Кросс-платформенная аналитика

ManyChat

Лавриненко Лавриненко

Пара фактов обо мне

- Несколько своих стартапов
- Сбербанк онлайн
- ManyChat
- Помогаю организовывать Podlodka iOS Crew

ManyChat

- Messenger Marketing (Facebook, SMS, email)
- Более 1 000 000 клиентов по миру
- Мобильное приложение поддержки LiveChat
- Бизнес хочет ответить клиенту максимально быстро

ManyChat

О чем будем говорить?

- Что такое и зачем нужна аналитика? – 10 минут
- Боль аналитики в мобилке – 5 минут
- Превращаем аналитику в код – 20 минут
- Заворачиваем код в KMM и распространяем – 10 минут
- Где взять готовый проект, где можно все пощупать? – 60 секунд
- Ваши вопросы – сколько потребуется:)

ManyChat

Что такое аналитика?

ManyChat

Однажды ты просыпаешься с мыслью

Пробежать марафон

ManyChat

Как это сделать?

– Пойти в секцию
– Бегать по утрам
– Купить беговые кроссовки
– ...

ManyChat

Но как понять,
что из этого эффективно?
Что делать в первую очередь?

ManyChat

А что если

Купить фитнес браслет и начать мерить:

– Пульс

– Количество километров

– Время за километр

ManyChat

Всё это - показатели эффективности

На их основании можно принимать решения и адаптироваться. Даже на бегу :)

ManyChat

В продукте

- Снимаем метрики с продукта в процессе разработки
- Принимаем решения на их основании

ManyChat

Какие бывают метрики?

- DAU/WAU/MAY
- Retention
- LTV
- Длина сессий
- Воронки
- Adoption

ManyChat

MAU/DAU

Количества пользователей, которые зашли минимум раз в месяц/день

Чем полезно:

- Позволяет оценить вовлеченность
- Позволяет сегментировать аудиторию, выделяя тех, для кого продукт создает максимальную ценность.

ManyChat

MAU/DAU

Количества пользователей, которые зашли минимум раз в месяц/день

Минусы критерия:

- Легко меняться без изменения самого продукта.
- Сильно отличаться для продуктов из разных категорий.

MAU/DAU

Допустим, у вас dating приложение

Ваша метрика DAU, позволяет опредлить сегменты активных пользователей, которые вы можете конвертировать в деньги

ManyChat

Retention

- Смотрят на промежутке времени : 7, 21 день
- Обычно рассматривают по ключевому действию (допустим, отправка сообщения в случае LiveChat ManyChat)
- Сколько пользователей из пришедших продолжают совершать действия, спустя промежуток времени

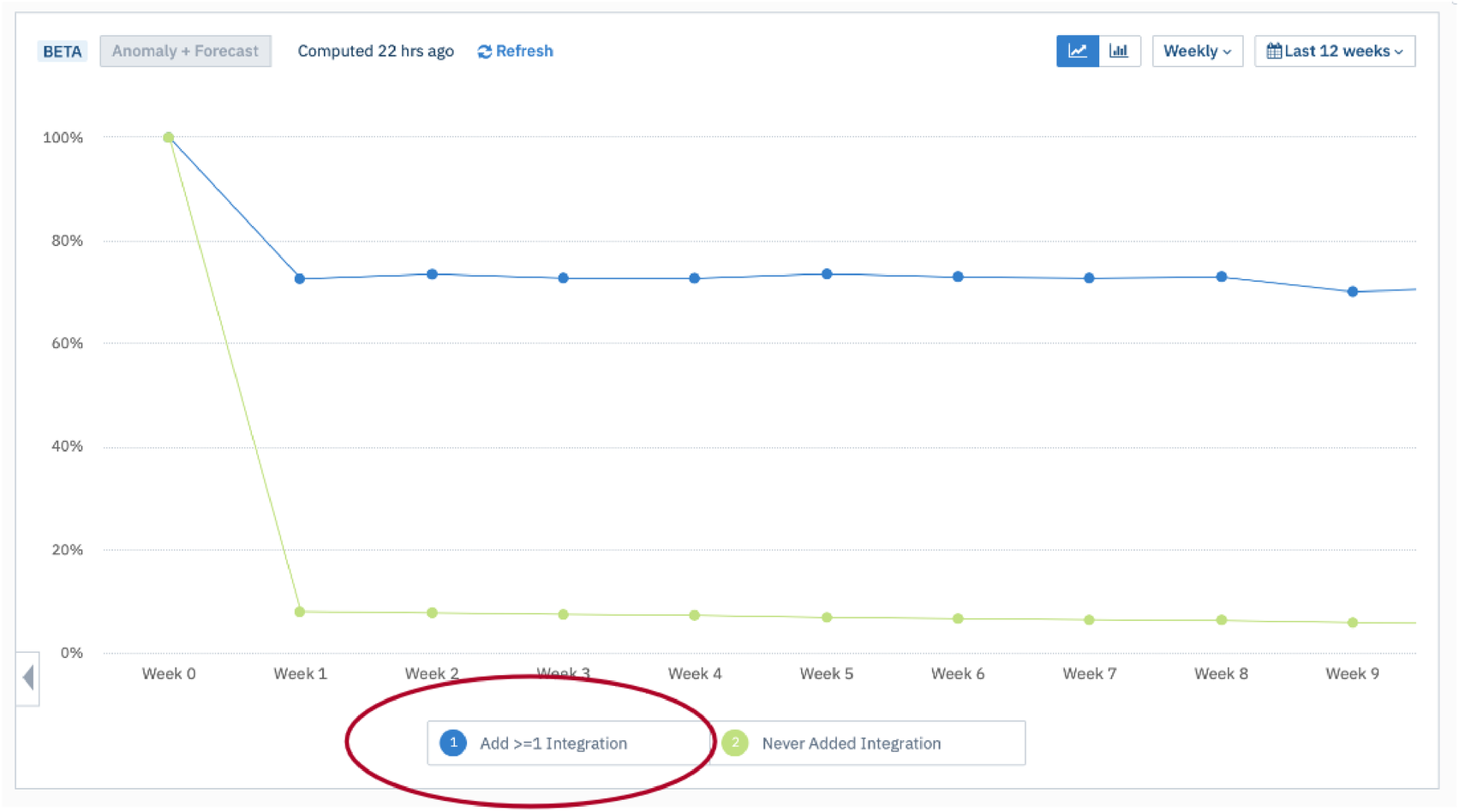
ManyChat

Retention

Чем полезно:

- Сигнал того, находят ли пользователи ценность в вашем приложении
- Часто ли используют при A/B, выбирать вариант с большим показателем

ManyChat


BETA
Anomaly + Forecast
Computed 22 hrs ago
Refresh
Weekly
Last 12 weeks
100%
80%
60%
40%
20%
0%
Week 0
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
1
Add >=1 Integration
2
Never Added Integration

ManyChat


BETA
Anomaly + Forecast
Computed 22 hrs ago
Refresh
Weekly
Last 12 weeks
100%
80%
60%
40%
20%
0%
Week 0
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
1
Add >=1 Integration
2
Never Added Integration

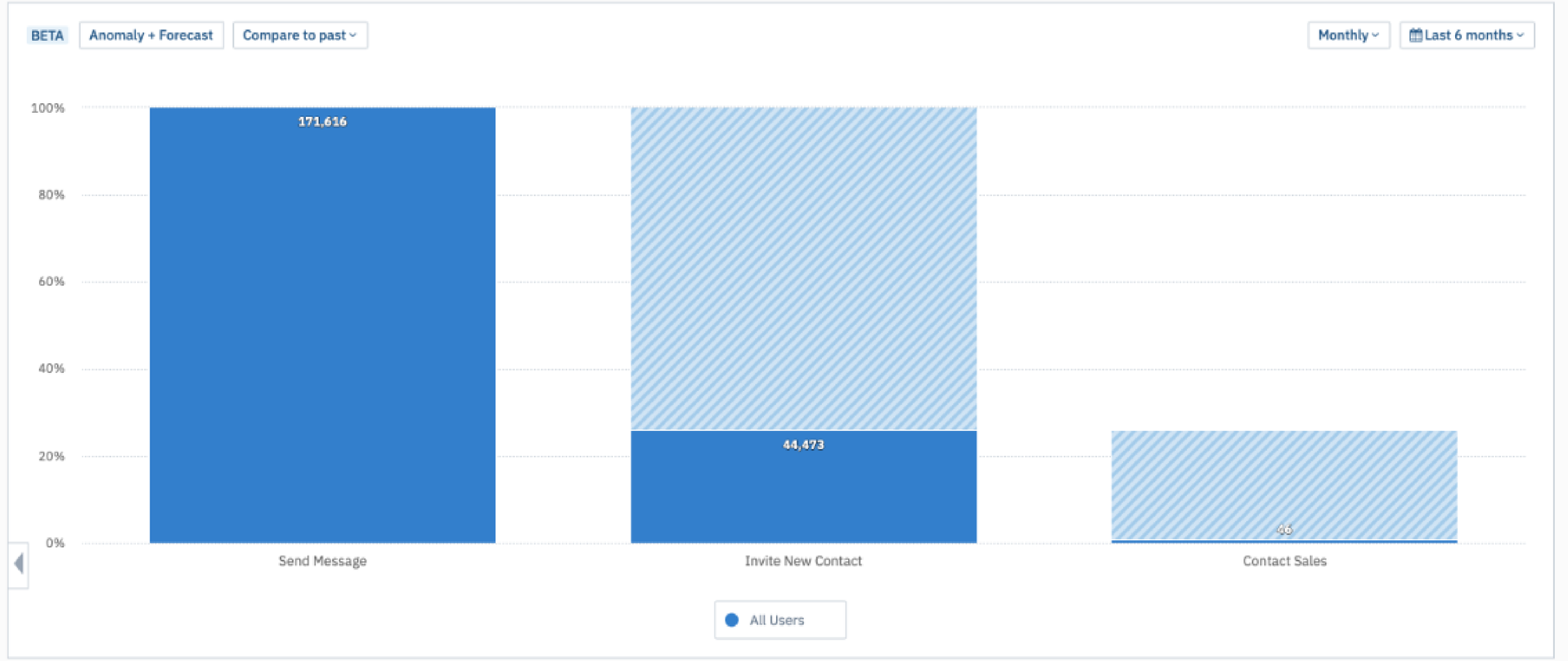
ManyChat

Воронки

Цепочка связанных событий

- Позволяет оценить, сколько пользователей отсеялось на каждом этапе
- Позволяет найти узкие места в сценарии
- Отследить ценный, но труднодоступный функционал

ManyChat


BETA
Anomaly + Forecast
Compare to past
Monthly
Last 6 months
100%
80%
60%
40%
20%
0%
171,616
44,473
46
Send Message
Invite New Contact
Contact Sales
All Users

ManyChat

Воронки

- Цепочка связанных событий
- Позволяет оценить, сколько пользователей отсеялось на каждом этапе
- Позволяет найти узкие места в сценарии
- Отследить ценный, но труднодоступный функционал

ManyChat

Зачем мне это,
я же only developer, after all

- Аналитика влияет на принятие решений
- Правильные решения → компания получает больше денег
- Деньги → рост компании, команды и вас персонально

ManyChat

Где собирать аналитику

- Front
- Back

ManyChat

Почему не только front

Front не надежен:

- События могут не дойти из-за ошибок сети
- Не полный контекст события
- Не все события вашего сервиса связаны с UI

ManyChat

Front + Back

- Позволяет строить кросс-сценарии
- Back хорошо фиксирует бизнес-сценарии
- Front – взаимодействия клиента с продуктом

ManyChat

Как можно отправить аналитику с мобилки

- Amplitude
- Firebase
- Flurry
- ...
- Написать свою аналитику

ManyChat

Лучше - больше

- Все врут
- Несколько аналитик позволят вам минимизировать ошибку

ManyChat

ManyChat

- Firebase
- Свою аналитику

ManyChat

Как добавляется аналитика?
И почему это больно?

