

UN-ION DISCORD BOT

60-я НК БГУИР

Структура презентации

- ▶ Общие понятия
- ▶ Задумка и проектирование
- ▶ Развитие бота на примере эволюции его архитектуры и функционала
- ▶ Итоговая архитектура и функционал
- ▶ Структуры данных
- ▶ Типичный алгоритм сервиса

Что такое Discord?

Discord - это платформа для общения в режиме реального времени, разработанная для групповых обсуждений, обмена сообщениями, голосовых звонков и видеозвонков. Она предоставляет возможность создания серверов, на которых пользователи могут создавать текстовые и голосовые каналы для общения по интересам.



СерверН

Сервер Hummel009

Мероприятия

ТЕКСТОВЫЕ КАНАЛЫ

основной

ГОЛОСОВЫЕ КАНАЛЫ

Основной

Hummel009

здесь то...

основной

Это ваш новый сервер. Здесь приведены шаги, которые помогут вам начать с ним работу. Вы можете найти больше советов в нашем [руководстве для начинающих](#).

Пригласите друзей

Персонализируйте свой сервер с помощью значка

Отправьте первое сообщение

Загрузите приложение Discord

Добавьте первое приложение

24 апреля 2024 г.

Hummel009

Сегодня, в 18:31

Привет, чат!

Написать #основной

Боты в Discord

Боты выглядят, как обычные аккаунты, но управляются не человеком, а программой. Обычно они разрабатываются для облегчения административных задач, предоставления дополнительной функциональности и просто ради развлечения участников.

Существует множество популярных ботов. Самые известные — это Dyno, MEE6, Dank Memer, Rythm, Tatsumaki и Poll Bot.

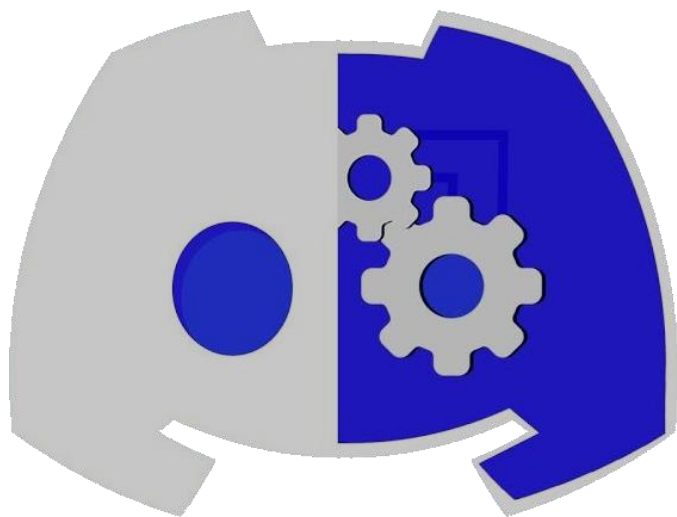


MEE6

Discord API

Бесплатные боты в Discord обычно предоставляют наиболее популярные и простые функции. Если же задача поставлена достаточно уникальная, то бесплатного бота может и не найтись. В худшем случае, ни один платный бот не выполняет то, что нужно именно вам.

В таком случае встаёт вопрос разработки своего бота. Благо, Discord предоставляет хорошо документированный API, который позволяет любому желающему создавать ботов с кастомным функционалом. Для этого требуется лишь опыт работы с HTTP-запросами.



Проблемы реализации

Уже на стадии разработки бота могут возникнуть различные проблемы. Например, работа с HTTP-запросами к Discord API достаточно утомительна за счёт относительно низкоуровневого подхода. Существуют сторонние библиотеки с наборами готовых функций, но они иногда устаревают и оказываются заброшенными.

Развёртывание уже готового бота — тоже непростая задача. Чтобы бот работал круглосуточно, нужно либо покупать сервер (довольно дорого), либо арендовать хост (относительно дорого), либо запускать бота на своём компьютере и держать его включённым круглосуточно (неудобно).



Нестандартная идея

Боты для Discord, как и боты для других мессенджеров и соцсетей, чаще всего пишутся на Python. Однако, в теории, разработка бота возможна на любом языке, который поддерживает работу с HTTP-запросами.

