

JetBot

Выполнили:

Саввинов Дмитрий, Власов Святослав

Руководитель:

Александр Подхалюзин



Введение

- Цель работы – добавить поддержку ботов в мессенджер **JetChat**.

JetChat ⚙️

My chats

- # jetchat-dev
- # hackathon-2016
- # jetchat-ux
- # practicies
- # go-lang-idea-plugin
- # jetchat-features
- # jetchat-support
- # jetchat-qa
- # jetchat-test-messages

[New group](#)

- Alexander Podkhalyuzin
- Svyatoslav Vlasov
- Pavel Nikitin
- Konstantin Bulenkov
- Victor Kropp
- Andrey Cheptsov
- Yann Cébron

🔍 Search everything

jetchat-dev ☆

▼ New topic

Стендап

Andrey 1 week ago

Тесты

Andrey 3 weeks ago

Интеграция GitHub

Andrey 4 weeks ago ☆

JetChat to rename

Alexander 4 weeks ago

Боты

Andrey 1 month ago

Давайте обсудим X

Andrey 1 month ago

Оффлайн

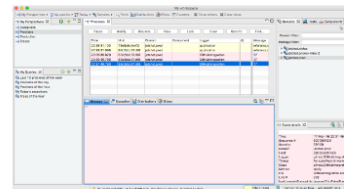
Kirill 1 month ago

Вопрос по React

Andrey 1 month ago

UI очень медленный

сделайте



см нижний правый угол

я сервак поднял но если снова упадет я отключу интеграцию

на время

----- begin log -----

ERROR | 18-Mar-2016 00:29:38 | 88c2dcb37c86 |

[jetchat.prod](#)

| GitHubIntegration\$GitHubMessageHandler\$\$anonfun\$collectMessages\$2\$\$anonfun\$apply\$12 | 17

ERROR | 18-Mar-2016 0



0:46 | 88c2dcb37c86 |

<https://github.com/svloyso/JetChat>

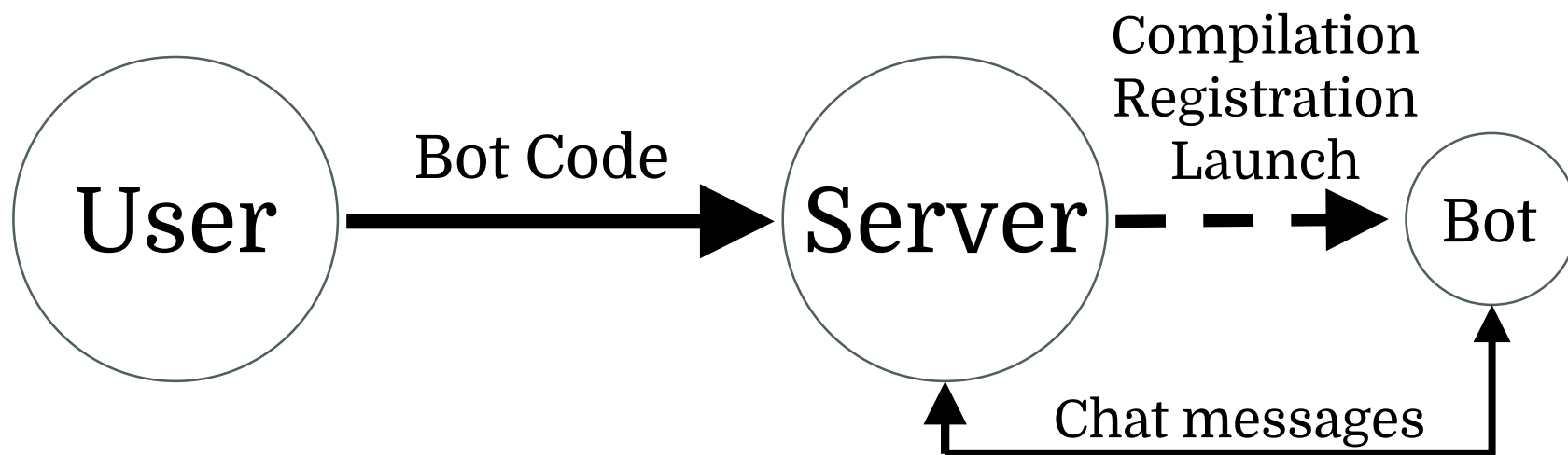
Инструменты

- Scala – основной язык,
- Play – Web Framework для Scala,
- SBT – система сборки,
- Slick – ORM для Scala,
- Akka – библиотека для написания акторных систем.



