Telegram Chat Bot для ISP

Технические условия:

1. Операционная система исполнения: Ubuntu Linux, Docker
2. Прикладной протокол взаимодействия с пользователем: Telegram ChatBot API
3. Протокол взаимодействия ядра системы с логическими модулями: JSON over Apache Kafka
4. Платежная система: Yandex.Касса (https://kassa.yandex.ru/blog/telegram)
5. Протокол взаимодействия с платежной системой: Telegram ChatBot API
6. VCS: Git (Gitlab), предоставляется заказчиком
7. CI среда: Jenkins, предоставляется заказчиком
8. Репозиторий артефактов:
    a. Docker Registry, предоставляется заказчиком
    b. Nexus, предоставляется заказчиком
9. Способ поставки: Docker контейнер
10. Свойства поставки ПО: итеративная поставка, CI/CD, git-flow
11. Обязательные средства реализации ядра системы:
    a. Scala 2.12
    b. SBT
    c. java-telegram-bot-api (https://github.com/pengrad/java-telegram-bot-api)
    d. SLF4J with Log4j
    e. Scala Test, Mockito
    f. Typesafe Config
    g. Scala StyleCheck для SBT (http://www.scalastyle.org/sbt.html)
    h. Scoverage (покрытие тестов)
    i. Apache Curator (взаимодействие с Zookeeper)
    j. Jackson
12. Обязательные средства реализации логических модулей системы:
    a. Python3
    b. Python native logging (каждый модуль ведет свой уникальный журнал)
13. Управление задачами: GitLab
14. Процесс CI: см. картинку далее (раздел: Пайплайн CI/CD)
15. Покрытие кода unit-тестами: не ниже 80%, только публичные методы

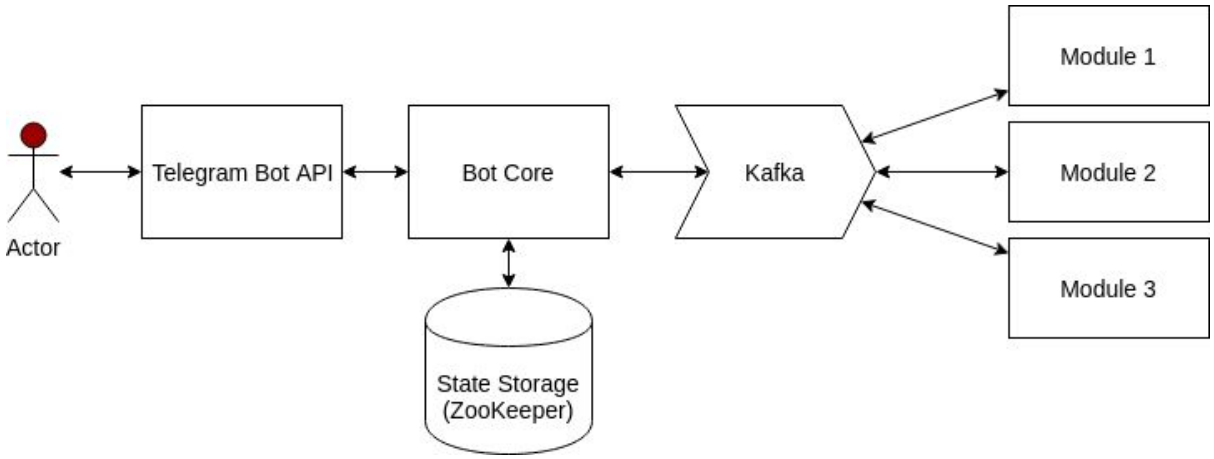
Обобщенная архитектура бота

Для хранения состояния бот использует Zookeeper, взаимодействие с которым осуществляется посредством Apache Curator. Zookeeper - обязательный компонент для развертывания Apache Kafka, в связи с чем представляется разумным использовать его же и для хранения состояния вместо реляционной или NoSQL СУБД, данные состояния хранятся в виде JSON. Бот может обрабатывать два типа команд:

1. команды от пользователя
2. периодические команды

Самостоятельно бот выполняет лишь валидацию и маршрутизацию команд, при этом фактическое исполнение делегируется логическим модулям обработки, реализованным на Python, за исключением функций, реализуемых посредством Telegram Bot API, например, платежа картой.

Бот не обязан сохранять in-flight состояние операций в Zookeeper, он может использовать для этого свою оперативную память. В большинстве случаев данное поведение приемлемо, однако, в случае наличия операций, действие которых необходимо возобновить или продолжить, он может это делать. Результаты операций, доступные для продолжительного использования бот должен сохранять в Zookeeper.