Необходимость создания бота для собственного сервера и опыт работы с Java/Kotlin привёл меня к мысли о создании **бота, который будет компилироваться и работать на телефоне**. Телефон — идеальный бесплатный хост, который работает круглосуточно и имеет доступ к мобильному интернету.



Выбор языка и библиотеки

Поскольку Java входит в число самых популярных языков программирования, неудивительно, что для неё существует сторонняя библиотека по работе с Discord API — **Javacord**.

Javacord требует для работы JDK 8 (1.8), следовательно, его уровень байткода совместим с абсолютным большинством версий Android.

Kotlin, предоставляющий наиболее современный пользовательский опыт работы с Android API, также совместим с JDK 8 (1.8). Стоит отметить, что все библиотеки для Java совместимы и с Kotlin.



Архитектура и технологический стек

Первоначально я решил разработать бота на Kotlin, который будет компилироваться под Windows. Доступ к терминалу облегчал тестирование и отладку приложения, что удобно на ранней стадии разработки.

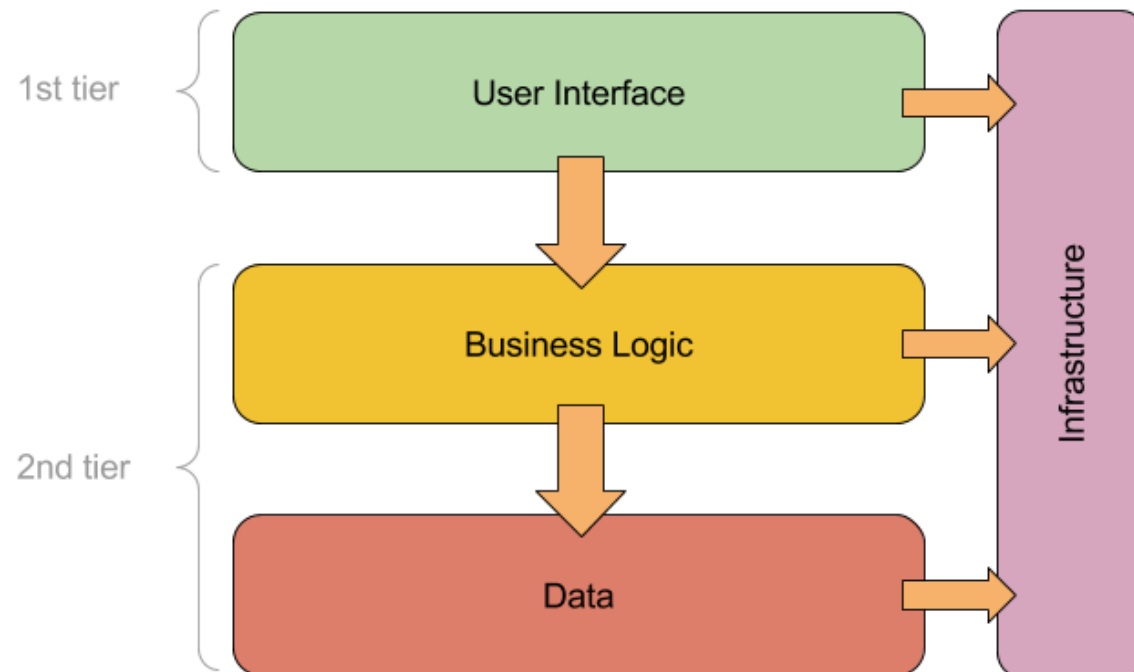
Бот начал реализовываться в виде **многомодульного проекта Gradle** (Kotlin DSL). Первыми появились модули **Windows** и **Common**. Согласно задумке, первый модуль должен был содержать сугубо GUI и входные данные (токен бота и идентификатор владельца), а второй — весь функционал, который позже сможет работать и на Android.



Архитектура и технологический стек

У ботов нет общепринятой архитектуры — некоторые разработчики пишут ботов даже в процедурном стиле.

Имея опыт работы с Java, я принял решение использовать Layered Architecture (Controller, Service, DAO) с применением паттернов Singleton Factory и Bean.



Архитектура и технологический стек

Для работы компонентов было задействовано три библиотеки, не считая Javacord: **Zip4j**, **Gson** и **Apache HTTP Client**.

