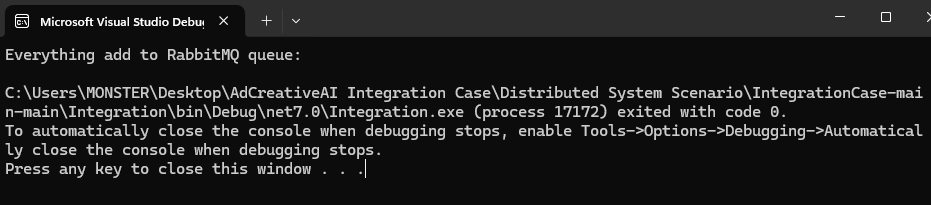
For the distributed system scenario, I queued data insertion requests. I used RabbitMQ for this process. In a scenario where the application runs on more than one server, requests will be sorted in a queue order, even if parallel or async. Let's then assume that a Listener application runs to process this queued data. This application will extract the data from the queue, read it and process it. In the Listener application, the necessary checks (IsExists) will be made and the addition will be made by referring to the Repository application where database storage operations are performed. Since the console application is used in this case, no information is transferred to the user interface by the consumer during the dequeue process. A background service could be run inside the console application and the consumer could be embedded into it. But since it offers both code readability and modularity better, I performed consumer operations through the Listener console application in the Solution (Therefore, the Listener and Integration application should be run together). In a real life scenario, communication between relevant applications can most likely be achieved via API/service. For example, if we were to consider a web application, we could return the information to the user after consumer transactions with SignalR.

**Integration App**



**Listener App**