ManyChat

На примере ManyChat

- Два нативных приложения
- Разница в разработке полгода
- Много фич повторяется на другой платформе с улучшениями
- На каждой платформе своя кодовая абстракция над аналитикой

ManyChat

Как добавляется аналитика

- Кто-то придумывает события и документирует
- Разработчик при решении задачи смотрит док и руками перебивает код
- Кто-то проверяет это перед релизом
- И все это x2 (для обеих платформ)

ManyChat

Что может пойти не так

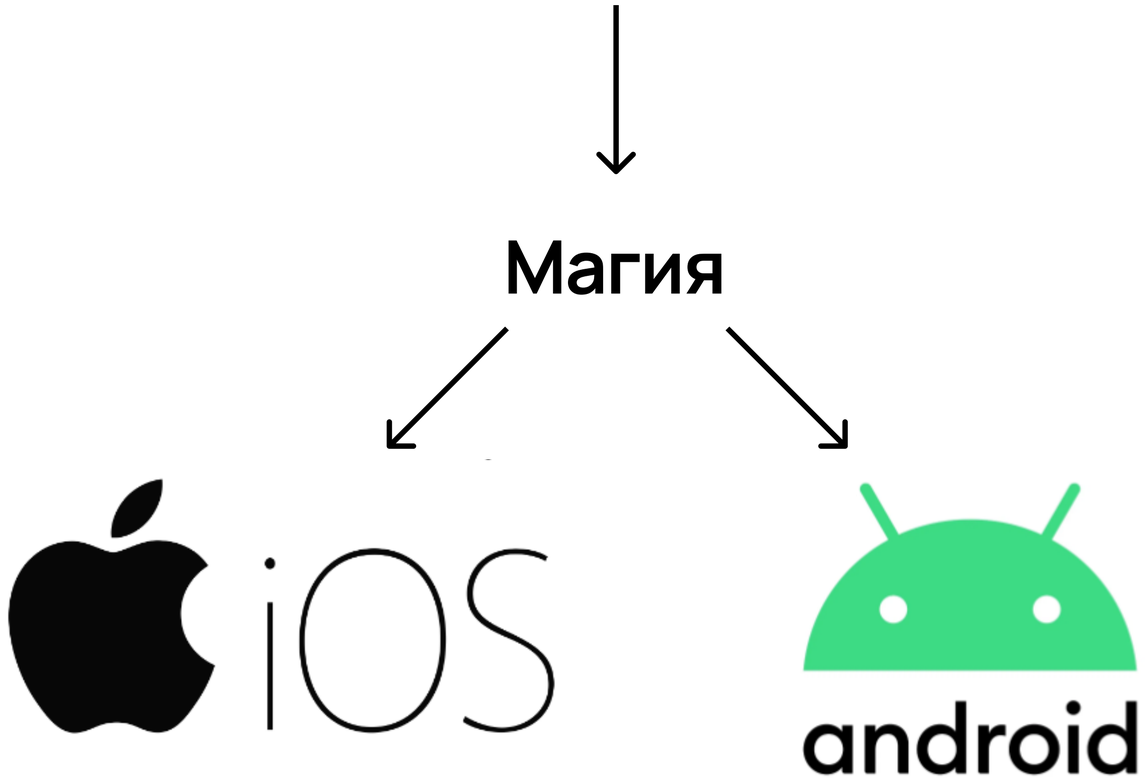
- Легко ошибиться при переносе
- На код ревью можно не заметить очепятки/забытые параметры
- Надо не забыть тщательно проверить тестировщикам
- Может пойти не так на любой платформе

ManyChat

А что, если будет только один источник правды?

ManyChat

Список событий



ManyChat

Что хотим получить?

```
// Где-то в коде

let event = DialogList.CloseDialog(accountID: accountID, subID: subID)
analytics.send(event: event)
```

ManyChat

Что хотим получить?

```
let accountID = AccountIDParam(value: ...)

...

let event = DialogList.CloseDialog(accountID: accountID, subID: subID)
analytics.send(event: event)
```

ManyChat

На обеих платформах

ManyChat

Составим план?

- Выбираем формат хранилища событий
- Выкачиваем данные из хранилища событий
- Превращаем в код
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах
- Подключаем привычным для платформы способом

ManyChat

Составим план?

- Выбираем формат хранилища событий
- Выкачиваем данные из хранилища событий
- Превращаем в код
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах
- Подключаем привычным для платформы способом

Скрипт

ManyChat

Скрипт

- Можно написать на любом языке
- Мы выбрали Python: простой, много библиотек, аналитики на нем пишут
- В докладе мы будем фокусироваться на проектировании и алгоритмах
- Код будет в конце на github

А как данные выглядят?

- Удобный формат для работы с данными как в интерфейсе, так и в коде – таблица
- У 99% табличных сервисов есть API, позволяющее выгрузить данные
- В ManyChat мы используем Google Таблицы
- Но алгоритм для вашего сервиса вряд ли будет отличаться

ManyChat

Как мне выбрать формат?

- Поговорите с аналитиками/тем человеком, который будет хранилищем пользоваться
- **Договоритесь о нейминге (Например, Экран/Компонент___элемент___действие)**
- Dialog___Close___Pressed, ___ потому что Firebase не захотел есть
- Проговорите версинирование
- **Обсудить набор обязательных параметров**
- **Как связывать события и параметры**

ManyChat

Табличка событий в ManyChat

general metadata						дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентфкатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
					int	timestamp	string	int	string	int	int	int	int
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**	**event_id**	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021	x	x	x	x	x	x	x	x	
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021		x	x	x	x	x	x	x	x	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021		x	x	x	x	x	x	x		x

ManyChat

А как данные в таблице выглядят?

general metadata				
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021	

ManyChat

А как данные в таблице выглядят?

general metadata				
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021	

уникальный id, альтернатива имени

ManyChat

А как данные в таблице выглядят?

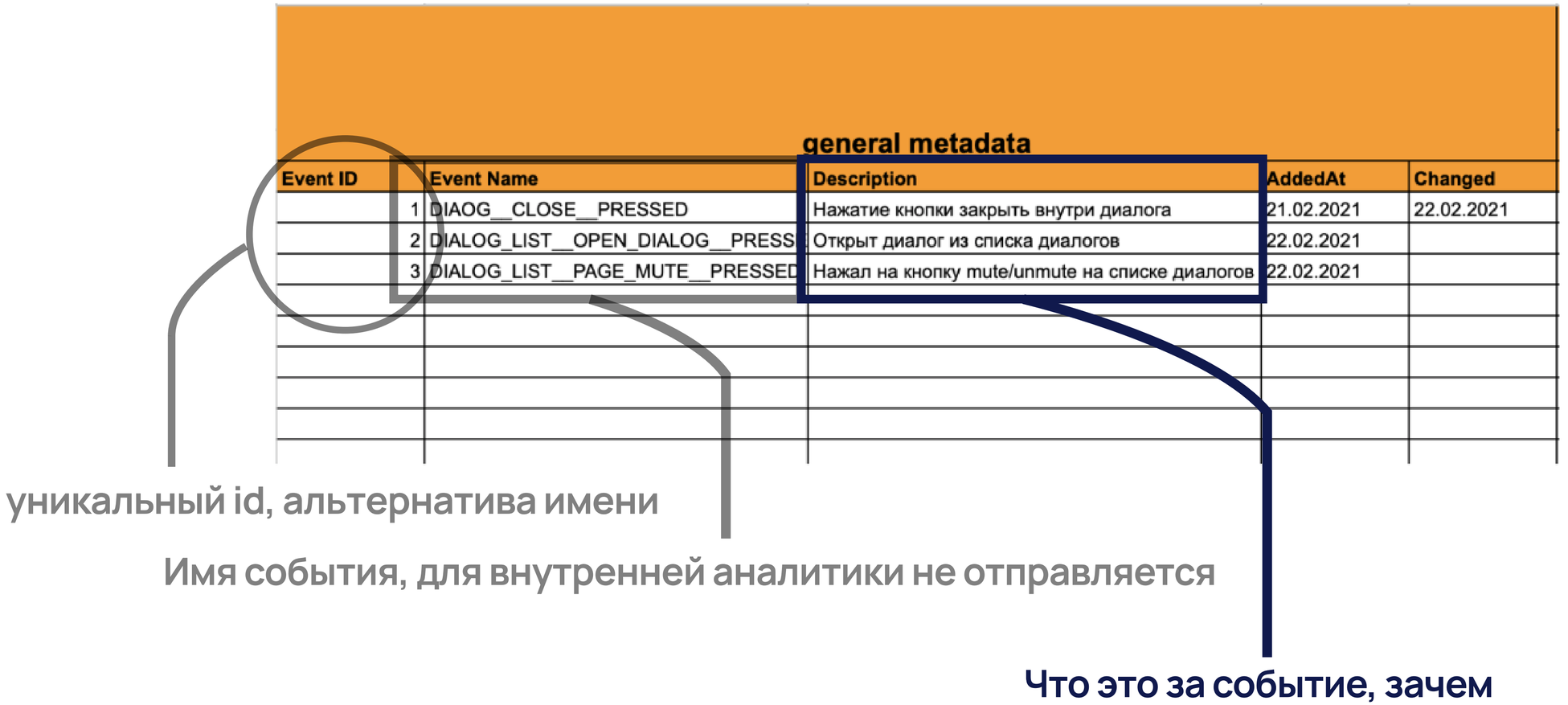
general metadata				
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021	

уникальный id, альтернатива имени

Имя события, для внутренней аналитики не отправляется

ManyChat

А как данные в таблице выглядят?



	general metadata			
Event ID	Event Name	Description	AddedAt	Changed
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021
2	DIALOG_LIST__OPEN_DIALOG__PRESS	Открыт диалог из списка диалогов	22.02.2021	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021	

уникальный id, альтернатива имени

Имя события, для внутренней аналитики не отправляется

Что это за событие, зачем

ManyChat

А как данные в таблице выглядят?

ManyChat

А как данные в таблице выглядят?

	дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентифкатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	timestamp	string	int	string	int	int	int	int
event_id	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x		x

«Бесконечный» список параметров

ManyChat

А как данные в таблице выглядят?

	дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентифкатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	timestamp	string	int	string	int	int	int	int
event_id	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x		x

«Бесконечный» список параметров

6 первый параметров не отправляются вместе с событием

ManyChat

А как данные в таблице выглядят?