- ▶ Поскольку бот не взаимодействует с базой данных, то было решено, что слой DAO будет предназначен для работы с файловой системой. Всего было добавлено три компонента — функции для работы с bin-файлами данных, функции для работы с json-файлами конфигурации и функции для работы с zip-архивами.
- ▶ Слой Service, согласно изначальному плану, должен был содержать шесть компонентов — функции для проверки доступа к команде, функции для авторизации бота, функции владельца, функции админа, функции бота и функции рядового участника.
- ▶ Слой Controller представляет собой Listener'ы шины событий библиотеки Javacord.

```
dependencies {  
    implementation("net.lingala.zip4j:zip4j:2.11.5")  
    implementation("com.google.code.gson:gson:2.10.1")  
    implementation("org.apache.httpcomponents.client5:httpclient5:5.3.1")  
    implementation(dependencyNotation: "org.javacord:javacord:3.8.0") {  
        exclude(group = "org.bouncycastle")  
    }  
}
```

Первая фаза разработки

Во время первой фазы разработки проводилось написание основного кода для функций бота, а также его отладка, оптимизация и исправление ошибок. Так, в определённый момент был совершён переход от обычных команд Discord к интеракционным, которые являются отдельным ивентом и позволяют не запускать проверки на каждом сообщении в чате.

Для разработки графического интерфейса был использован **Compose Desktop** — фреймворк для разработки декларативных GUI на Kotlin, который позволяет применить опыт и стиль разработки интерфейсов Android на других платформах.




Итоги первой фазы разработки

В результате первой фазы разработки был завершён весь основной функционал бота, а именно:

- ▶ Добавление и удаление дней рождения, менеджеров и секретных каналов бота.
- ▶ Установка языка бота и шанса цитирования/простановки реакции.
- ▶ Массовое удаление сообщений в чате.
- ▶ Рандомное цитирование участников и простановка реакций к сообщениям.
- ▶ Поздравление участников с днём рождения.
- ▶ Импорт и экспорт данных бота в формате zip-архива.
- ▶ Дистанционное выключение бота.
- ▶ Рандомный ответ на вопрос участника, выбор слова из списка, выбор числа из диапазона, вывод информации о сервере.
- ▶ Работа с нейросетью «Порфирьевич» для продления текста.

Итоги первой фазы разработки

 БОГДАН

/add_birthday

/add_birthday [user_id] [month_number] [day_number]

/add_manager

/add_manager [role_id]

/add_secret_channel

/add_secret_channel [channel_id]

/answer

/answer [your question]


/choice

/choice [one] [two] [three]

/clear_birthdays

/clear_birthdays {user_id}

/clear_data

 БОГДАН

/clear_managers

/clear_managers {role_id}

/clear_messages

/clear_messages

/clear_secret_channels

/clear_secret_channels {channel_id}

/complete

/complete [text]

/exit

/exit

/export

/export

/import

/import

/info

/info

/nuke

/nuke [number]

/random

/random [number]


/set_chance

/set_chance [number]


/set_language

/set_language [ru/en]

Пример функционала


 Hummel009 использует **random**

Богдан ПРИЛОЖЕНИЕ Сегодня, в 18:32


 hummel009

Поспех


Выпадковы лік: 53.

 **Богдан** ПРИЛОЖЕНИЕ 08.04.2024 22:27


Привет, можешь скинуть 4 лабу пж

 **Богдан** ПРИЛОЖЕНИЕ 19.12.2023 0:00

@PlushBoy рлву, с днём рождения!


 Hummel009 использует **complete**

Богдан ПРИЛОЖЕНИЕ Сегодня, в 17:22


 hummel009

Поспех

Владимир Жириновский запретил студенту БГУИР Алексею Дубовскому проносить в библиотеку книги Свидетелей Иеговы, сославшись на их якобы непристойный характер.

 Hummel009 использует **answer**

Богдан ПРИЛОЖЕНИЕ Сегодня, в 18:36

 hummel009

Успех

- Привет, ты умеешь отвечать на вопросы?
- Так и есть!

