Знакомство с RabbitMQ

Для успешного выполнения лабораторной работы необходимо:

1. Установить и настроить RabbitMQ на локальном компьютере
2. Познакомиться с Web-интерфейсом брокера сообщений
3. Познакомиться с основными обменниками(Exchanges) RabbitMQ
   * Default Exchange
   * Direct Exchange
   * Topic Exchange
   * Fanout Exchange

Введение

RabbitMQ - это брокер сообщений, позволяющий отправлять и принимать сообщения в различных компонентах вашей системы. RabbitMQ использует свой собственный протокол AMQP который работает поверх TCP/IP. Официальный сайт, на котором есть соответствующая документация и примеры кода на разных языках находится по адресу https://www.rabbitmq.com/

1. Установка и настройка RabbitMQ на локальном компьютере

Для установки брокера сообщений на локальном компьютере с использованием системы Windows необходимо установить фреймворк Erlang OTP(otp\_win64\_25.1.2.exe) и сам RabbitMQ(rabbitmq-server-3.11.2.exe). Так же необходимо учитывать, что версии могут отличаться.  
По умолчанию наш сервер брокера сообщений установится в папку *C:\Program Files\RabbitMQ Server.*

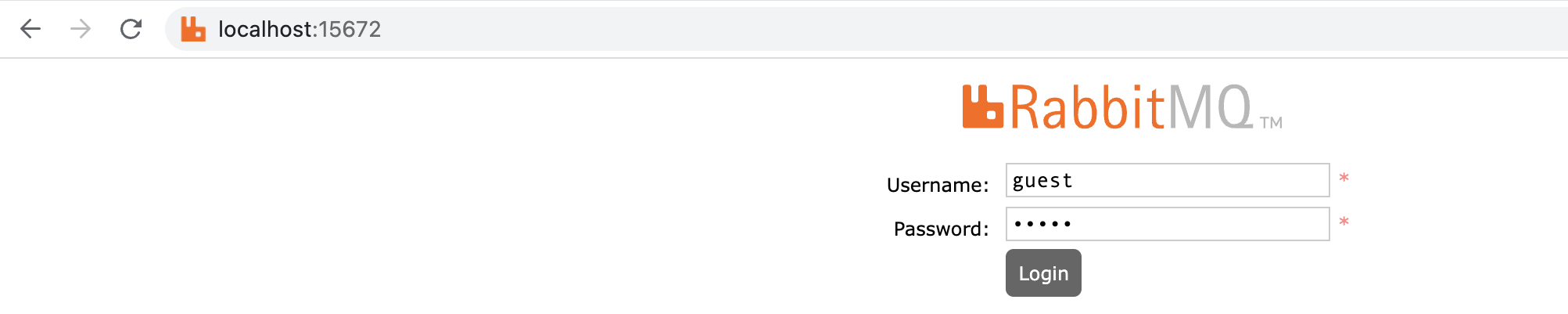
Если все установилось корректно и выше указанная папка существует и даже имеет что то внутри, попробуем включить Web-интерфейс для нашего RabbitMQ. Для этого откроем консоль (ОТ ИМЕНИ АДМИНИСТРАТОРА) и перейдем по пути *C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.11.2\sbin*

Далее выполним следующие команды:

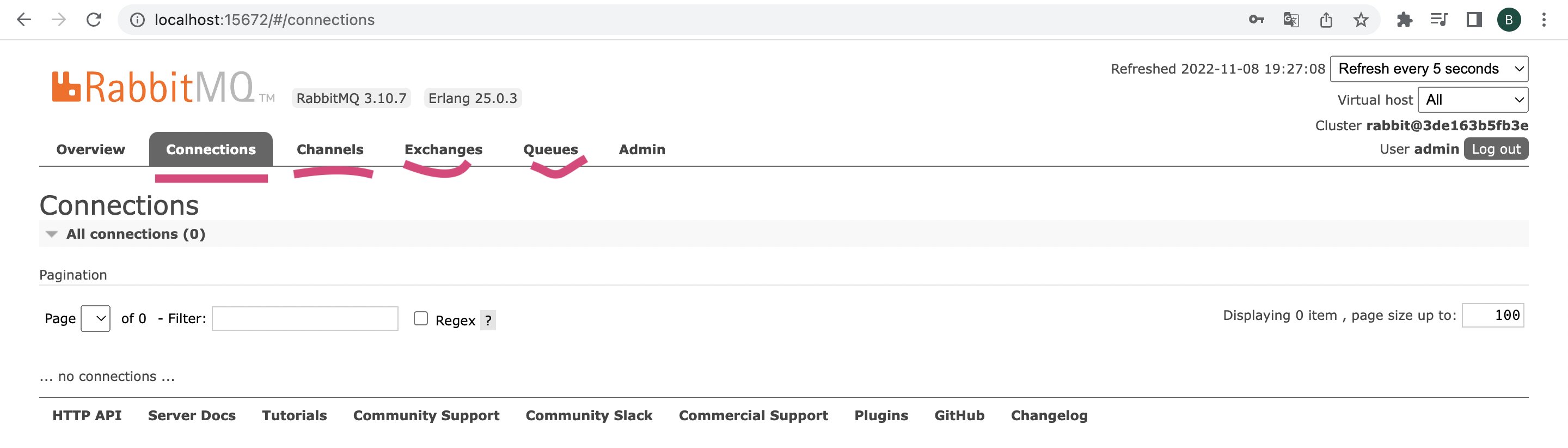
И перезапустим наш сервер RabbitMQ:

Well done, перейдем в браузере по пути <http://localhost:15672> и убедимся в работоспособности сайта. Стандартные параметры для входа:

**Username: guest**

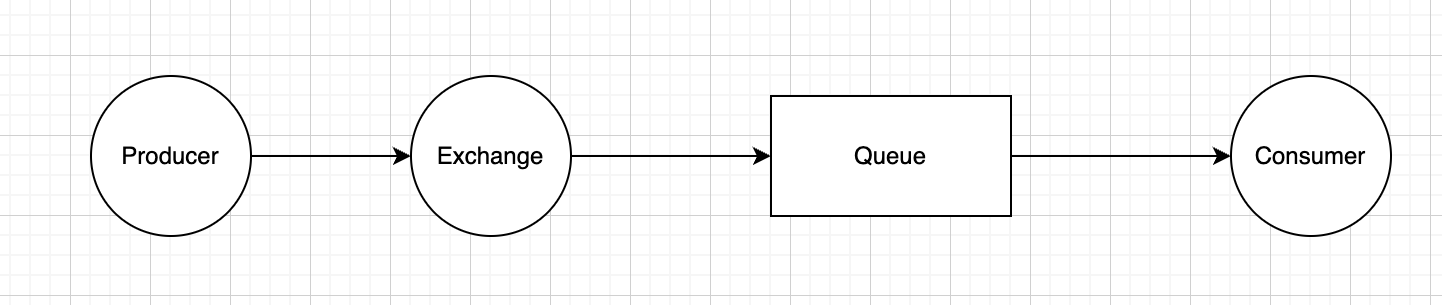
**Password: guest**

2) Знакомство с Web-интерфейсом брокера сообщений



Тут мы видим основные вкладки для управления брокером: Connections, Channels, Exhanges, Queues. В рамках самостоятельной работы попробовать создать очередь и в нее отправить сообщение через DefaultExchange.

3) Познакомиться с основными обменниками(Exchanges) RabbitMQ

Принято называть отправителя Producer, а получателя Consumer. Как таковой возможности отправлять сообщения непосредственно в очередь RabbitMQ не предоставляет. Механизм устроен таким образом, что Producer отправляет сообщение в обменник Exchange, а далее уже он выбирает в какую очередь положить сообщение. Тем временем Consumer подписывается на получение сообщений из очереди и забирает их при непосредственном получении.

Чтобы подключиться к нашему серверу RabbitMQ необходимо установить nuget-пакет RabbitMQ.Client. Это официальная библиотека брокера сообщений, с которой мы будем работать.

