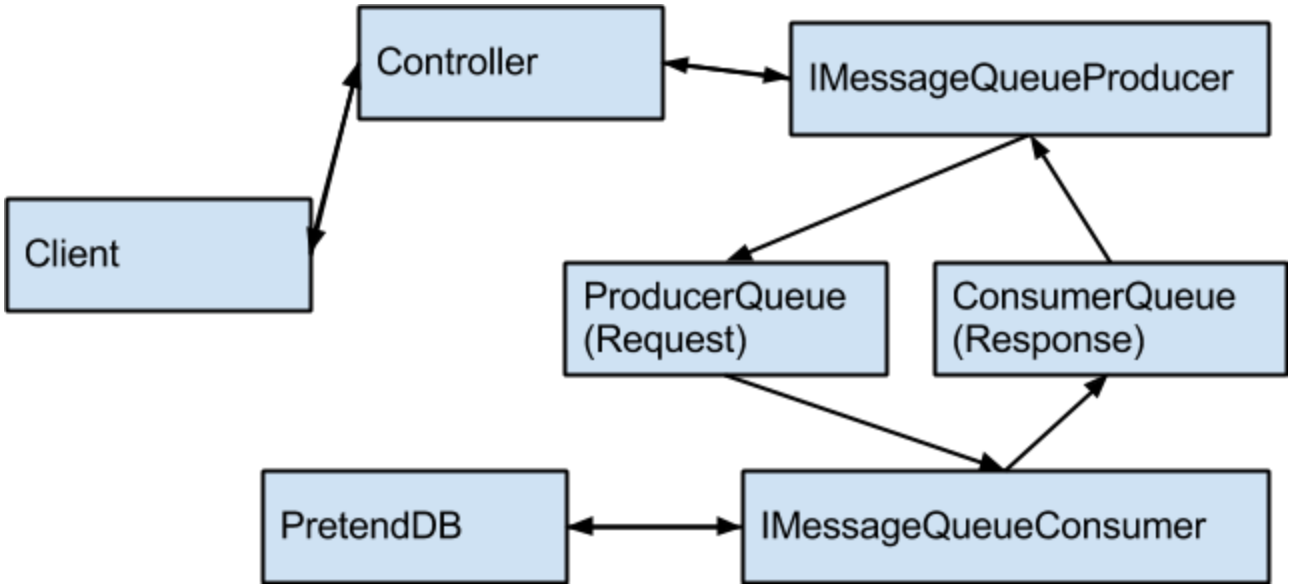
Architecture:



Above is my attempt at visualizing our queue architecture. First it starts with the client contacting the server for a web page. For this example lets say the client is accessing the courses index. The courses controller then calls “GetAll” from the IMessageQueueProducer. Each controller creates an IMessageQueueProducer for each model type it will need to query the database for. Each IMessageQueueProducer connects to (or creates if the queue does not exist) two queues.One for sending request to the database (the Producer Queue) and one for receiving responses from the database (the Consumer Queue). The queues are named according to the model type given to the the IMessageQueueProducer to ensure it will send message to a queue the IMessageQueueConsumer is watching. Continuing with the example, the IMessageQueueProduer will create a “GetAll” request and tag the request with a GUID. The IMessageQueueProducer then waits until a message with the same GUID appears in the Consumer Queue. Meanwhile the IMessageQueueConsumer is waiting to process the next available message that comes into the Producer Queue. This is set up at the start of the program. There is one consumer for each Model. When a message does come in the IMessageQueueConsumer raises an event to allow the PretendDB to process the data. In our example the IMessageQueueConsumer would receive the “GetAll” request from the Producer Queue and raise the GetAll event. The PretendDB then retrieves all courses from the database and returns that list to the IMessageQueueConsumer to place into a response. The IMessageQueueConsumer then sends a response with the same GUID to the Consumer Queue. Finally the producer finds this message and returns the results to the controller.
One final important note about the producer and consumer is that they can only handle basic CRUD operations. Any processing such as sorting or queries are handled by the controller.

Topics:

Bandwidth:
Most of our message are very small as they consist of a few strings and a Model. These small

message will not add too much to the total Bandwidth consumed by each user. However, when a user access the courses index, we send over 5000 course through the queue. This message take up about 4mb. If 100 user were to access the courses index simultaneously, which is very possible since that is our default page, the total bandwidth consumption would be around 400mb.