Вторая фаза разработки

Во второй фазе разработки началась разработка третьего модуля проекта — **Android**. В ходе разработки пришлось вносить коррективы в модуль **Common**, поскольку работа с файловой системой в Android ограничена и зависит от **App Context**.

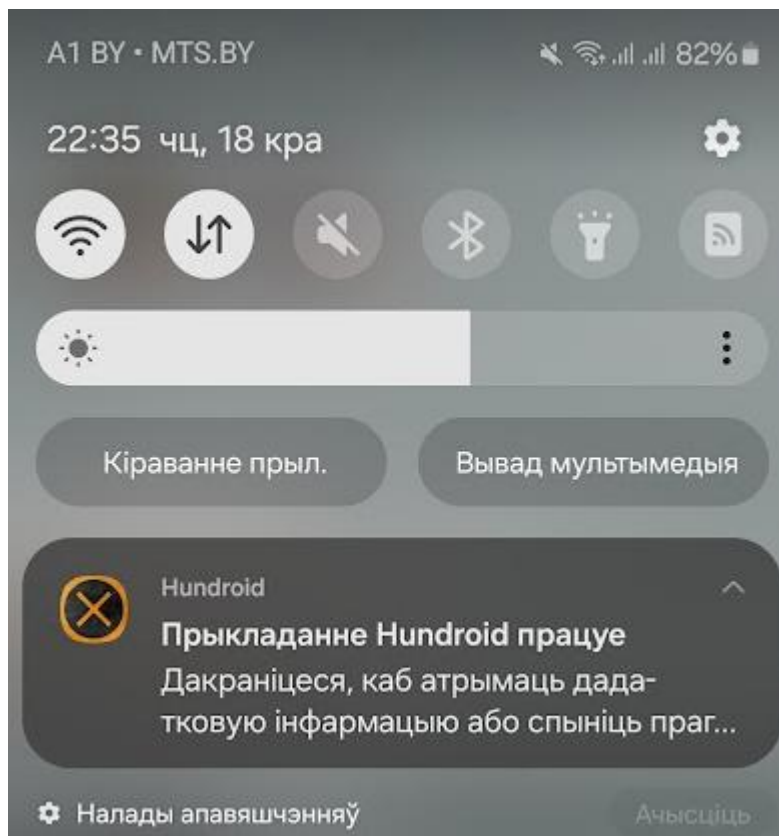
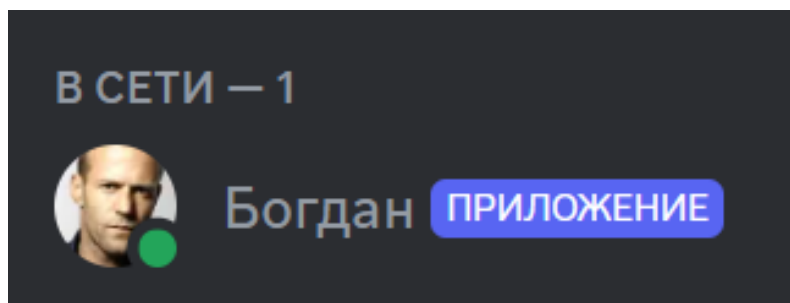
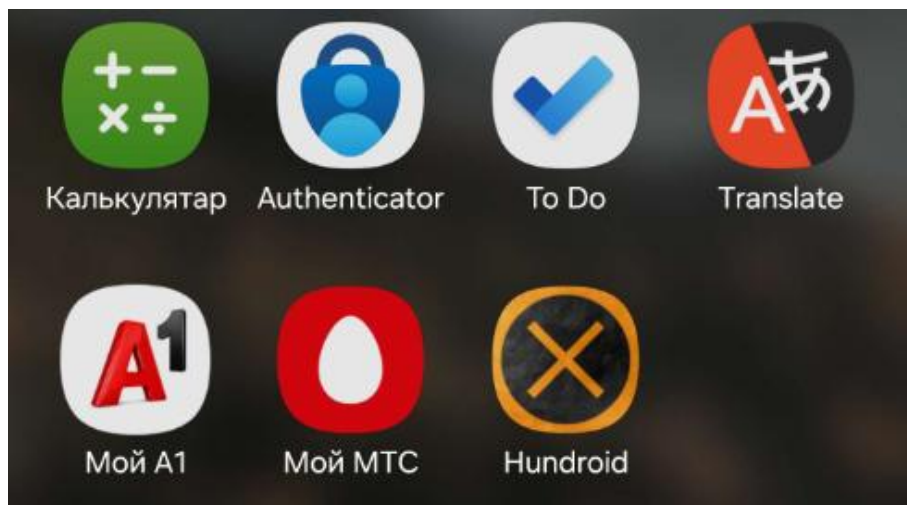
Методом проб и ошибок был выбран тип приложения — **форграунд-сервис** (то есть сервис, который работает в фоновом режиме, но при этом явно, с уведомлением). Именно этот способ позволяет создать приложение, которое, при отсутствии режима экономии, будет работать в Android бесконечно.