<https://github.com/svloyso/JetChat>

Создание бота



<https://github.com/svloyso/JetChat>

Архитектурные задачи

1. Придумать гибкую архитектуру, минимально зависящую от изменений в ядре чата,
2. Архитектура должна естественным образом обобщаться на случай акторов, поднятых вне сервера, на стороне пользователя.

<https://github.com/svloyso/JetChat>

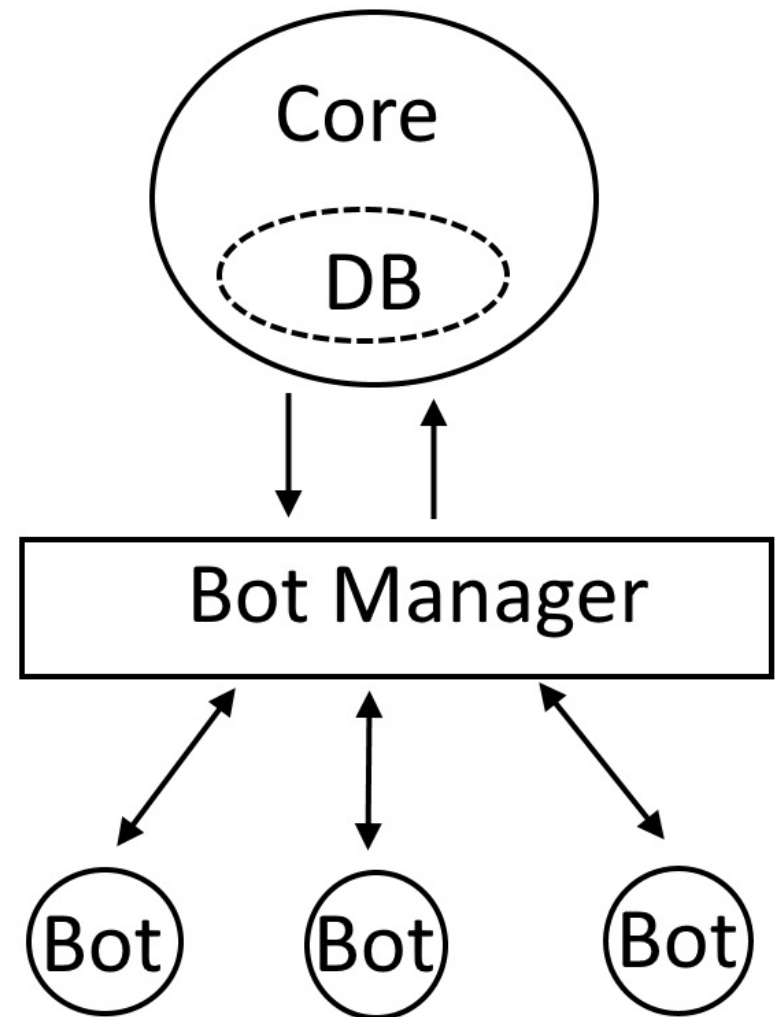
BackEnd

Задачи:

1. Предоставление удобного интерфейса для написания DSL,
2. Инкапсуляция и изолирование функций ядра от ботов,
3. Предоставление интерфейса для взаимодействия ботов и ядра,
4. Предоставление интерфейса для сборки, запуска, остановки и уничтожения ботов,
5. Persistence для ботов.