Ожидаемый тип

	дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентификатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	timestamp	string	int	string	int	int	int	int
event_id	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x		x

«Бесконечный» список параметров

6 первый параметров не отправляются вместе с событием

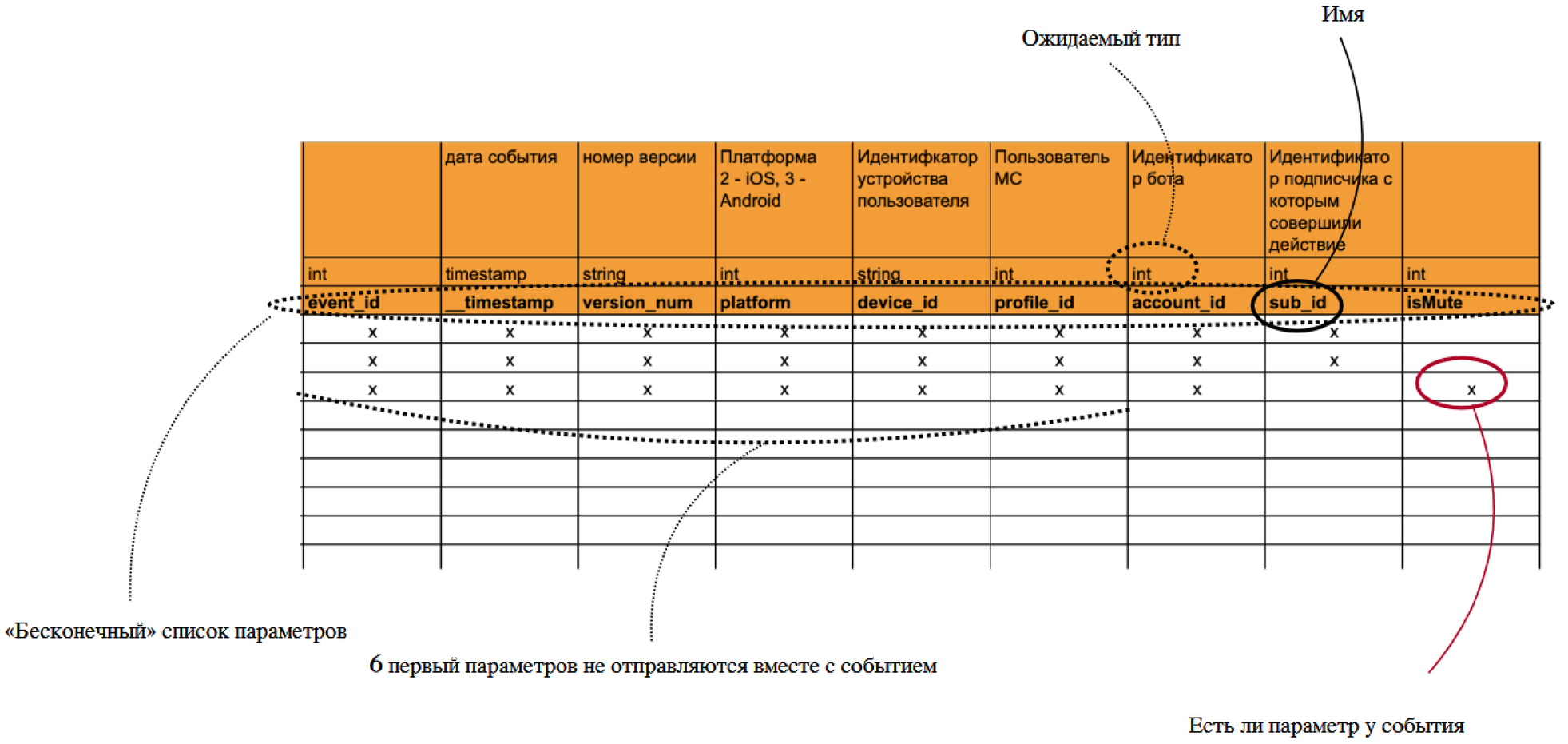
ManyChat

А как данные в таблице выглядят?

	дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентфикатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	timestamp	string	int	string	int	int	int	int
event_id	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x		x

ManyChat

А как данные в таблице выглядят?



	дата события	номер версии	Платформа 2 - iOS, 3 - Android	Идентфкатор устройства пользователя	Пользователь MC	Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	timestamp	string	int	string	int	int	int	int
event_id	**__timestamp**	**version_num**	**platform**	**device_id**	**profile_id**	**account_id**	**sub_id**	**isMute**
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x		x

ManyChat

Но у меня нет такой таблицы : (

- Определяем формат
- Замораживаем добавление аналитики
- Переносим события с источника истины/приложения
- Привыкаем к процессу table first

ManyChat

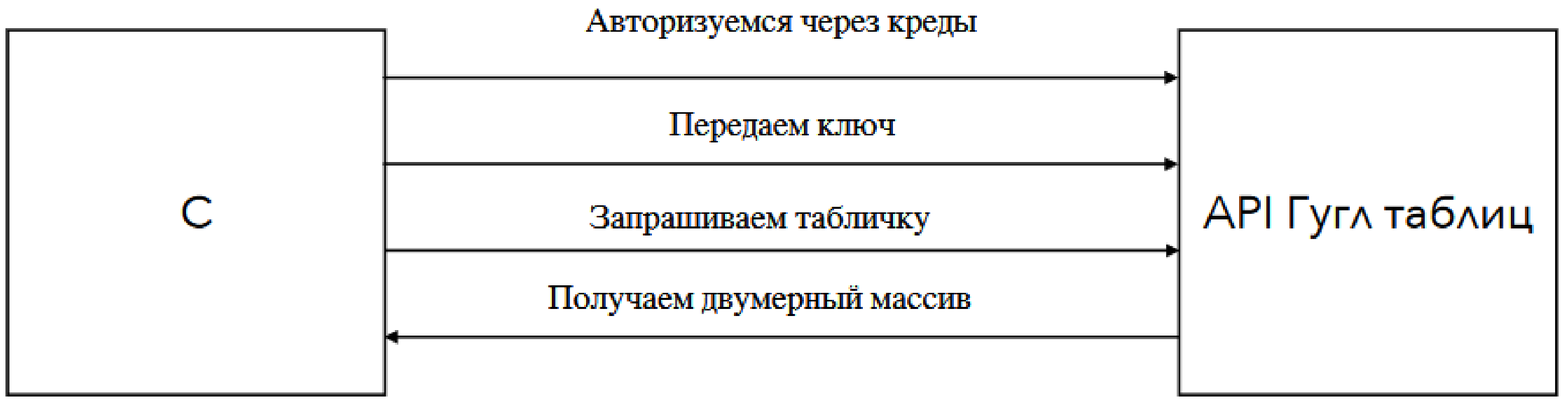
План

- Выбираем формат хранилища событий ✅
- Выкачиваем данные из хранилища событий
- Превращаем в код
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах
- Подключаем привычным для платформы способом

ManyChat

Достаем данные

- Настраиваем доступ к API
- Получаем креды и ключ



ManyChat

Достаем данные

array[][] ==

	general metadata					дата события
					int	timestamp
Event ID	Event Name	Description	AddedAt	Changed	event_id	__timestamp
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021	x	x
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021		x	x
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021		x	x

array[2][1]

ManyChat

Достаем данные

general metadata						дата события
					int	timestamp
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**	**event_id**	**__timestamp**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021	x	x
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021		x	x
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021		x	x

- Все ID – tableContent[2...][0]
- Все Event Name – tableContent[2...][0]
- Все имена параметров – tableContent[3...][12...]
- Все типы параметров – tableContent[2...][12...]

ManyChat

План

- Выбираем формат хранилища событий ✅
- Выкачиваем данные из хранилища событий ✅
- Превращаем в код
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах
- Подключаем привычным для платформы способом

ManyChat

Превращаем в код

ManyChat

Данные → код

- В каком виде мы будем передавать код?
- Как он будет выглядеть?

В каком виде мы будем передавать код?

- Кросс-платформенная библиотека
- Нативный код

ManyChat

В каком виде мы будем передавать код?

Кросс-платформенность	Две нативные библиотеки
Одинаковый код на обеих платформах	Нативный код, более платформенно-ориентированный
Код работает одинаково	Два скрипта генерации – возможны расхождения

ManyChat

В каком виде мы будем передавать код?

Кросс-платформенность	Две нативные библиотеки
Одинаковый код на обеих платформах	Нативный код, более платформенно-ориентированный
Код работает одинаково	Два скрипта генерации – возможны расхождения
Интересный опыт, может зайти в будущем	

ManyChat

Варианты кросс-платформенности

- React Native
- Flutter
- C++
- Koltin native

ManyChat

Требования к решению

- Только бизнес логика
- UI не нужен

ManyChat

Варианты кросс-платформенности

- C++
- Kotlin native

Kotlin Native

Как выглядят события?

```
let accountID = AccountIDParam(value: ...)

...

let event = DialogList.CloseDialog(accountID: accountID, subID: subID)
analytics.send(event: event)
```

ManyChat

Параметры

```
data class AccountIdParam(override val value: Long)

data class AccountIdParam(override val value: String)
```

Должно ли между ними быть что-то общее?

- Мы хотим работать исключительно с кодом
- Но на бекенд/аналитику будет отправляться строковое имя/ID
- Либо храним строковое значение
- Либо превращаем код обратно в строку

ManyChat

Параметры

```
abstract class EventParam<T> {
    val name: String = classNameToEventParamName()
    abstract val value: T
}
```

Параметры

```
abstract class EventParam<T> {
    val name: String = classNameToEventParamName()
    abstract val value: T
}

abstract class LongParam(override val value: Long) : EventParam<Long>()

abstract class StringParam(override val value: String) : EventParam<String>()
```

Параметры

```
data class AccountIdParam(override val value: Long): LongParam(value)

data class AnotherParam(override val value: String): StringParam(value)
```

Давайте их сгенерим

Идентификатор бота	Идентификатор подписчика с которым совершили действие	
int	int	int
account_id	**sub_id**	**isMute**
x	x	
x	x	
x		x

```
data class AccountIdParam(override val value: Long): LongParam(value)
```