Графический интерфейс был разработан при помощи **Jetpack Compose**, то есть той же технологией, что и интерфейс для Windows.



Итоги второй фазы разработки

В результате второй фазы разработки был создан третий модуль, позволяющий запустить бота на телефоне. Была добавлена **иконка приложения™**. Программа показала высокую стабильность и длительную работу в круглосуточном режиме.

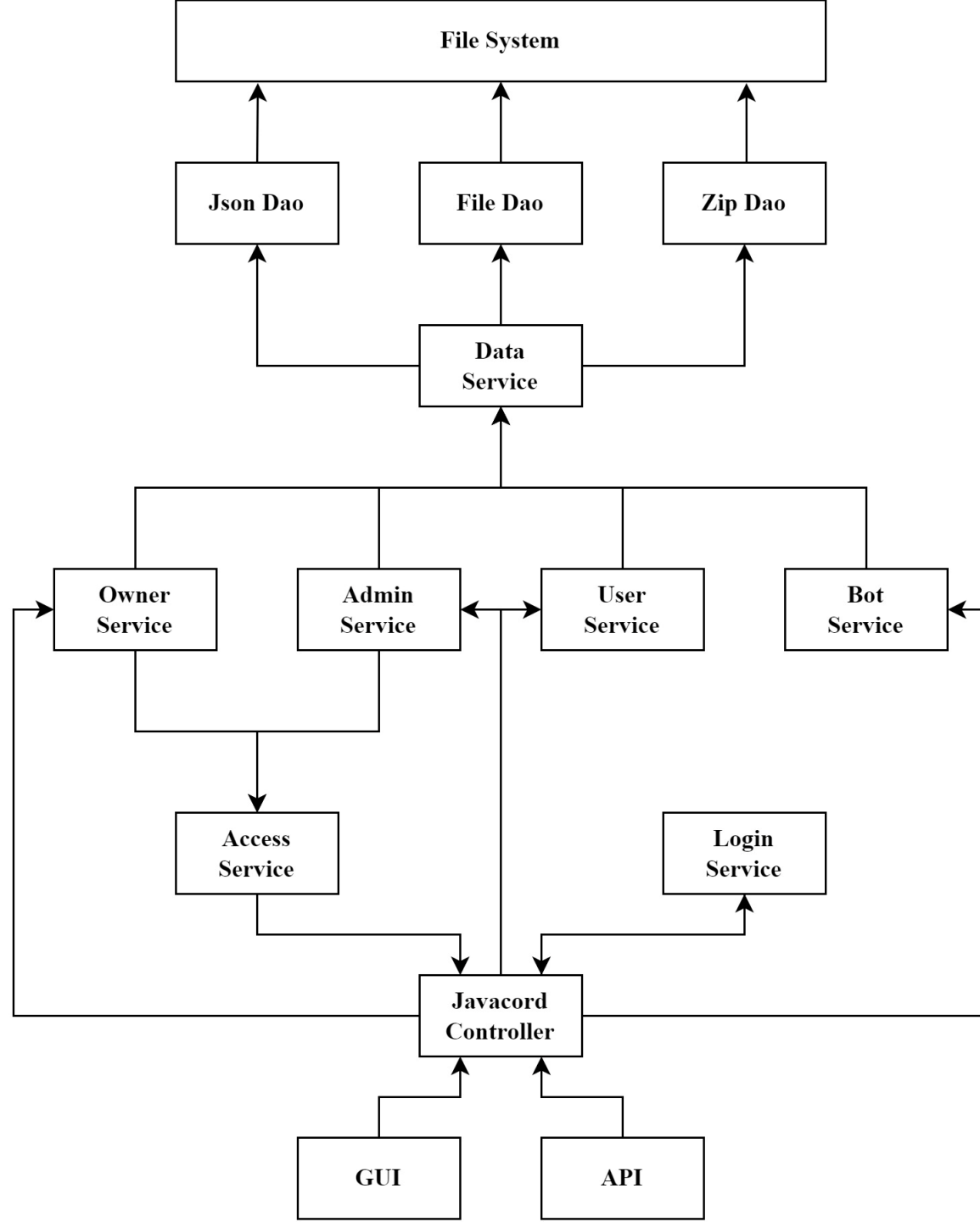


Третья фаза разработки

В ходе третьей фазы проводилась «полировка» уже существующего функционала, повышение стабильности и переработка плохих архитектурных решений.

В частности, именно в этот момент был введён седьмой компонент сервисов — функции для взаимодействия с DAO. Он позволил систематизировать обращения к DAO и собрать их все в одном месте, чтобы безошибочно спроектировать структуру папок с данными.

```
override fun loadServerData(server: Server): ServerData {  
    val folderName = server.id.toString()  
    val filePath = "servers/$folderName/data.json"  
    return jsonDao.readFromJson(filePath, ServerData::class.java) ?: getServerDataFromDiscord(server)  
}  
  
override fun saveServerData(server: Server, serverData: ServerData) {  
    val folderName = server.id.toString()  
    val filePath = "servers/$folderName/data.json"  
    jsonDao.writeToJson(filePath, serverData)  
}
```



Итоги разработки

Помимо новых сервисов, была разработана новая система локализации на основе форматированных строк. Это позволяет делать качественные переводы на любой язык, учитывая то, что в разных языках порядок слов и построение предложений могут отличаться.

```
