Integration Exercise

Implement a PDF converter application (e.g. a web app or C# console app) that lets customers upload a text document. The application converts the document into a PDF (or simulates the conversion) and sends an email back to the customer (or simulates this).

Illustration of the Use case

1. Customer uploads a text document
2. Your app converts the document into a PDF
3. Your app sends the PDF back to the customer in email

What Integration Style to Choose?

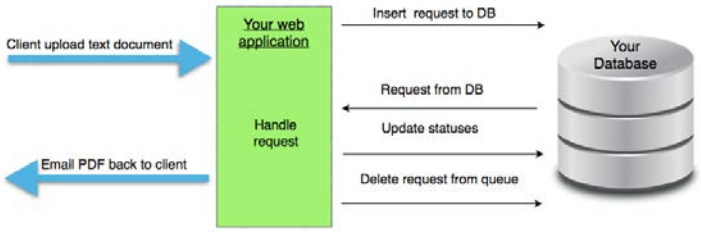
Is a database an option?

Should we consider using a database as a queue? We would need to put each task into database table, e.g.

```
INSERT INTO pdf_job (name, status, email) VALUES ("Bible", "NEW", "mymail@mail.com");
```

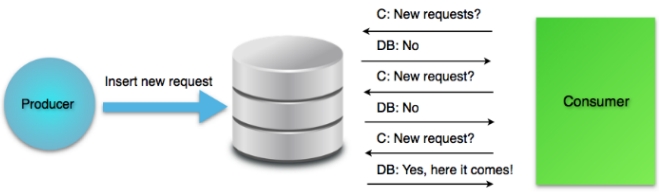
We must write code to:

- insert the new requests into the database
- take an input from the database, perhaps change a status column, with values such as "NEW" and "PROCESSING"
- handle the request
- update the database status column to "FINISHED" or removes the request from the table.



If we let the database act as a queue, the application has to ask the server over and over again for the latest queued requests.

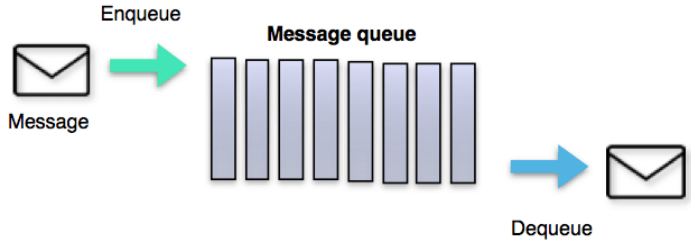
Messages that have already been consumed need to be filtered out. E.g:




```
SELECT * FROM pdf_job_queue WHERE handled = false ORDER BY date;
```

How about using a message queue!

Message queues are built to handle this kind of scenario, and it is easy to queue or dequeue messages from the queue.



The queue can receive a PDF processing request and let the consuming application do all the processing, and pop the message of the queue.

Messaging options

For this exercise you have two options.

MSMQ

You can use MSMQ.

MSMQ is a messaging queuing technology based on the MSMQ protocol.

Get inspiration from the demo code shown in class (.NET solution named *MSMQDemo)*.

RabbitMQ (AMQP)

You can use RabbitMQ.

RabbitMQ is a message broker that uses the AMQP protocol. It comes in different flavors:

- as a cloud service
- with Docker container
- your own server (the school has a central server that we will use later on)

I suggest you start with RabbitMQ as a service (CloudAMQP): https://www.cloudamqp.com/

Getting started with a CloudAMQP service: https://www.cloudamqp.com/docs/index.html

You can choose from a whole range of programming languages, see RabbitMQ tutorials here (use the first one called “Hello World”.

You can get inspiration from a small Java demo program that uses a RabbitMQ cloud service: https://github.com/Tine-m/Rabbit-cloud