ManyChat

Давайте их сгенерим

Массив имен

Массив типов

ManyChat

Давайте их сгенерим

account_id

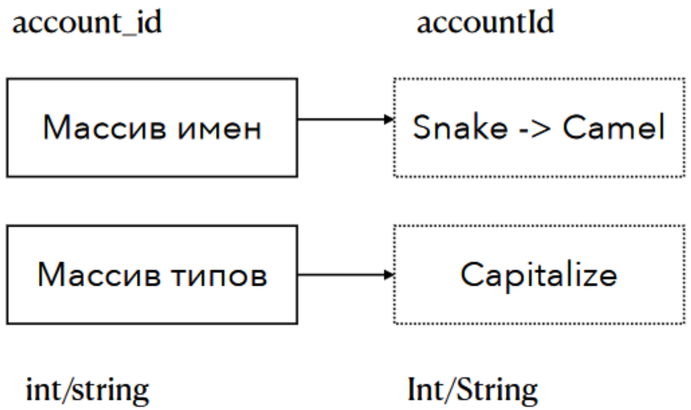
Массив имен

Массив типов

Int/string

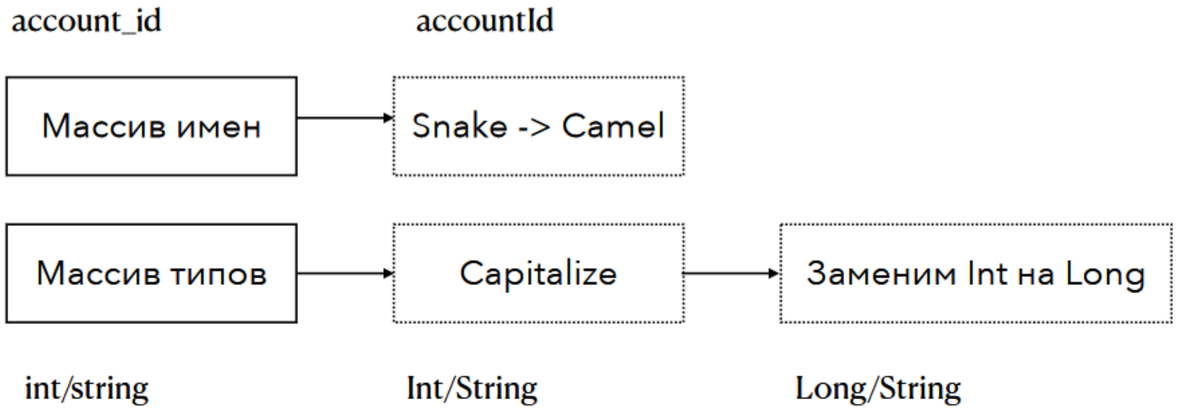
ManyChat

Давайте их сгенерим



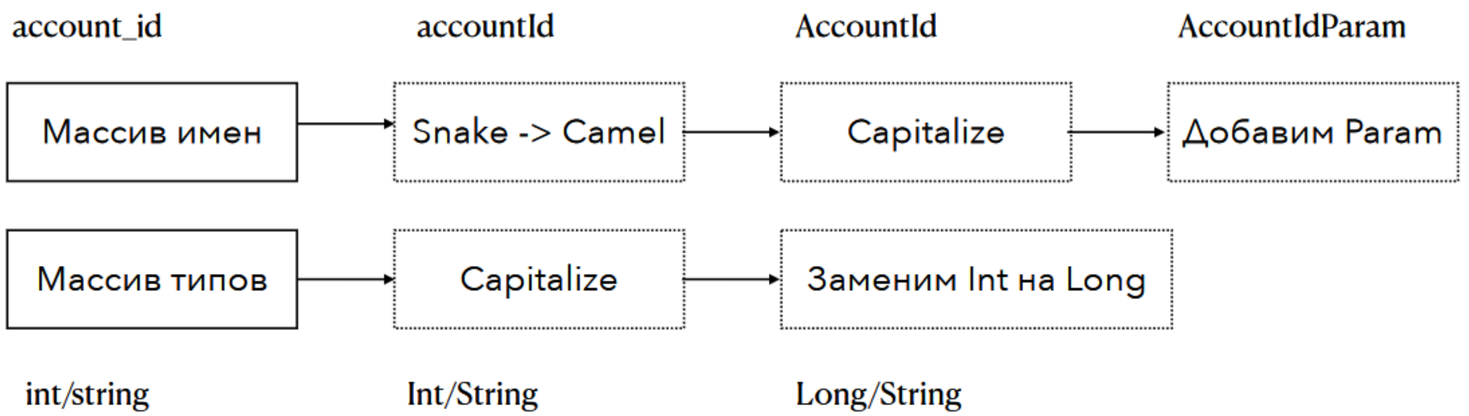
ManyChat

Давайте их сгенерим



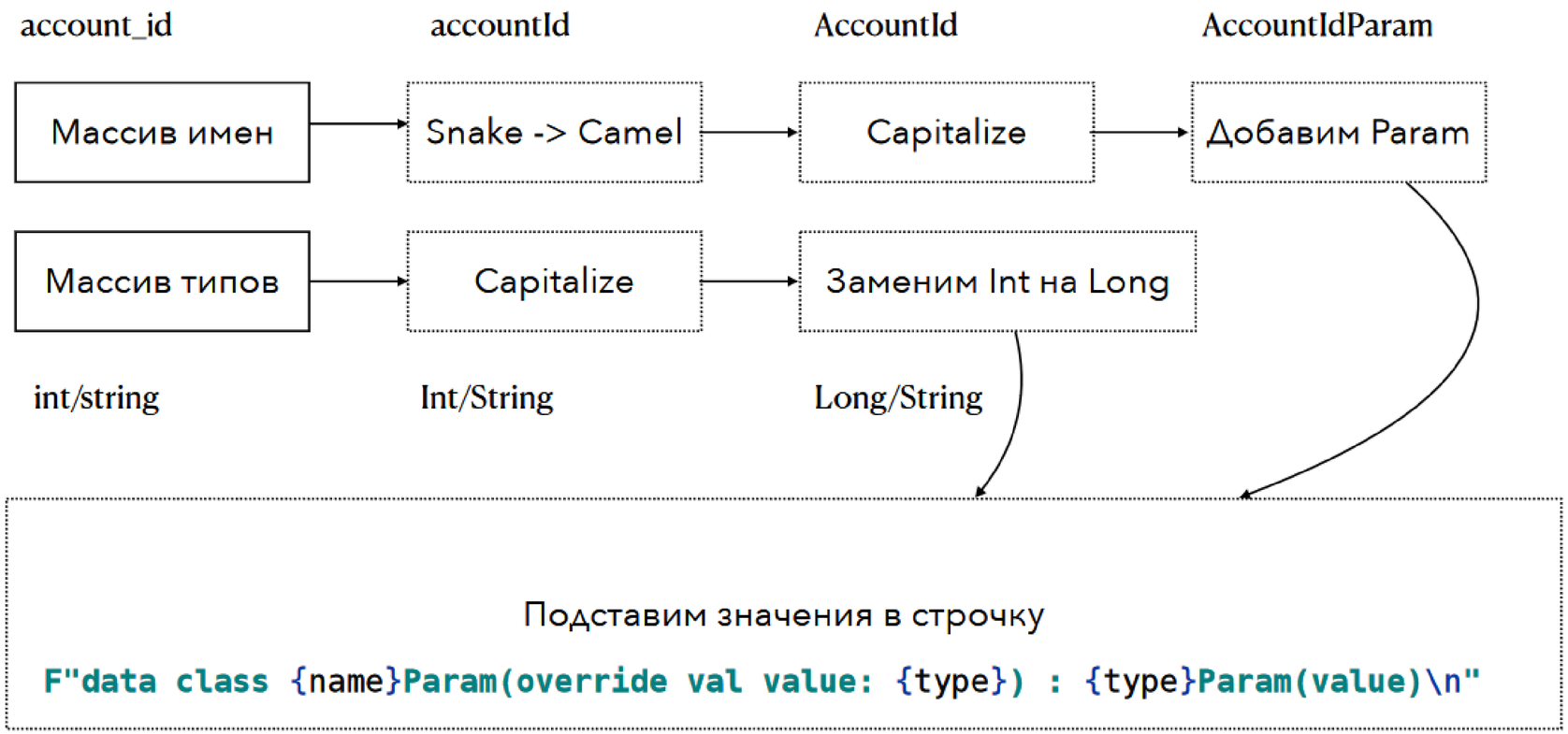
ManyChat

Давайте их сгенерим



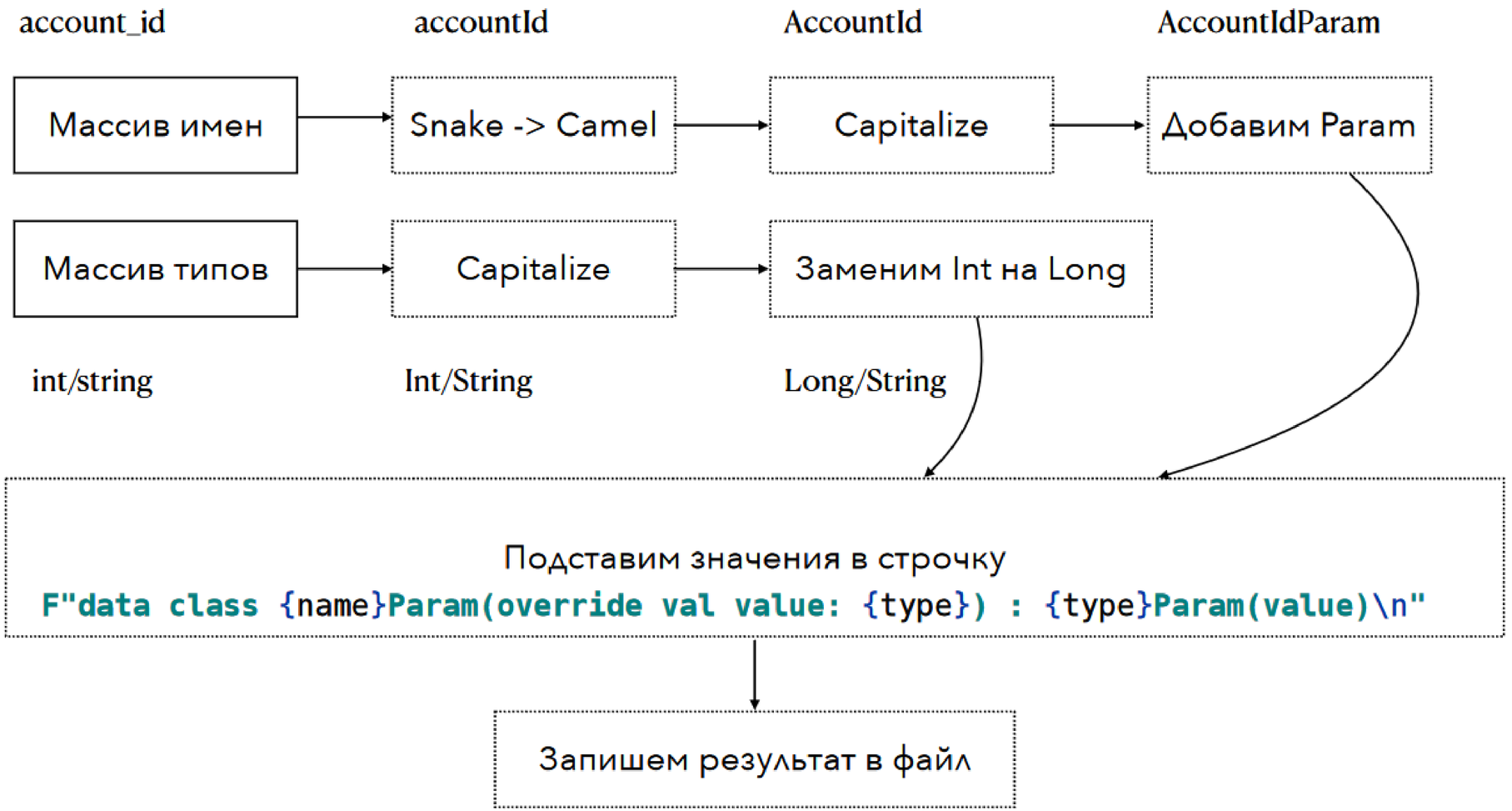
ManyChat

Давайте их сгенерим



ManyChat

Давайте их сгенерим



ManyChat

Давайте их сгенерим

eventParams.generated.kt

```kotlin
package com.mobile.analytics.event

import com.mobile.analytics.event.*

data class AccountIdParam(override val value: Long) : LongParam(value)
data class SubIdParam(override val value: Long) : LongParam(value)
data class ActionParam(override val value: String) : StringParam(value)
data class TimespendParam(override val value: Long) : LongParam(value)
data class PageParam(override val value: Long) : LongParam(value)
data class TotalCountParam(override val value: Long) : LongParam(value)
data class NamespaceParam(override val value: String) : StringParam(value)
data class RunsCountParam(override val value: Long) : LongParam(value)
data class DepthParam(override val value: Long) : LongParam(value)
data class StatusParam(override val value: String) : StringParam(value)
data class SymbolsCountParam(override val value: Long) : LongParam(value)
data class FieldIdParam(override val value: Long) : LongParam(value)
```