"set_chance": "Усталяваны шанс паведамлення: %d.",  
"current_language": "Бягучая мова інтэрфейса: %s.",  
"current_chance": "Бягучы шанс паведамлення: %d.",  
"added_manager": "Дададзена кіраўніцкая роль: <@%d>.",  
"added_channel": "Дададзены сакрэтны канал: <#%d>.",  
"added_birthday": "Дададзены дзень нараджэння: <@%d>, %s.",  
"removed_manager": "Выдалена кіраўніцкая роль: <@%d>.",  
"removed_channel": "Выдалены сакрэтны канал: <#%d>.",  
"removed_birthday": "Выдалены дзень нараджэння: <@%d>.",  
"cleared_managers": "Кіраўніцкія ролі сервера ачышчаны.",  
"cleared_channels": "Сакрэтныя каналы сервера ачышчаны.",  
"cleared_birthdays": "Дні нараджэння сервера ачышчаны."
```

Структуры данных

В презентации уже были рассмотрены слои архитектуры, но не были рассмотрены структуры данных. Всего бот использует четыре структуры данных: это **ServerData**, **ApiResponse** и **BotData**.

- ▶ **ServerData** — самый важный класс данных, содержащий всю информацию о сервере.
- ▶ **ApiResponse** — класс, хранящий ответ с API «Порфирьевича».
- ▶ **BotData** — синглтон, хранящий данные из GUI Windows/Android.

```
data class ServerData(  
    val dataVer: Int,  
    val serverId: String,  
    val serverName: String,  
    var chance: Int,  
    var lang: String,  
    val lastWish: Date,  
    val secretChannels: MutableSet<Channel>,  
    val managers: MutableSet<Role>,  
    val birthdays: MutableSet<Birthday>  
) {  
    data class Date(var day: Int, var month: Int)  
  
    data class Role(var id: Long)  
  
    data class Channel(var id: Long)  
  
    data class Birthday(var id: Long, var date: Date)
```

