Техническое задание

Модуль Node.js для работы с RabbitMQ

Краткое описание

Написать модуль-обертку на Node.js над NPM-пакетами amqplib и amqplib-auto-recovery.

Главная и основная задача

Написать такой функционал, чтобы при массовом помещении сообщений в RabbitMQ при нештатных ситуациях (когда сообщение не удалось поместить в очередь по какой-то причине) передаваемые данные не потерялись.

Схематичное описание

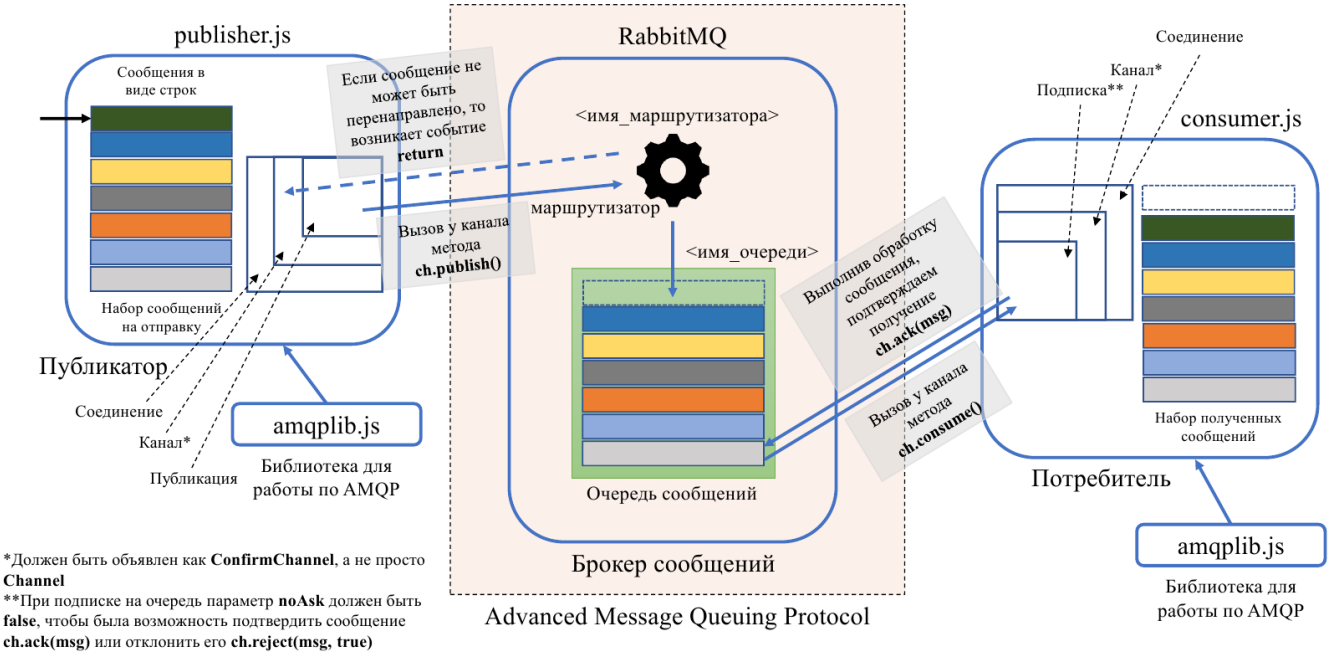

Рисунок 1 – Тестирование RabbitMQ с подтверждениями Публикатору и Потребителем

Подробное описание

Модуль должен представлять собой класс, в котором будут присутствовать методы:

1. Установка соединения по AMQP (connect)
2. Создание канала в режиме подтверждения (create_confirm_channel)
3. Утверждение очереди (assert_queue)
4. Публикация сообщения (publish)
5. Подписка на очередь (consume)

«Рыба» может выглядеть примерно так:

Листинг 1 – Набросок модуля для работы с Node.js

```
/**
 * Работа с протоколом AMQP
 *
 */
'use strict';

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                            Инициализация                                <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Стандартные модули
const path = require('path');

// Глобальные константы
const APP_DIR = path.dirname(require.main.filename);

// Модули NPM
const amqp = require('amqplib/callback_api');

// Пользовательские модули
const settings = require(`${APP_DIR}/settings`);

// Конфигурация приложения
const config = settings.getConfig();

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                           Описание функций                              <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


// AMQP-клиент
const AMQPClient = class AMQPClient {

	// Строка подключения к RabbitMQ
	conn_str_amqp = null

	// Тайм-аут сердцебиения (сек.)
	heartbeat = 5

	// Конструктор
	constructor() {
		// Заполнение строки подключения
		this.conn_str_amqp = settings.getRabbitMQConnectionString();
	}

	// Устанавливает соединение по AMQP
	connect() {
		// ...
	}

	// Запуск публикатора
	create_confirm_channel(conn) {
		// ...
	}
```

```
        // Утверждение очереди
        assert_queue(ch, queue, options) {
                // ...
        }

        // Публикация сообщения
        publish(ch, exchange, routing_key, content) {
                // ...
        }

        // Подписка на очередь
        consume(ch, queue, process_msg) {
                // ...
        }

}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                            Экспорт функций                                <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

module.exports = AMQPClient

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Модуль должен обеспечивать автоматический реконнект к RabbitMQ в случае нештатных ситуаций (рестарт RabbitMQ, ребут Linux, «горячее» отключение виртуальной машины с Linux где развернут RabbitMQ).

Дополнительно к модулю настройки над amqplib должны быть предоставлены инструкции как его использовать и примеры кода с вызовом функций и методов.

Необходимо предоставить не только модуль, но и пример, демонстрирующий его работу. В самом простом варианте это может быть пять массивов:

1. Сообщения на отправку
2. Опубликованные в RabbitMQ
3. Не опубликованные в RabbitMQ
4. Сообщения, которые не могут быть перенаправлены
5. Полученные потребителем

Это необходимо, чтобы оценить статистику потерь сообщений при нештатных ситуациях.

Вспомогательная информация

Может помочь при выполнении этой задачи.

Генератор сообщений

Листинг 2 – Модуль, генерирующий сообщения на Node.js

```
/**
 *
 * @note
 * Генератор сообщений
 *
 * @details
 * Реализует генерацию N-сообщений с M-длиной (в символах)
 *
 */
'use strict';

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                            Инициализация                                <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

//Стандартные модули
const path = require('path');

// Глобальные константы
const APP_DIR = path.dirname(require.main.filename);

// Пользовательские модули
const settings = require(`${APP_DIR}/settings`);

// Конфигурация приложения
const config = settings.getConfig();

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                           Описание функций                              <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

// Генератор сообщений
const Generator = class Generator {

	// Конструктор
	constructor() {}

	// Генерация одного сообщения
	make_message(len) {
		let message 			= '';
		const characters 		=
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
		const characters_length = characters.length;
		for (let i = 0; i < len; i++) {
			message += characters.charAt(Math.floor(Math.random() *
characters_length));
		}
		return message;
	}
```

```
	// Генерация N-сообщений
	make_messages(len, count) {
		let messages = []
		for (let i = 0; i < count; i++) {
			messages.push(this.make_message(len))
		}
		return messages;
	}

}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// >                          Выполнение функций                              <
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

module.exports = Generator

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Установка RabbitMQ на Debian 10

Листинг 3 – Установка RabbitMQ на Debian 10

```
$ apt-get update
$ apt-get install erlang (deb http://dl.bintray.com/rabbitmq-erlang/debian buster
$ erlang-21.x)
$ apt-get install rabbitmq-server
$ service rabbitmq-server start
$ rabbitmq-plugins enable rabbitmq_management
$ rabbitmqctl add_user admin admin
$ rabbitmqctl set_user_tags admin administrator
$ rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"
```

Установка RabbitMQ на Docker

Листинг 4 – Установка RabbitMQ на Docker – docker-compose.yml

```
version: '3'
services:

  rabbitmq:
      build: ./ServiceRabbitMQ
      environment:
       - RABBITMQ_ERLANG_COOKIE=SWQOKODSQALRPCLNMEQG
       - RABBITMQ_DEFAULT_USER=rabbit
       - RABBITMQ_DEFAULT_PASS=rabbit
      ports:
       - '15672:15672'
       - '5672:5672'
```

Листинг 5 – Установка RabbitMQ на Docker – Dockerfile

```
FROM rabbitmq:3-management
COPY ./conf/enabled_plugins /etc/rabbitmq/enabled_plugins
```

Листинг 6 – Установка RabbitMQ на Docker – enabled_plugins

```
[rabbitmq_management, rabbitmq_management_visualiser].
```

Установка Node.js на Debian 10

Листинг 7 – Установка Node.js на Debian 10

```
https://github.com/nodesource/distributions/blob/master/README.md
https://nodejs.org/en/download/package-manager/
apt-get install -y nodejs
```

Установка Node.js на Docker

Листинг 8 – Установка RabbitMQ на Node.js – docker-compose.yml

```
version: '3'
services:

  nodejs_amqplib_rabbitmq:
    build:
      context: ./AmqpLibRabbitMQ
      dockerfile: Dockerfile
    restart: always
    tty: true
    volumes:
      - ./AmqpLibRabbitMQ:/app
    ports:
      - "8080:8080"
```

Листинг 9 – Установка RabbitMQ на Node.js – Dockerfile

```
# Базовый образ
FROM node:13.8.0-buster-slim
# Папка приложения
ARG APP_DIR=app
# Рабочая директория
WORKDIR ${APP_DIR}
```