ManyChat

Теперь к самим событиям

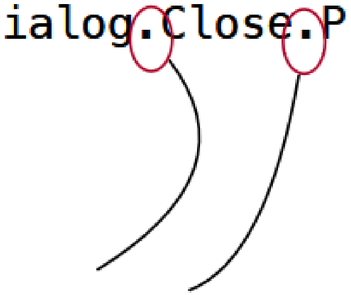
```
val event = Dialog.Close.Pressed(accountIdParam, subIdParam)
```

```
let event = Dialog.Close.Pressed(accountIdParam: accountIdParam, subIdParam: subIdParam)
```

ManyChat

Теперь к самим событиям

```
val event = Dialog.Close.Pressed(accountIdParam, subIdParam)

let event = Dialog.Close.Pressed(accountIdParam: accountIdParam, subIdParam: subIdParam)
```



Неймспейсы

ManyChat

Namespace

```
sealed class Dialog: Event() {
    sealed class Close : Dialog() {
        class Pressed(accountIdParam: AccountIdParam, subIdParam: SubIdParam) : Close() {
            override val id = 1
            override val params = arrayOf(accountIdParam, subIdParam).toMap()
        }
    }
}
```

ManyChat

Namespace

- Повышает читаемость
- Сложнее генерить

Базовый класс

```
abstract class Event {
    abstract val id: Long
    abstract val params: Map<String, Any>
    val name: EventName = classNameToEventName()
}
```

Исходные данные

general metadata				
Event ID	**Event Name**	**Description**	**AddedAt**	**Changed**
1	DIAOG__CLOSE__PRESSED	Нажатие кнопки закрыть внутри диалога	21.02.2021	22.02.2021
2	DIALOG_LIST__OPEN_DIALOG__PRESSE	Открыт диалог из списка диалогов	22.02.2021	
3	DIALOG_LIST__PAGE_MUTE__PRESSED	Нажал на кнопку mute/unmute на списке диалогов	22.02.2021	

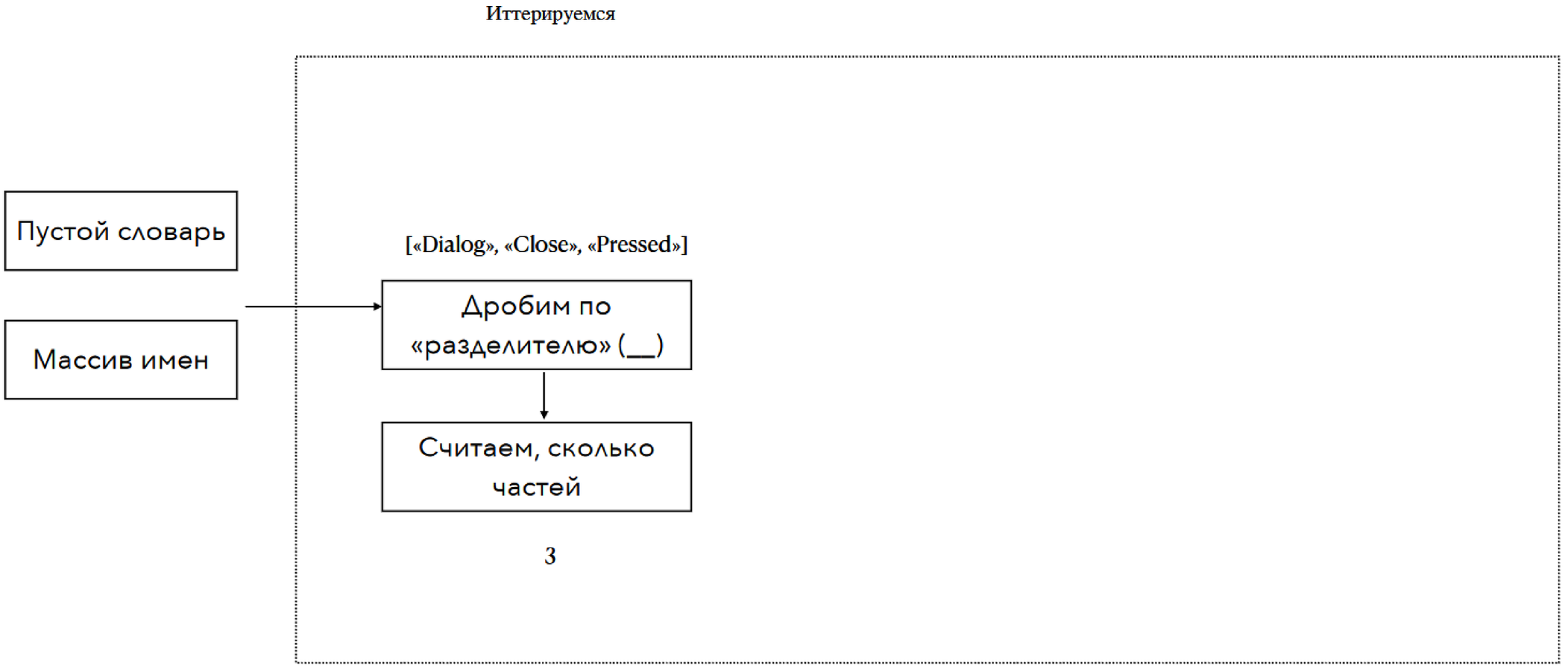
ManyChat

Namespace

Пустой словарь

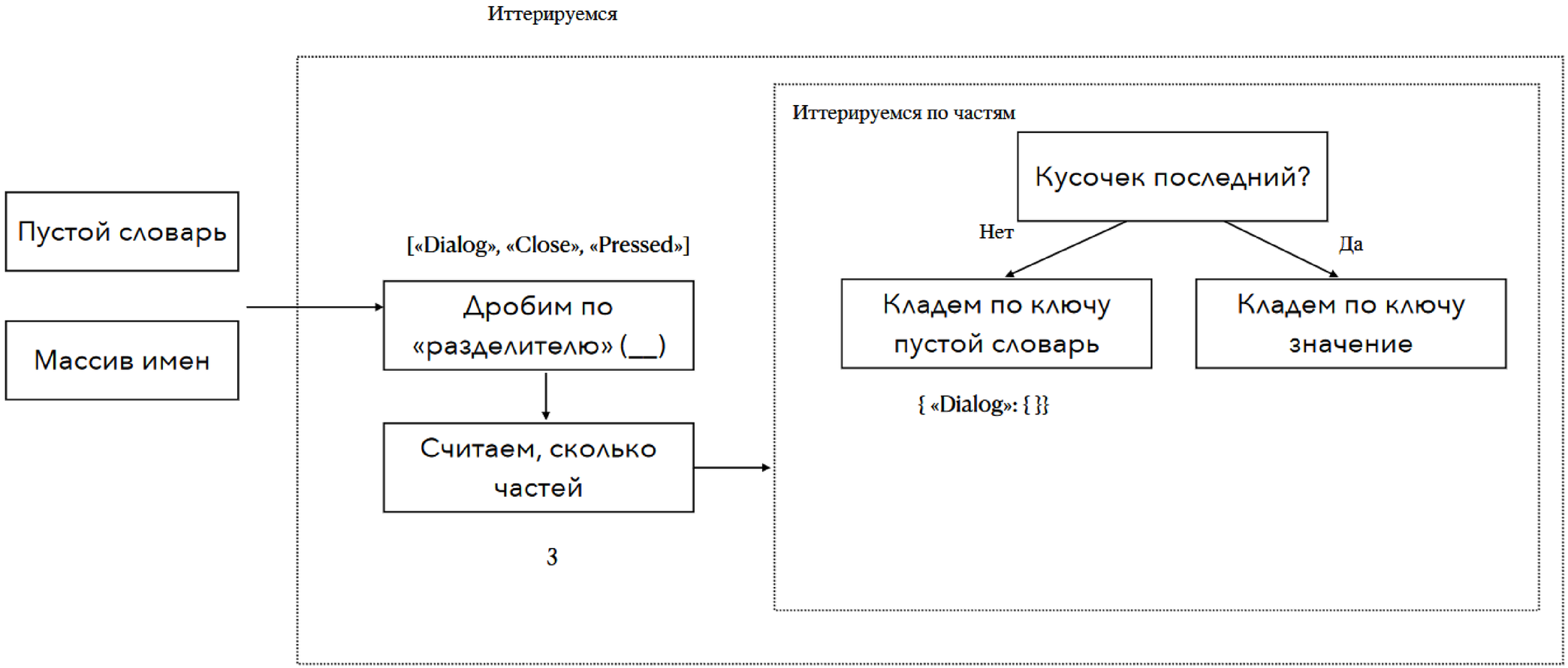
Массив имен

Namespace



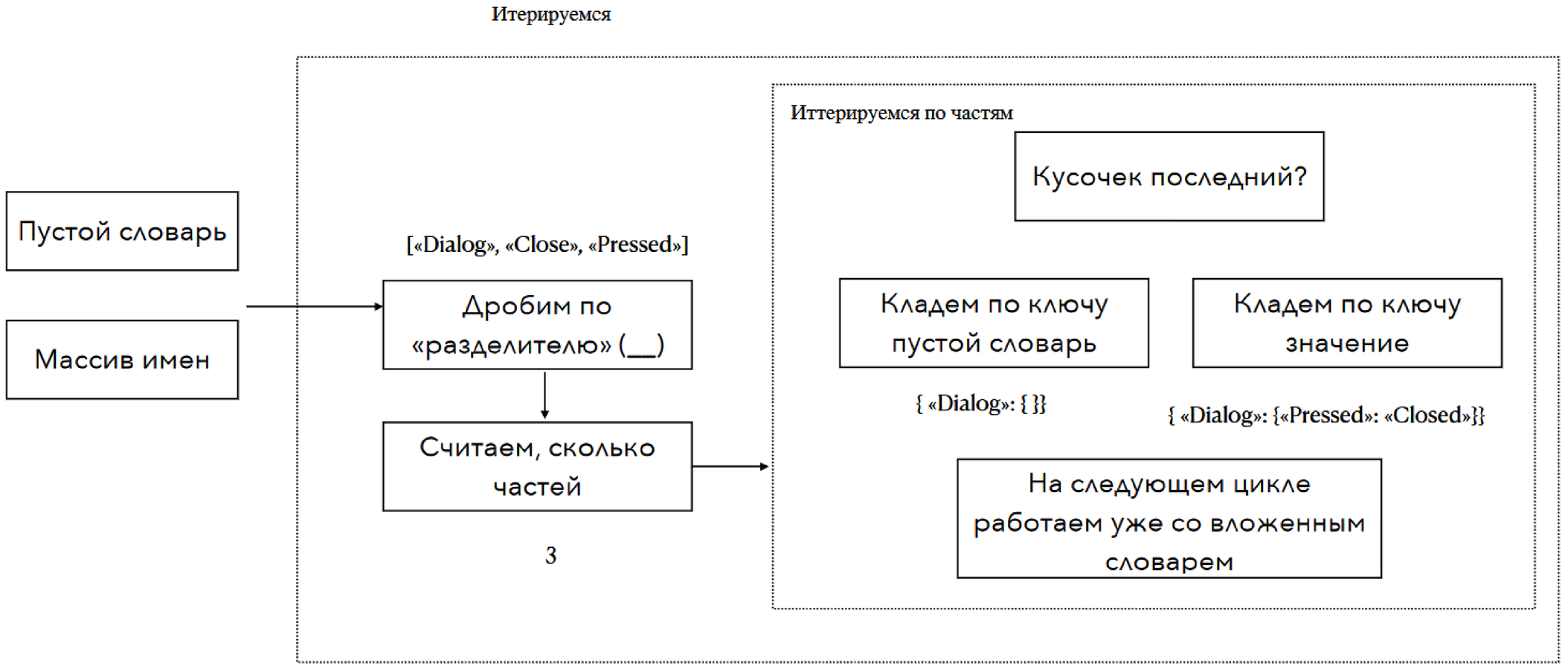
ManyChat

Namespace



ManyChat

Namespace



ManyChat

Namespace

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

ManyChat

Генерим события

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

```
sealed class Dialog: Event() {
   – рекурсивный вызов функции –
```

