Тестовое задание «Telegram бот с эхо сервером»

**Доступные языки программирования**: Python, Kotlin (Java)

**Доступные фреймворки и библиотеки**:

Для создание веб интерфейса разрешается использование любых библиотек и фреймворки, но преимуществом будет использование Django, Flask, Ktor, Spring

Для работы создания http запросов разрешается использование любых библиотек (К примеру таких как *Requests* для языка Python или *Retrofit (преимущественно)* для языков Java или Kotlin)  
Для работы с хранилищем данных пользователей разрешается использование любых библиотек, но преимущественно будет использование библиотек Ktorm, Expose.

**Описание задачи:** В качестве задачи вам предлагается реализовать эхо телеграмм-бота с веб интерфейсом

**Функциональность:**

* Многопользовательская поддержка (иначе говоря, этим ботом должны иметь возможность пользоваться одновременно несколько телеграмм пользователей)
* Эхо вывод данных с количеством сообщений через определенную задержку с буферизацией сообщений пользователя

\*Эхо вывод данных – вывод введенного пользователям сообщения (к примеру, пользователь вводит «привет» = бот выводит «привет 1», пользователь вводит «как дела?» = бот выводит «как дела? 2», пользователь вводит «пока» = бот выводит «пока 3»)

\*Буферизация данных – сохранение данных в буфер, до момента вывода сообщения пользователем ( Очередь сообщений пользователя)

* Сообщение пользователя должно проходить через веб-интерфейс (см. схему работы)
* Бот должен запоминать количество введенных сообщений пользователем
* Бот должен иметь возможность изменять задержку вывода сообщений через веб-интерфейс (см. схему работы)
* Запуск бота должен происходить из консоли одной командой
* Запуск бота должен поддерживать несколько аргументов запуска (см. список аргументов запуска)

**Cхема работы:**



1) При запуске бота пользователь вводит команду /start. В этот момент сервис должен "запоминать" пользователя

2) Весь дальнейший ввод информации, после начала использования бота и сохранения пользователя, считается как **echo message**.

3) При вводе эхо сообщения, эхо сообщение сохраняется в очередь сообщений (Обратите внимание, что у каждого пользователя своя очередь)

4) сообщение сохраняется в очереди на время **delay**

5) После времени задержки сообщение отправляется (см. описание веб-интерфейса)

6) Запрос проходит через сеть (или не проходит, в случае использования localhost) и принимается интерфейсом

7) После приема запроса добавляется индекс номера сообщение, производится повторная задержка со временем **delay** и отправляет измененное сообщение пользователю

8) Веб интерфейс должен иметь функцию обновления времени задержки delay

**Описание веб интерфейса:**

*1. /receiveEchoMessage*

POST

Данный запрос принимает эхо сообщение от отправителя.

Запрос принимает JSON:  
{

"message" : String, (Поле с сообщением)

"user\_sender" : String (Поле - телеграмм пользователь, отправитель запроса)

}

Запрос возвращает JSON:

{

"is\_ok" : Boolean, (Логическая переменная успешной обработки запрос)

"message\_number" : Int (Номер сообщения пользователя)

}

2. /updateQueueDelay

PUT

Данный запрос устанавливает новое время задержки запросов **delay**

Сервер принимает JSON:

{

new\_delay : Int (длительность задержки в милисекундах)

}

Сервер возвращает статус 200 в случае успешной обработки. В случае неуспешной обработки возвращать статус 500

**Список аргументов запуска:**

1) bot\_key - ключ используемого телеграмм бота

2) default\_delay - изначальное время задержки delay

Пример запуска сервиса должен находиться в текстовом файле проекта

Пример запуска сервиса на Python:

/python bot.py -bot\_key secretKey -default\_delay 2000

Пример запуска на Java/Kotlin gradle:

./graldlew run -x test --args="-bot\_key secretKey -default\_delay 2000"