BotManager

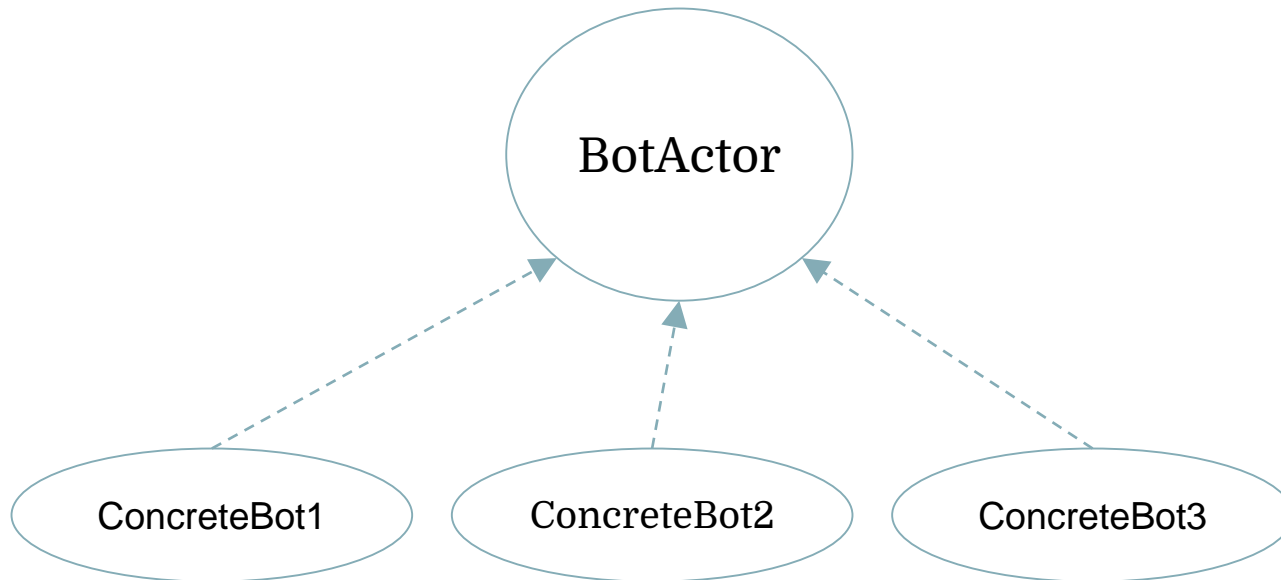
1. Изоляция ботов от ядра чата
2. Реализация взаимодействия между ботами и ядром
3. Сборка и запуск новых ботов
4. Persistence кода и состояния ботов



BotActor

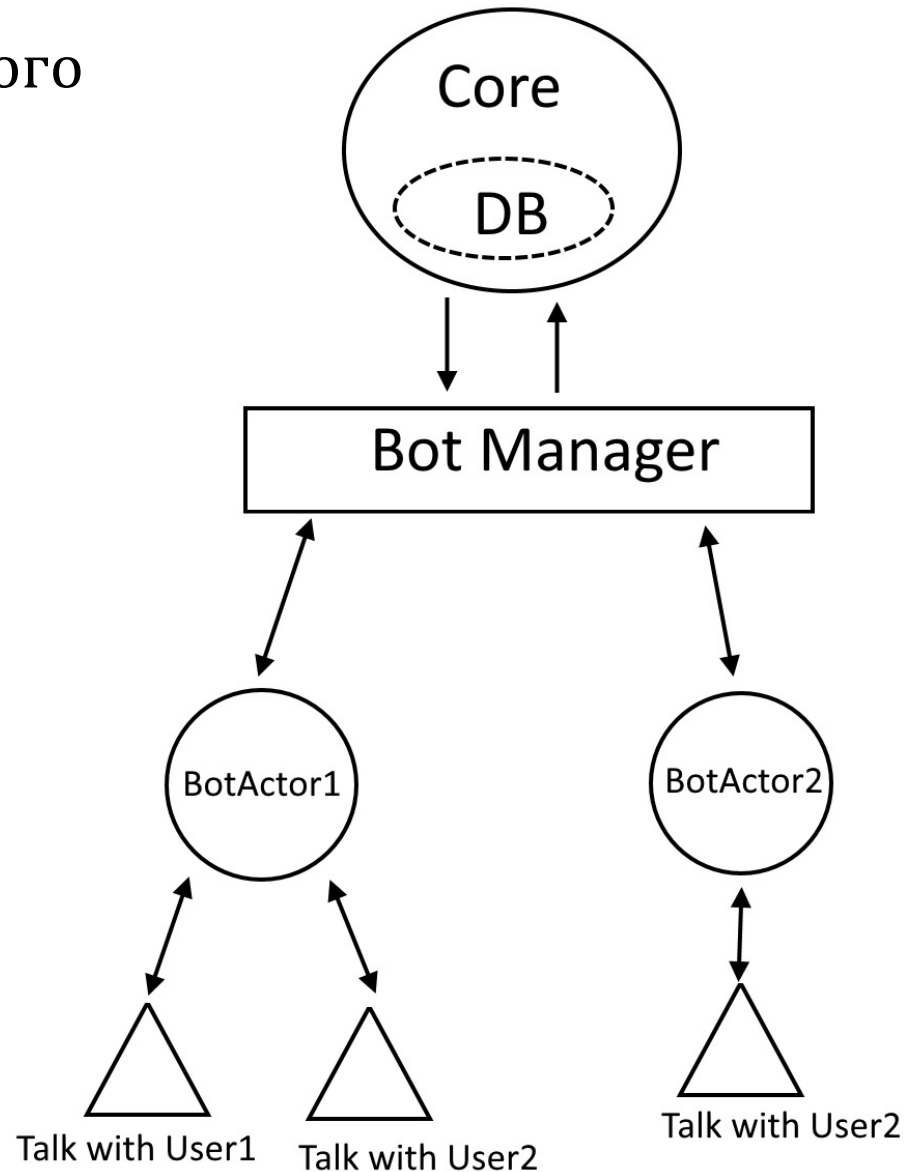
1. Базовый абстрактный класс для актора бота;
2. Интерфейс для взаимодействия бота и BotManager'a;
3. Persistence. Интерфейс для dump'a и restore'a состояния бота.

```
def send(...) = manager ! BotSend(...)
def sendDirect(...) = manager ! BotDirSend(...)
def getUserList = manager ? GetUserList
def getUserByName(...) = manager ? GetUserByName(...)
```