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

```
sealed class Dialog: Event() {
— рекурсивный вызов функции —
    sealed class Close : Dialog() {
```

Глубина * « «

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

```
sealed class Dialog: Event() {
— рекурсивный вызов функции —
    sealed class Close : Dialog() {
— генерация параметров, генерация события
```

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
     },
     …
  },
  …
}
```

```
sealed class Dialog: Event() {
– рекурсивный вызов функции –
    sealed class Close : Dialog() {
– генерация параметров, генерация события
        class Pressed(
              accountIdParam: AccountIdParam,
              subIdParam: SubIdParam) : Close() {
```

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
     },
     …
  },
  …
}
```

```
sealed class Dialog: Event() {
– рекурсивный вызов функции –
    sealed class Close : Dialog() {
– генерация параметров, генерация события
        class Pressed(
              accountIdParam: AccountIdParam,
              subIdParam: SubIdParam) : Close() {

 –– генерируем еще события, если есть
```

ManyChat

Генерим события

```
{
  «Dialog»: {
    «Close»: {
      «Pressed»
      …
    },
    …
  },
  …
}
```

```
sealed class Dialog: Event() {
– рекурсивный вызов функции –
    sealed class Close : Dialog() {
– генерация параметров, генерация события
        class Pressed(
              accountIdParam: AccountIdParam,
              subIdParam: SubIdParam) : Close() {

 —— генерируем еще события, если есть

    }
```

ManyChat

Corner Case

- Нельзя завессти события вида DIALOG___CLOSE и DIALOG___CLOSE___PRESSES

ManyChat

Победа

- Файлы с событиями
- Файлы с паарметрами
- Файлы с базовыми классами
- Пора завернуть в библиотеку

ManyChat

План

- Выбираем формат хранилища событий ✅
- Выкачиваем данные из хранилища событий ✅
- Превращаем в код ✅
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах
- Подключаем привычным для платформы способом

ManyChat

Что хотим от библиотечки

- Подключается привычным образом: Cocoapods и MavenRepository
- Собрать можно одной кнопкой
- Можно организовать это через git submodules сгенерацией кода в билд-фазе
- Но нам бы хотелось иметь стабильный бинарь и сохранить единственный способ подключения зависимостей