Пример алгоритма функции

Все алгоритмы сервисов изначально похожи, отличаясь лишь на глубоких уровнях вложенности.

```
override fun clearMessages(event: InteractionCreateEvent) {  
    val sc = event.slashCommandInteraction.get()  
  
    if (sc.fullCommandName.contains( other: "clear_messages")) {  
        sc.respondLater().thenAccept {  
            val server = sc.server.get()  
            val serverData = dataService.loadServerData(server)  
  
            val embed = if (!accessService.fromAdminAtLeast(sc, serverData)) {  
                EmbedBuilder().access(sc, serverData, I18n.of( key: "no_access", serverData))  
            } else {  
                dataService.wipeServerMessages(server)  
                EmbedBuilder().success(sc, serverData, I18n.of( key: "cleared_messages", serverData))  
            }  
            sc.createFollowupMessageBuilder().addEmbed(embed).send().get()  
        }.get()  
    }  
}
```

Статистика:

- Внесено 268 коммитов
- Изменено 40 тысяч строк кода
- Высокое качество подтверждено платформой Sonarcloud

README

GPL-3.0 license

code smells 20

maintainability A

security A

bugs 0

vulnerabilities 0

duplicated lines 8.2%

reliability A

quality gate passed

technical debt 3h

lines of code 1.9k

Мультиплатформенный бот для Discord. Может быть запущен с Windows либо Android. Поскольку телефон у большинства людей всегда включен и находится в зоне досягаемости интернета, то это, фактически, бесплатный хост.

Main Branch Summary

2k Lines of Code

Quality Gate Status

Passed

Measures

Last analysis 5 days ago • b90dfa09

New Code

Overall Code

Security

0 Open issues

A

0 H 0 M 0 L

Reliability

0 Open issues

A

0 H 0 M 0 L

Maintainability

20 Open issues

A

10 H 7 M 3 L

Accepted Issues

0

🔔

Coverage

A few extra steps are needed for SonarCloud to analyze your code coverage

Setup coverage analysis

🔗

🛂

Duplications

8.2%

No conditions set on 2.2k Lines

🔄

Security Hotspots

0

```
override fun addBirthday(event: InteractionCreateEvent) {
    val sc = event.slashCommandInteraction.get()

    if (sc.fullCommandName.contains("add_birthday")) {
        sc.respondLater().thenAccept {
            val server = sc.server.get()
            val serverData = dataService.loadServerData(server)

            val embed = if (!accessService.fromAdminAtLeast(sc, serverData)) {
                EmbedBuilder().access(sc, serverData, I18n.of("no_access", serverData))
            } else {
                val arguments = sc.arguments[0].stringValue.get().split(" ")
                if (arguments.size == 3) {
                    try {
                        val userId = arguments[0].toLong()
                        val month = if (arguments[1].toInt() in 1..12) arguments[1].toInt() else throw Exception()
                        val range = ranges[month] ?: throw Exception()
                        val day = if (arguments[2].toInt() in range) arguments[2].toInt() else throw Exception()
                        if (!sc.server.get().getMemberById(userId).isPresent) {
                            throw Exception()
                        }
                        serverData.birthdays.add(ServerData.Birthday(userId, ServerData.Date(day, month)))

                        val date = I18n.of(Month.of(month).name.lowercase(), serverData).format(day)

                        EmbedBuilder().success(
                            sc, serverData, I18n.of("added_birthday", serverData).format(userId, date)
                        )
                    } catch (e: Exception) {
                        EmbedBuilder().error(sc, serverData, I18n.of("invalid_format", serverData))
                    }
                } else {
                    EmbedBuilder().error(sc, serverData, I18n.of("invalid_arg", serverData))
                }
            }

            sc.createFollowupMessageBuilder().addEmbed(embed).send().get()

            dataService.saveServerData(server, serverData)
        }.get()
    }
}
```


Спасибо за внимание!

Полезные ссылки:

- ▶ [Hummel009/UN-ION-Discord-Bot: Мой бот для моего сервера в Discord \(github.com\)](#)
- ▶ [Javacord/Javacord: An easy to use multithreaded library for creating Discord bots in Java. \(github.com\)](#)
- ▶ [Javacord](#)
- ▶ [Develop for Android | Android Developers](#)
- ▶ [Android Compose Tutorial | Jetpack Compose | Android Developers](#)
- ▶ [Gradle Build Tool](#)
- ▶ [Kotlin Programming Language \(kotlinlang.org\)](#)
- ▶ [Maven Repository: Search/Browse/Explore \(mvnrepository.com\)](#)