Каждый модуль - это отдельный процесс под управлением среды Python3, который осуществляет ровно одну задачу в цикле:

1. Прочитать сообщение из назначенного топика Kafka (используется 1 топик с одной партицией для чтения и одной партицией для записи данных), модуль определенного типа - синглтон.
2. Разобрать команду
3. Если выполнение команды возможно, **выполнить**, иначе вернуть ошибку с ее описанием:
    a. выполнить - OK (результат)
    b. выполнить - FAIL (результат)
4. Сохранить состояние Kafka Consumer в Zookeeper

Команда, отправляемая модулю обязательно содержит ID, login, и имеет внутренний TTL, которые используются следующим образом:

- ID предназначен для того, чтобы сопоставить ответ с запросом, ID ответа от модуля должен иметь тот же ID;
- login используется для аутентификации и определения области действия команды
- TTL - время, в течение которого команда валидна, к примеру, 60s

Пример команды:

```
{
  “requestId”: 1L,
  “authInfo”: “buratino”,
  “command”: “/balance”
}
```

Пример ответа на команду:

```
{
  “responseId”: 1L,
  “message”: “Ваш баланс 150 RUB”
}
```

Если необходимо, команды могут складываться в кэш, например, https://github.com/google/guava/wiki/CachesExplained с TTL и по истечению TTL удаляться из кэша с соответствующим журналированием.

Важные моменты реализации:

1. реализация должна позволять запуск на многоузловой системе;
2. docker-контейнеры конфигурируются с помощью JSON/YML конфигурационных файлов, передаваемых посредством docker bind;
3. должно быть реализовано тщательное журналирование с уровнями DEBUG, INFO, WARN, ERROR
4. для упрощения реализации все модули создаются в виде mock-модулей, без фактического бэкенда
5. Поддержка Unicode символов в сообщениях как сервером так и модулями.
6. Билд любой ветки должен проходить перед слиянием, ветка должна содержать необходимый набор тестов для тестирование всего функционала (unit, it).

Операции бота

Бот поддерживает следующие операции:

1. Аутентификация пользователя
2. Запрос баланса пользователя
3. Периодическое уведомление о балансе пользователю (daily)
4. Уведомление о снижении баланса пользователю (hourly)
5. Запрос о возможности получения кредитных средств и Запрос кредитных средств
6. Внесение платежа
7. Контакты службы поддержки

Аутентификация пользователя

Для аутентификации бот запрашивает у пользователя название аккаунта и выдает инструкцию по дальнейшим действиям. Инструкция генерируется модулем, например:

1. /hello
2. Добрый день, я #BotName#, для начала работы сообщите свой логин в биллинговой системе.
3. vasya
4. Отлично, зайдите в свой аккаунт по ссылке https://billing.com/link и скопируйте мне число, из обращения с заголовком “#BotName# - Аутентификация”
5. 1352556
6.
    a. Замечательно, теперь, когда мы знакомы, я могу выполнять ряд команд. Для ознакомления с ними введите /help
    b. К сожалению, я не могу узнать Вас, попробуйте ввести число еще один раз или запросите другое, введя команду /hello снова

Запрос баланса

Для запроса баланса пользователь выполняет команду /balance, если пользователь аутентифицирован, то происходит обращение к модулю, который выдает баланс, иначе, бот выдает сообщение о том, что необходимо аутентифицироваться, например, “Для выполнения данной команды необходимо сначала представиться, с помощью команды /hello”.

> /balance
< Ваш баланс 150 RUB

Наименования валюты возвращается модулем обработки, может ничего не возвращаться, к примеру.

Периодическое уведомление о балансе пользователю

Nb. Возможно, что для Telegram проще запрашивать аргументы команд интерактивно, а не аргументом, потому что это не совсем удобно.

Для периодического отчета о балансе пользователь использует команду /report-balance [HH:MM], на которую бот выдает текущий баланс и сообщение о том, что “Текущий баланс 300 RUB. Каждый день в HH:MM я буду сообщать о текущем балансе, отключение отчета с помощью команды /no-report-balance”. Если пользователь не задает аргумент HH:MM, то уведомление производится в 10:00 каждого дня.

Для отключения отчета необходимо ввести команду /no-report-balance, которая возвращает “Уведомление о балансе больше не будет производиться [в HH:MM], для включения отправьте команду /report-balance [HH:MM]”.

