channel is closed, connection is closed, or TCP connection is lost) without sending an ack, RabbitMQ will understand that a message wasn't processed fully and will re-queue it. If there are other consumers online at the same time, it will then quickly redeliver it to another consumer. That way you can be sure that no message is lost, even if the workers occasionally die.

There aren't any message timeouts; RabbitMQ will redeliver the message when the consumer dies. It's fine even if processing a message takes a very, very long time.

Message acknowledgments are turned on by default. In previous examples we explicitly turned them off by setting the noAck ("no manual acks") parameter to true. It's time to remove this flag and send a proper acknowledgment from the worker, once we're done with a task.

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body;

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] Received {0}", message);

int dots = message.Split('.').Length - 1;

Thread.Sleep(dots \* 1000);

Console.WriteLine(" [x] Done");

channel.BasicAck(deliveryTag: ea.DeliveryTag, multiple: false);

};

channel.BasicConsume(queue: "task\_queue", noAck: false, consumer: consumer);