Укажем параметры для подключения к брокеру:

*var factory = new ConnectionFactory*

*{*

*HostName = "localhost",*

*UserName = "guest",*

*Password = "guest",*

*VirtualHost = "/",*

*Port = 5672*

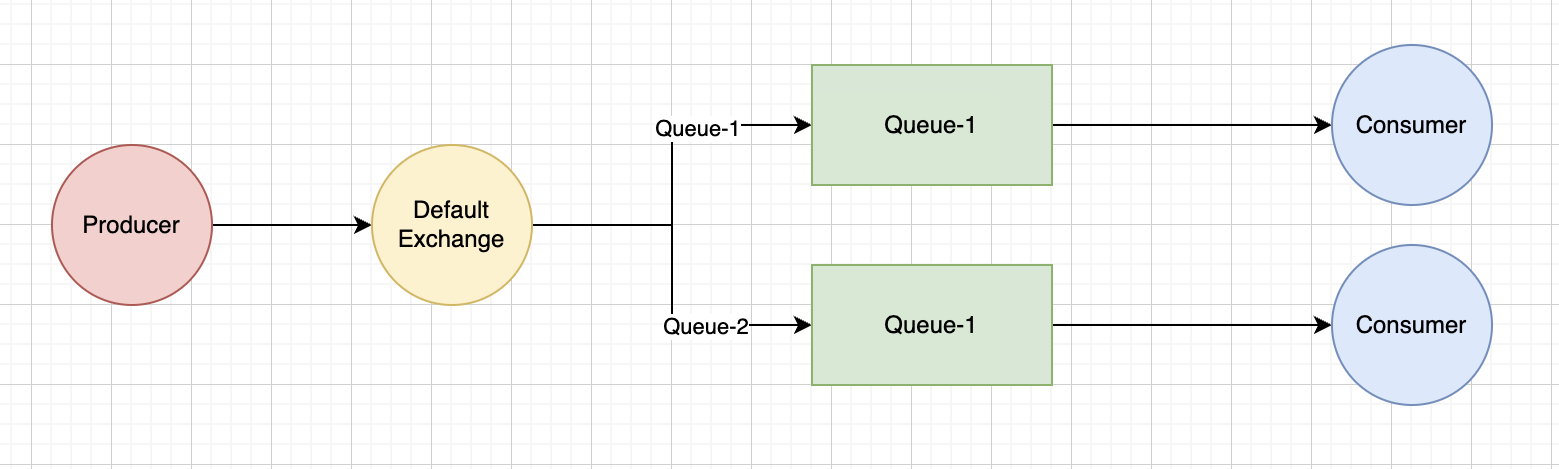
*};*

Теперь создадим подключение и канал с которым далее будем работать:

*using var connection = factory.CreateConnection();*

*using var channel = connection.CreateModel();*

Default Exchange

Самый простой обменник предоставляет возможность отправлять сообщения по названию очереди

Как мы видим, отправителю достаточно указать название очереди, в которую мы хотим отправить сообщение, а получателю только подписаться на нее.

Код Producer:

*channel.QueueDeclare("my-queue-name", false, false, false, null);*

*var message = "Hello from RabbitMQ";*

*var body = System.Text.Encoding.UTF8.GetBytes(message);*

*channel.BasicPublish("", "my-queue-name", null, body);*

Код Consumer:

*channel.QueueDeclare("my-queue-name", false, false, false, null);*

*var consumer = new EventingBasicConsumer(channel);*

*consumer.Received += (sender, e) =>*

*{*

*var body = e.Body;*

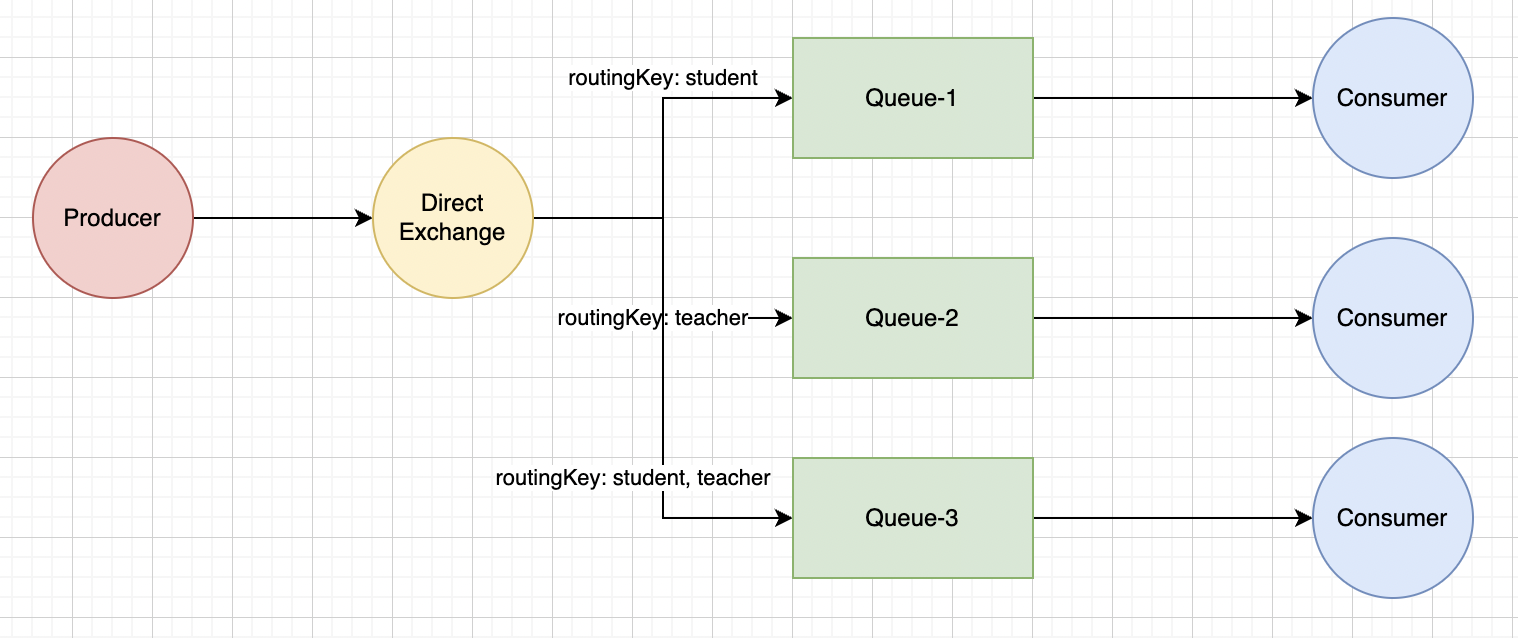
*var message = Encoding.UTF8.GetString(body.ToArray());*

*};*

*channel.BasicConsume("my-queue-name", true, consumer);*

Обратите внимание, что сообщения передаются в байтах, поэтому перед отправкой и получением необходима конвертация.

Direct Exchange

Для следующего типа обменника необходимо ввести такое понятие как привязки(bindings). Они служат для указания направлений наших сообщений. В следующей схеме мы имеем 3 очереди, которые связаны с обменником по значениям **routingKey**.

При отправки сообщения по значению *routingKey=student* мы попадем в очередь с названием **Queue-1**, соответсвенно по значению *routingKey=teacher* мы попадем в очередь **Queue-2**. Так же очередь может быть привязана к нескольким значениям *routingKey* как например очередь **Queue-3**.

Код Producer:

*channel.ExchangeDeclare("direct-exchange-name", ExchangeType.Direct);*

*var message = "Hello from RabbitMQ";*

*var body = System.Text.Encoding.UTF8.GetBytes(message);*

*channel.BasicPublish("direct-exchange-name", "student", null, body);*

Тут мы объявляем обменник с именем direct-exchange-name и указываем его тип Direct. Как и в прошлом примере отправляем наше сообщение, но вместо названия очереди указываем routingKey student

Код Consumer:

*channel.ExchangeDeclare("direct-exchange-name", ExchangeType.Direct);*

*var queueName = channel.QueueDeclare().QueueName;*

*channel.QueueBind(queue: queueName,*

*exchange: "direct-exchange-name",*

*routingKey: "student");*

*var consumer = new EventingBasicConsumer(channel);*

*consumer.Received += (sender, e) =>*

*{*

*var body = e.Body;*

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

*};*

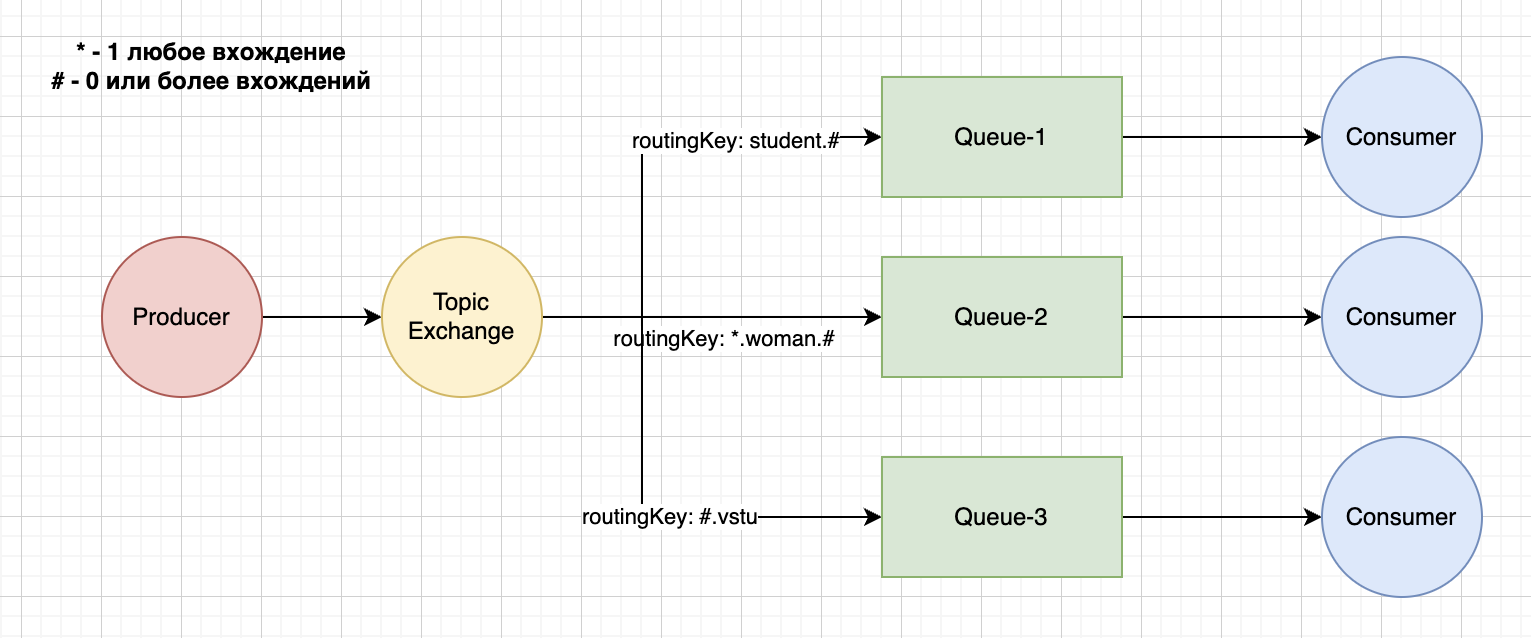
*channel.BasicConsume(queueName, true, consumer);*

Для надежности конструкции объявим еще раз наш обменник вызовом метода ExhcnageDeclare(). Если обменника с таким названием еще не существует, он будет создан, в противном случае ничего не произойдет. Далее мы создаем очередь с рандомным названием в методе QueueDeclare() и связываем наш обменник с очередью по ключу student в методе QueueBind()

Topic Exchange

Следующий тип обменника добавляет правила для маршрутизации по очередям. RoutingKey содержит последовательность символов, разделенные точкой и добавляется функционал правил:

**\* - 1 любое вхождение слова**

**# - 0 или более вхождения слова**

Код для отправителя и получателя идентичен прошлому типу обменника, за исключением параметра routingKey. Теперь при отправки сообщения по ключу student.man.vstu оно будет попадать в **Queue-1** и **Queue-3**

Код Producer:

*var exchangeName = "topic-exchange-name";*

*channel.ExchangeDeclare(exchangeName, ExchangeType.Topic);*

*var message = "Hello from RabbitMQ";*

*var body = System.Text.Encoding.UTF8.GetBytes(message);*

*channel.BasicPublish(exchangeName, "student.man.vstu", null, body);*

Код Consumer:

*var exchangeName = "topic-exchange-name";*

*channel.ExchangeDeclare(exchangeName, ExchangeType.Topic);*

*var queueStudentName = channel.QueueDeclare().QueueName;*

*channel.QueueBind(queue: queueStudentName,*

*exchange: exchangeName,*

*routingKey: "student.#");*

*var consumerStudent = new EventingBasicConsumer(channel);*

*consumerStudent.Received += (sender, e) =>*

*{*

*var body = e.Body;*

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

*};*

*channel.BasicConsume(queueStudentName, true, consumerStudent);*

*var queueVSTUName = channel.QueueDeclare().QueueName;*

*channel.QueueBind(queue: queueVSTUName,*

*exchange: exchangeName,*

*routingKey: "#.vstu");*

*var consumerVSTU = new EventingBasicConsumer(channel);*

*consumerVSTU.Received += (sender, e) =>*

*{*

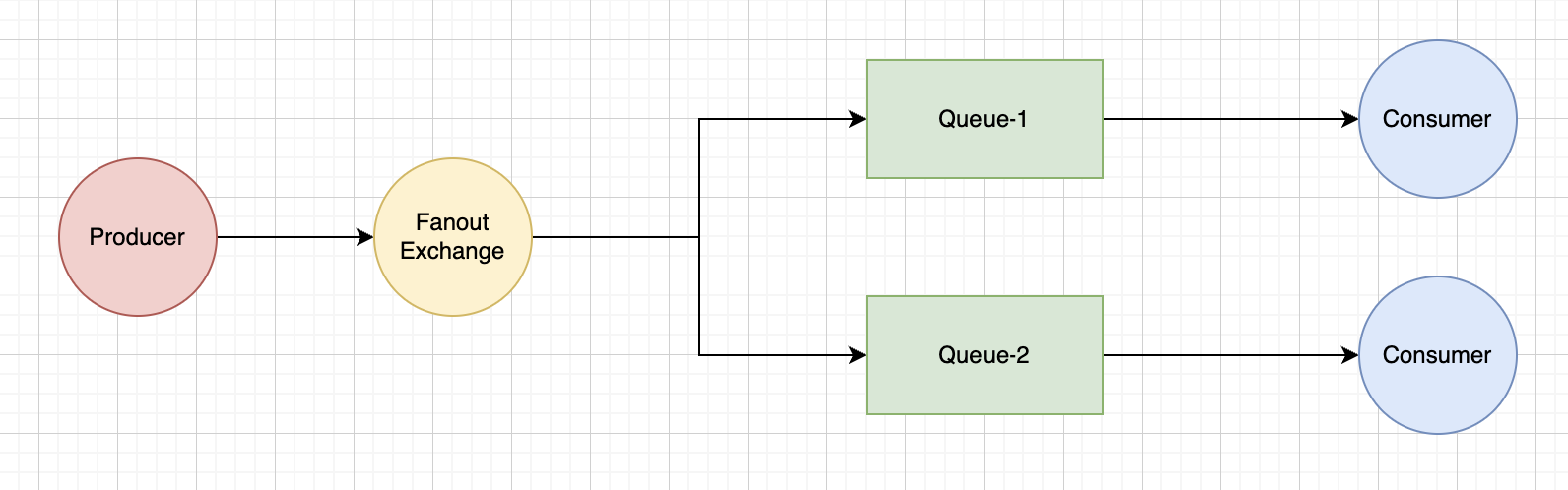
*var body = e.Body;*

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

*};*

*channel.BasicConsume(queueVSTUName, true, consumerVSTU);*

Fanout Exchange

Следующий тип обменника достаточно прост, он не использует **routingKey** для доставки сообщений, а отправляет их во все очереди, которые к нему привязаны.

Код Producer:

*channel.ExchangeDeclare("fanout-exchange-name", ExchangeType.Fanout);*

*var message = "Hello from RabbitMQ";*

*var body = System.Text.Encoding.UTF8.GetBytes(message);*

*channel.BasicPublish("fanout-exchange-name", "", null, body);*

Код Consumer:

*channel.ExchangeDeclare("fanout-exchange-name", ExchangeType.Fanout);*

*var queueFirstName = channel.QueueDeclare().QueueName;*

*channel.QueueBind(queue: queueFirstName,*

*exchange: "fanout-exchange-name",*

*routingKey: "");*

*var consumerFirst = new EventingBasicConsumer(channel);*

*consumerFirst.Received += (sender, e) =>*

*{*

*var body = e.Body;*

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

*};*

*channel.BasicConsume(queueFirstName, true, consumerFirst);*