BotActor, Talk

Talk – беседа конкретного бота с конкретным пользователем



DSL

DSL - Domain Specific Language

```
...  
val network = new BasicNetwork();  
network.addLayer(new BasicLayer(null,true,2));  
network.addLayer(new BasicLayer(new  
    ActivationSigmoid(),true,3));  
network.addLayer(new BasicLayer(new  
    ActivationSigmoid(),false,1));  
network.getStructure().finalizeStructure();
```

```
... ..  
val network =  
    (InputLayer having bias having 2.neurons) +  
    (ActivationSigmoid having bias having 3.neurons) +  
    (ActivationSigmoid having 1.neurons)
```

...

Пример

```
01. bot = Bot("my-bot")
02. type List = collection.mutable.ListBuffer[String]
03.
04. bot storesData[List] "history" initWith new List()
05.
06. val start = State("Start") {
07.     case TextMessage(address, text) =>
08.         if (text == "hello") {
09.             say("Hello! :)")
10.             moveTo("Listening")
11.         }
12. }
13.
```

Пример

```
14. val listening = State("Listening") {
15.     case TextMessage(address, text) =>
16.         val pattern = "find (\d*)".r
17.         text match {
18.             case pattern(count) => data.history.take(count)
19.             case "Bye!" => say("Good bye"); terminate()
20.             case other => data.history.append(other)
21.         }
22. }
23.
24. bot startsWith start
25.
26. bot + start + listening
```

Приемы и паттерны

1. Инфиксная запись

`bot.startsWith(start) →`
`bot startsWith start`

2. Scala Dynamics

`data.get("history") → data.history`

3. Method Chaining

`bot.storesData[Int]("foo", 0) →`
`bot storesData[Int] "foo" initWith 0`

4. Scala Macros / Parsers-Combinators

Улучшение синтаксиса

Демонстрация результатов

Направления дальнейшего развития

- Создание независимого сервиса для создания кросс-платформенных ботов;
- Написание пользовательского интерфейса для загрузки ботов;
- Дальнейшее улучшение DSL с помощью Scala Macros;
- Введение политик безопасности для ботов (борьба со спамом, потенциальными взломами сервера, чрезмерным использованием серверного времени, etc);
- Работа с кластером.

Итоги

- Изучили современный стек технологий (Scala/Akka/Play/SBT/Slick);
- Опробовали себя в роли разработчиков гибкой и масштабируемой архитектуры, краткого и понятного API;
- Взглянули на устройство ботов изнутри;
- Получили опыт командной работы.

Спасибо за внимание!

<https://github.com/svloyso/JetChat>