Если же пользователь не аутентифицирован, бот выдает сообщение о том, что необходимо аутентифицироваться, например, “Для выполнения данной команды необходимо сначала представиться, с помощью команды /hello”.

> /report-balance
< Текущий баланс 300 RUB. Каждый день в 10:00 я буду сообщать о текущем балансе, отключение отчета с помощью команды /no-report-balance

> /report-balance 16:00
< Текущий баланс 300 RUB. Каждый день в 16:00 я буду сообщать о текущем балансе, отключение отчета с помощью команды /no-report-balance

> /no-report-balance

< Ежедневное уведомление о балансе отключено. Для повторного включения используйте команду /report-balance или /report-balance [HH:MM]

Последующая команда “затирает” действие предыдущей.

Уведомление о снижении баланса пользователю

Nb. Возможно, что для Telegram проще запрашивать аргументы команд интерактивно, а не аргументом, потому что это не совсем удобно.

Пользователь может настроить бота для уведомления, когда баланс становится ниже определенного уровня. Проверка по данному уведомлению должна производиться ежечасно.

Настройка производится командой /report-low-balance [SUM]. Если SUM не задается, что используется значение, заданное по умолчанию в модуле обработки. На данную команду бот отвечает: “Текущий баланс 300 RUB. Я буду уведомлять Вас в случае, если баланс счета опустится ниже SUM RUB, для отключения используйте команду /no-report-low-balance.”

Если же пользователь не аутентифицирован, бот выдает сообщение о том, что необходимо аутентифицироваться, например, “Для выполнения данной команды необходимо сначала представиться, с помощью команды /hello”.

> /report-low-balance
< Текущий баланс 300 RUB. Я буду уведомлять Вас в случае, если баланс счета опустится ниже 500 RUB, для отключения используйте команду /no-report-low-balance.

> /report-low-balance 1000
< Текущий баланс 300 RUB. Я буду уведомлять Вас в случае, если баланс счета опустится ниже 1000 RUB, для отключения используйте команду /no-report-low-balance.

When event happens:

< Баланс опустился ниже 1000 RUB. Для пополнения счета через Yandex.Касса используйте команду /pay.

> /no-report-low-balance
< Уведомление о низком балансе отключено. Для повторного включения используйте команду /report-low-balance или /report-low-balance SUM.

Запрос о возможности получения кредитных средств и Запрос кредитных средств

Nb. Возможно, что для Telegram проще запрашивать аргументы команд интерактивно, а не аргументом, потому что это не совсем удобно.

Пользователь может запросить кредитные средства для временного пополнения счета при отсутствии фактических средств. Для того, чтобы определить сколько средств пользователь может попросить и на какой срок, он выполняет команду /credit.

Бот реагирует одним из следующих способов:

> /credit
< Добрый день, выделение кредитных средств невозможно по причине: ....
> /credit
< Я могу выделить **1000** RUB кредитных средств сроком на **5** дней. Для запроса используйте команду /get-credit

> /get-credit
< Выделен кредит 1000 RUB сроком на 5 дней, кредит будет отозван 10 марта 2018 года в 17:00

> get-credit 1000 2
< Выделен кредит 500 RUB сроком на 2 дня, кредит будет отозван 7 марта 2018 года в 15:00

В случае успешного выделения кредита, его отзыв осуществляется в указанный дедлайн ботом обращением к кредитному модулю. После отзыва кредита пользователь получает уведомление:

< Кредитный лимит на сумму 1000 RUB, выделенный 5 марта 2018 года в 17:00 снят. Спасибо за обращение.

Внесение платежа

Nb. Возможно, что для Telegram проще запрашивать аргументы команд интерактивно, а не аргументом, потому что это не совсем удобно.

Внесение платежа осуществляется с помощью команды /pay [SUM] и telegram, команда запоминает предыдущее количество средств для оплаты и по-умолчанию использует его при последующих платежах. К примеру,

> /pay
< Хотите пополнить счет на сумму 1000 рублей пластиковой картой через Yandex.Касса?

> /pay 500
< Хотите пополнить счет на сумму 500 рублей пластиковой картой через Yandex.Касса?

> /pay
< Хотите пополнить счет на сумму 500 рублей пластиковой картой через Yandex.Касса?

NB Данный раздел требует уточнения в связи с необходимостью более детального анализа Telegram Bot API.

Контакты службы поддержки

Команда /support выводит текст с различными способами обращения в службу поддержки.

Пайплайн CI/CD