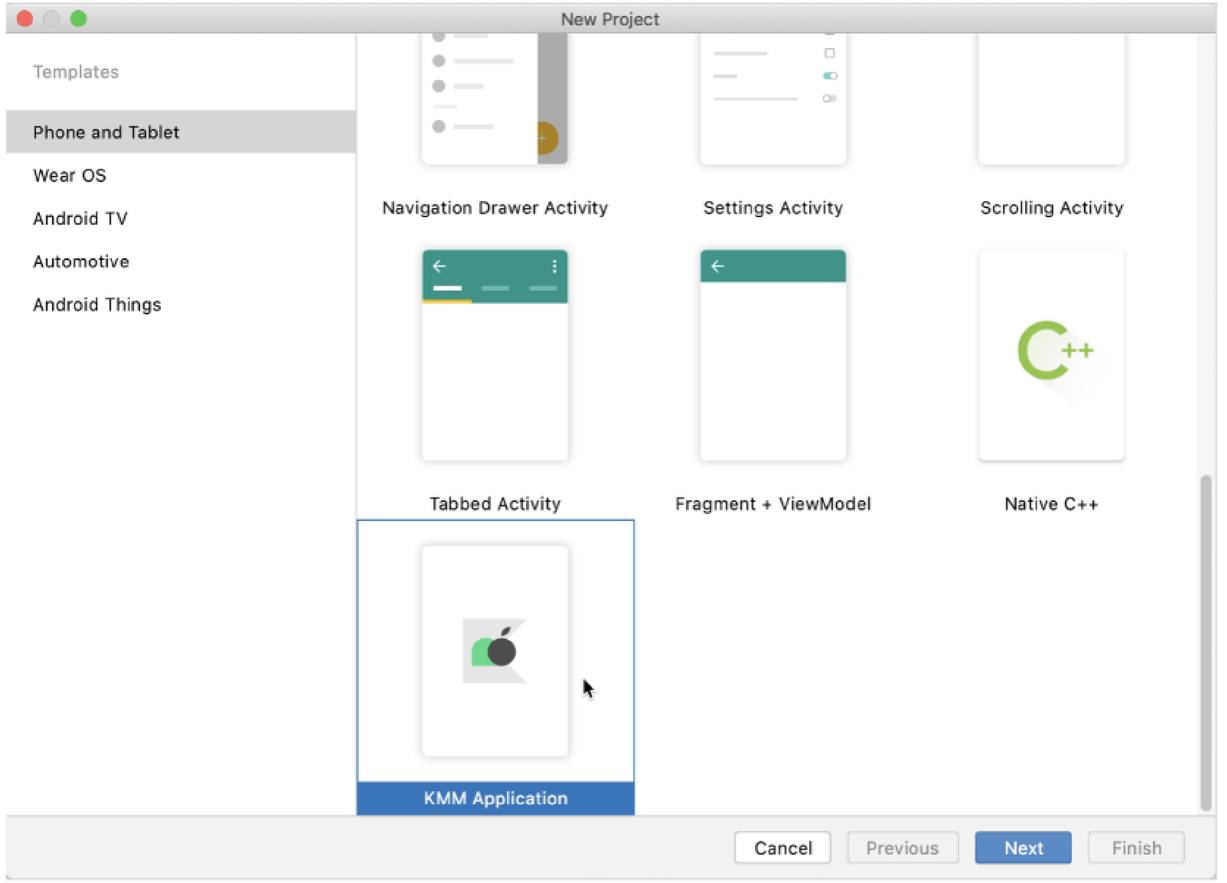
ManyChat

Создаем KMM проект

- Andriod Studio
- Плагинот Jetbrains — Kotlin Multiplatform Mobile

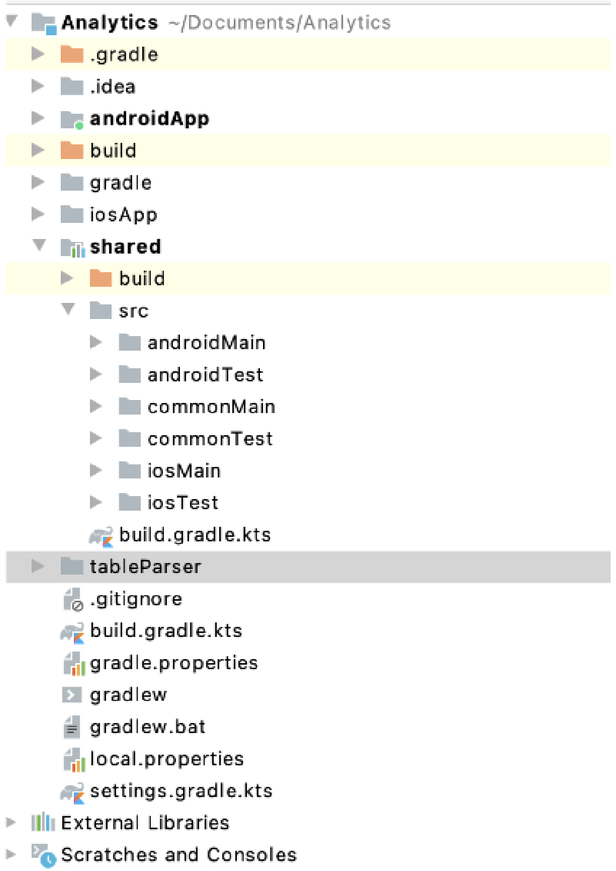
ManyChat

Создаем проект Koltin Native



ManyChat

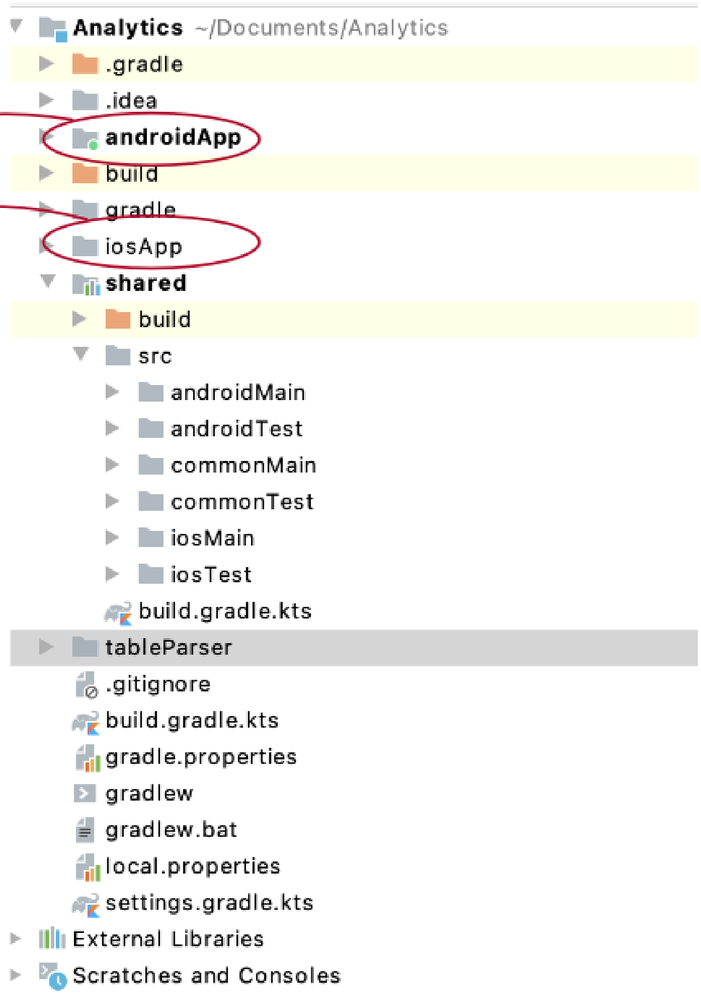
Создаем KMM проект



ManyChat

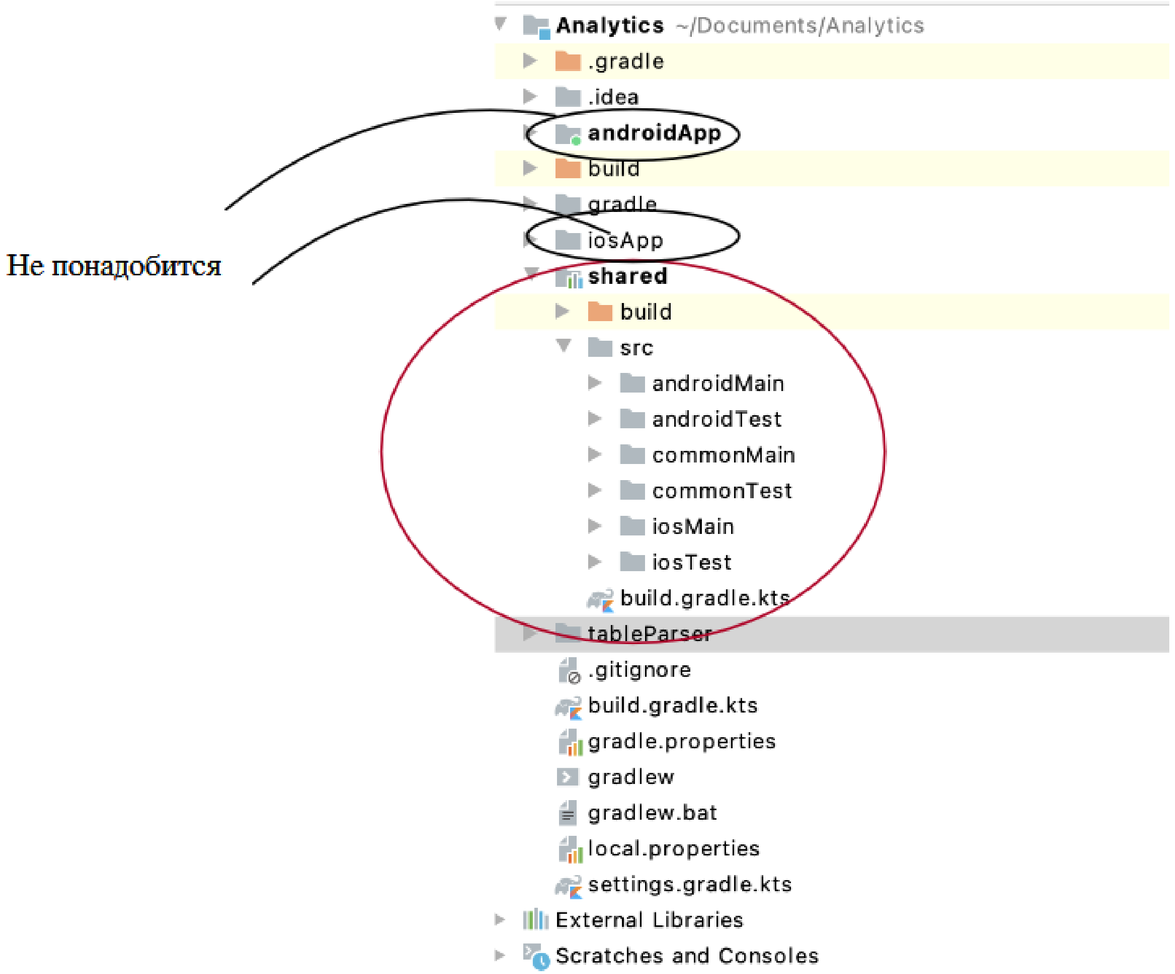
Создаем KMM проект

Не понадобится



ManyChat

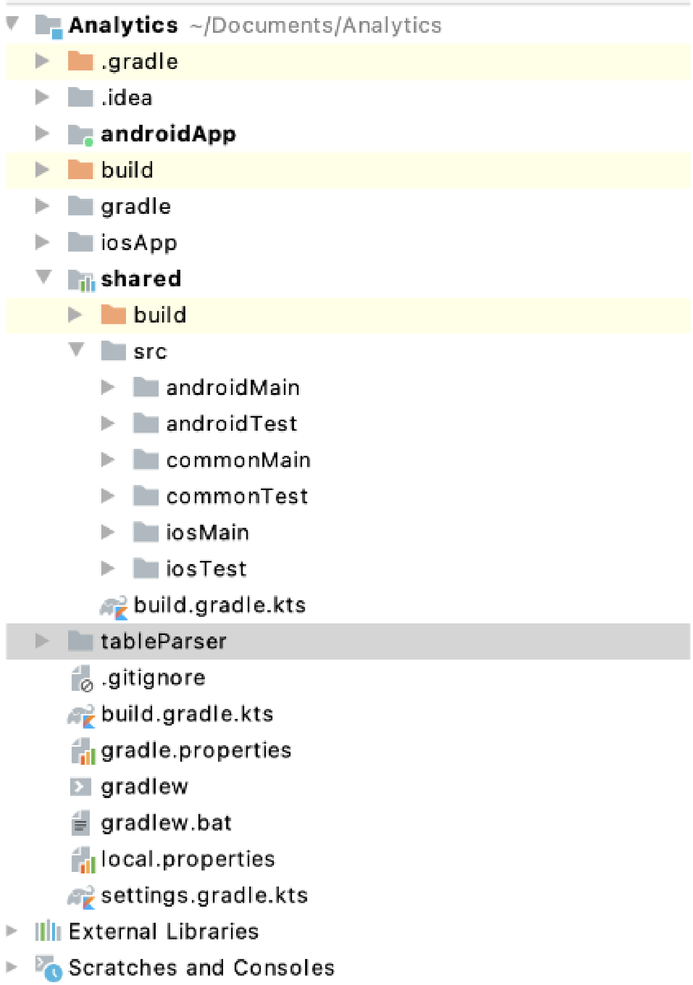
Создаем KMM проект



ManyChat

Создаем KMM проект

- Common — общий код. Сюда сложим наши события
- AndroidMain и iOSMain — код с доступом к платформенным библиотекам
- Build.gradle.kts — правила сборки



ManyChat

Пара слов про Gradle

- Формирует граф задач
- Задачи можете писать вы, а можно подключать из плагинов

Что уже есть

```
import org.jetbrains.kotlin.gradle.plugin.mpp.KotlinNativeTarget

plugins {
    kotlin("multiplatform")
    id("com.android.library")
}
```

Что уже есть

```
kotlin {
    android()
    ios {
        binaries {
            framework {
                baseName = "shared"
            }
        }
    }
    sourceSets {
        val commonMain by getting
        val commonTest by getting {
            dependencies {
                implementation(kotlin("test-common"))
                implementation(kotlin("test-annotations-common"))
            }
        }
        val androidMain by getting {
            dependencies {
                implementation("com.google.android.material:material:1.2.1")
            }
        }
        val androidTest by getting {
            dependencies {
                implementation(kotlin("test-junit"))
                implementation("junit:junit:4.13")
            }
        }
        val iosMain by getting
        val iosTest by getting
    }
}
```

ManyChat

Что уже есть

```
kotlin {
    android()
    ios {
        binaries {
            framework {
                baseName = "shared"
            }
        }
    }
    sourceSets {
        val commonMain by getting
        val commonTest by getting {
            dependencies {
                implementation(kotlin("test-common"))
                implementation(kotlin("test-annotations-common"))
            }
        }
        val androidMain by getting {
            dependencies {
                implementation("com.google.android.material:material:1.2.1")
            }
        }
        val androidTest by getting {
            dependencies {
                implementation(kotlin("test-junit"))
                implementation("junit:junit:4.13")
            }
        }
        val iosMain by getting
        val iosTest by getting
    }
}
```

ManyChat

Что уже есть

```
kotlin {
    android()
    ios {
        binaries {
            framework {
                baseName = "shared"
            }
        }
    }
    sourceSets {
        val commonMain by getting
        val commonTest by getting {
            dependencies {
                implementation(kotlin("test-common"))
                implementation(kotlin("test-annotations-common"))
            }
        }
        val androidMain by getting {
            dependencies {
                implementation("com.google.android.material:material:1.2.1")
            }
        }
        val androidTest by getting {
            dependencies {
                implementation(kotlin("test-junit"))
                implementation("junit:junit:4.13")
            }
        }
        val iosMain by getting
        val iosTest by getting
    }
}
```

ManyChat

Что уже есть

```
android {  this: LibraryExtension
    compileSdkVersion( apiLevel: 29)
    sourceSets["main"].manifest.srcFile( srcPath: "src/androidMain/AndroidManifest.xml")
    defaultConfig {  this: DefaultConfig
        minSdkVersion( minSdkVersion: 24)
        targetSdkVersion( targetSdkVersion: 29)
    }
}
```

ManyChat

Что уже есть

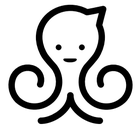
```
android {  this: LibraryExtension
    compileSdkVersion( apiLevel: 29)
    sourceSets["main"].manifest.srcFile( srcPath: "src/androidMain/AndroidManifest.xml")
    defaultConfig {  this: DefaultConfig
        minSdkVersion( minSdkVersion: 24)
        targetSdkVersion( targetSdkVersion: 29)
    }
}
```

```
val packForXcode by tasks.creating(Sync::class) {
    group = "build"
    val mode = System.getenv("CONFIGURATION") ?: "DEBUG"
    val sdkName = System.getenv("SDK_NAME") ?: "iphonesimulator"
    val targetName = "ios" + if (sdkName.startsWith("iphoneos")) "Arm64" else "X64"
    val framework = kotlin.targets.getByName<KotlinNativeTarget>(targetName).binaries.getFramework(mode)
    inputs.property("mode", mode)
    dependsOn(framework.linkTask)
    val targetDir = File(buildDir, "xcode-frameworks")
    from({ framework.outputDirectory })
    into(targetDir)
}

tasks.getByName("build").dependsOn(packForXcode)
```

ManyChat

Что уже есть

```
val packForXcode by tasks.creating(Sync::class) {
    group = "build"
    val mode = System.getenv("CONFIGURATION") ?: "DEBUG"
    val sdkName = System.getenv("SDK_NAME") ?: "iphonesimulator"
    val targetName = "ios" + if (sdkName.startsWith("iphoneos")) "Arm64" else "X64"
    val framework = kotlin.targets.getByName<KotlinNativeTarget>(targetName).binaries.getFramework(mode)
    inputs.property("mode", mode)
    dependsOn(framework.linkTask)
    val targetDir = File(buildDir, "xcode-frameworks")
    from({ framework.outputDirectory })
    into(targetDir)
}

tasks.getByName("build").dependsOn(packForXcode)
```



ManyChat

Gradle дает много новых возможностей

Если написать gradle tasks в консоли:

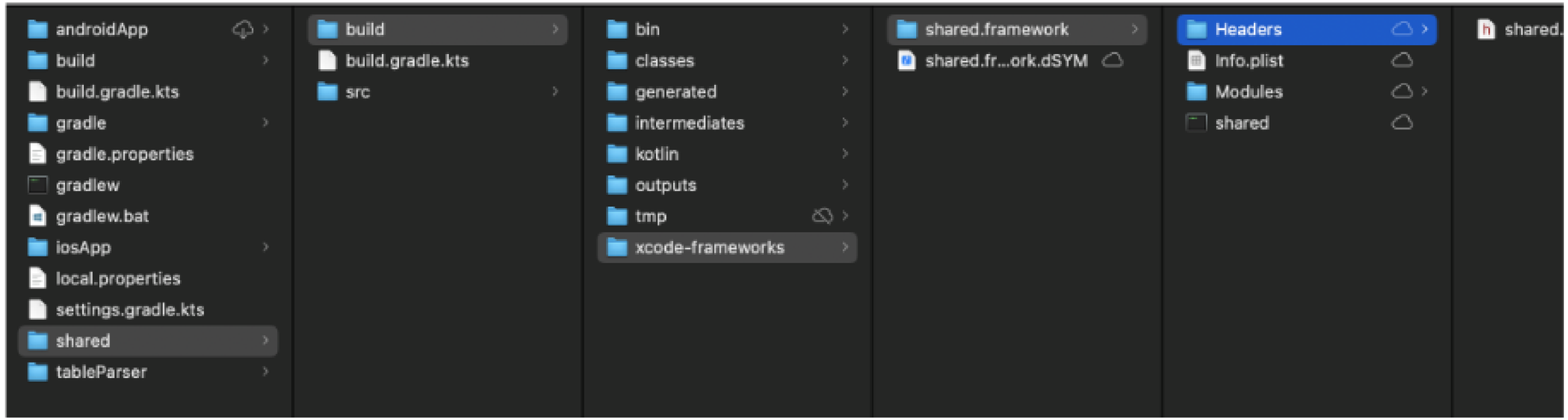
Много новых возможностей

```
allMetadataJar - Assembles a jar archive containing the metadata for all Kotlin source sets.
assemble - Assemble main outputs for all the variants.
assembleAndroidTest - Assembles all the Test applications.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
bundle - Assemble bundles for all the variants.
clean - Deletes the build directory.
cleanBuildCache - Deletes the build cache directory.
compileDebugAndroidTestSources
compileDebugSources
compileDebugUnitTestSources
compileIosMainKotlinMetadata - Compiles a klibrary from the 'iosMain' compilation for target 'common'.
compileKotlinIosArm64 - Compiles a klibrary from the 'main' compilation for target 'native'.
compileKotlinIosX64 - Compiles a klibrary from the 'main' compilation for target 'native'.
compileReleaseSources
compileReleaseUnitTestSources
compileTestKotlinIosArm64 - Compiles a klibrary from the 'test' compilation for target 'native'.
compileTestKotlinIosX64 - Compiles a klibrary from the 'test' compilation for target 'native'.
extractDebugAnnotations - Extracts Android annotations for the debug variant into the archive file
extractReleaseAnnotations - Extracts Android annotations for the release variant into the archive file
iosArm64MainKlibrary - Assembles outputs for compilation 'main' of target 'iosArm64'
iosArm64TestKlibrary - Assembles outputs for compilation 'test' of target 'iosArm64'
iosX64MainKlibrary - Assembles outputs for compilation 'main' of target 'iosX64'
iosX64TestKlibrary - Assembles outputs for compilation 'test' of target 'iosX64'
linkDebugFrameworkIosArm64 - Links a framework 'debugFramework' for a target 'iosArm64'.
linkDebugFrameworkIosX64 - Links a framework 'debugFramework' for a target 'iosX64'.
linkDebugTestIosArm64 - Links a test executable 'debugTest' for a target 'iosArm64'.
linkDebugTestIosX64 - Links a test executable 'debugTest' for a target 'iosX64'.
linkReleaseFrameworkIosArm64 - Links a framework 'releaseFramework' for a target 'iosArm64'.
linkReleaseFrameworkIosX64 - Links a framework 'releaseFramework' for a target 'iosX64'.
metadataCommonMainClasses - Assembles outputs for compilation 'commonMain' of target 'metadata'
metadataIosMainClasses - Assembles outputs for compilation 'iosMain' of target 'metadata'
metadataJar - Assembles an archive containing the main classes.
```

ManyChat

Gradle дает много новых возможностей

Если написать gradle packForXcode в консоли:



ManyChat

Публикуем код библиотеки для Андроид

```
plugins {
    …
    id("maven-publish")
}

repositories {
    mavenCentral()
    google()
    jcenter()
}

publishing {
    repositories {
        maven {
            name = artifactId
            url = uri(«URL репозитория»)
            credentials {
                username = project.findProperty("gpr.user") as String? ?: System.getenv("USERNAME")
                        ?: userName
                password = project.findProperty("gpr.key") as String? ?: System.getenv("TOKEN")
                        ?: githubToken
            }
        }
    }
}
```

ManyChat

Публикуем код библиотеки для Андроид

```
Publishing tasks
----------------
generateMetadataFileForAndroidPublication - Generates the Gradle metadata file for publication 'android'.
generateMetadataFileForIosArm64Publication - Generates the Gradle metadata file for publication 'iosArm64'.
generateMetadataFileForIosX64Publication - Generates the Gradle metadata file for publication 'iosX64'.
generateMetadataFileForKotlinMultiplatformPublication - Generates the Gradle metadata file for publication 'kotlinMultiplatform'.
generateMetadataFileForMetadataPublication - Generates the Gradle metadata file for publication 'metadata'.
generatePomFileForAndroidPublication - Generates the Maven POM file for publication 'android'.
generatePomFileForIosArm64Publication - Generates the Maven POM file for publication 'iosArm64'.
generatePomFileForIosX64Publication - Generates the Maven POM file for publication 'iosX64'.
generatePomFileForKotlinMultiplatformPublication - Generates the Maven POM file for publication 'kotlinMultiplatform'.
generatePomFileForMetadataPublication - Generates the Maven POM file for publication 'metadata'.
publish - Publishes all publications produced by this project.
publishAllPublicationsToSharedAnalyticsRepository - Publishes all Maven publications produced by this project to the SharedAnalytics repository.
publishAndroidPublicationToMavenLocal - Publishes Maven publication 'android' to the local Maven repository.
publishAndroidPublicationToSharedAnalyticsRepository - Publishes Maven publication 'android' to Maven repository 'SharedAnalytics'.
publishIosArm64PublicationToMavenLocal - Publishes Maven publication 'iosArm64' to the local Maven repository.
publishIosArm64PublicationToSharedAnalyticsRepository - Publishes Maven publication 'iosArm64' to Maven repository 'SharedAnalytics'.
publishIosX64PublicationToMavenLocal - Publishes Maven publication 'iosX64' to the local Maven repository.
publishIosX64PublicationToSharedAnalyticsRepository - Publishes Maven publication 'iosX64' to Maven repository 'SharedAnalytics'.
publishKotlinMultiplatformPublicationToMavenLocal - Publishes Maven publication 'kotlinMultiplatform' to the local Maven repository.
publishKotlinMultiplatformPublicationToSharedAnalyticsRepository - Publishes Maven publication 'kotlinMultiplatform' to Maven repository 'SharedAnalytics'.
publishMetadataPublicationToMavenLocal - Publishes Maven publication 'metadata' to the local Maven repository.
publishMetadataPublicationToSharedAnalyticsRepository - Publishes Maven publication 'metadata' to Maven repository 'SharedAnalytics'.
publishToMavenLocal - Publishes all Maven publications produced by this project to the local Maven cache.
```

ManyChat

Публикуем код библиотеки для Андроид

- publishAndroidPublicationTo{название репозитория}Repository

ManyChat

Подключаем библиотеку

```
allprojects {
    repositories {
        google()
        jcenter()
        maven {
            name = "SharedAnalytics-android"
            url = uri("URL")
            credentials {
                username = "username"
                password = "password"
            }
        }
```

ManyChat

Cocoapods плагин

- Работает только как dev pods
- Значит, что ваш проект должен содержать исходники проект агде-то рядом
- При попытке загрузить из удаленного репозитория библиотеку,
  мы столкнемся с ошибками
- Круто, когда на нем значительная часть проекта
- Но противоречит идее черной коробки и легкого подключения

ManyChat

Cocoapods plugin

```
kotlin {
    ios()

    cocoapods {
        summary = "CocoaPods test library"
        homepage = "https://github.com/JetBrains/kotlin"
        ios.deploymentTarget = "13.5"
        pod("AFNetworking") {
            version = "~> 4.0.0"
        }
        podfile = project.file("../ios-app/Podfile")
    }
}
```

4. Add the name and path of the Kotlin Pod you want to include in the Xcode project to `Podfile`.

```
use_frameworks!

platform :ios, '13.5'

target 'ios-app' do
        pod 'kotlin_library', :path => '../kotlin-library'
end
```

ManyChat

Cocoapods plugin

```
    spec.name                     = 'myModule'
    spec.version                  = '1.0'
    spec.homepage              = 'http://google.com'
    spec.source                   = { :git => 'git@github.com:myRepo.git' }
    spec.authors                 = '...'
    spec.license                  = '...'
    spec.summary                  = '...'

    spec.static_framework         = true
    spec.vendored_frameworks      = "myModule/build/cocoapods/framework/#{spec.name}.framework"
    spec.libraries                = "c++"
    spec.module_name      = "#{spec.name}_umbrella"

    spec.pod_target_xcconfig = {
        'KOTLIN_TARGET[sdk=iphonesimulator*]' => 'ios_x64',
        'KOTLIN_TARGET[sdk=iphoneos*]' => 'ios_arm'
    }

    spec.script_phases = [
        {
            :name => 'Build KN',
            :execution_position => :before_compile,
            :shell_path => '/bin/sh',
            :script => <<-SCRIPT
                set -ev
                REPO_ROOT="$PODS_TARGET_SRCROOT"
                "$REPO_ROOT/gradlew" -p "$REPO_ROOT" :myModule:syncFramework \
                    -Pkotlin.native.cocoapods.target=$KOTLIN_TARGET \
                    -Pkotlin.native.cocoapods.configuration=DEBUG \
                    -Pkotlin.native.cocoapods.cflags="$OTHER_CFLAGS" \
                    -Pkotlin.native.cocoapods.paths.headers="$HEADER_SEARCH_PATHS" \
                    -Pkotlin.native.cocoapods.paths.frameworks="$FRAMEWORK_SEARCH_PATHS"
            SCRIPT
        }
    ]
```

ManyChat

Cocoapods plugin

```
    spec.vendored_frameworks      = "myModule/build/cocoapods/framework/#{spec.name}.framework"

    spec.script_phases = [
        {
            :name => 'Build KN',
            :execution_position => :before_compile,
            :shell_path => '/bin/sh',
            :script => <<-SCRIPT
                set -ev
                REPO_ROOT="$PODS_TARGET_SRCROOT"
                "$REPO_ROOT/gradlew" -p "$REPO_ROOT" :myModule:syncFramework \
                    -Pkotlin.native.cocoapods.target=$KOTLIN_TARGET \
                    -Pkotlin.native.cocoapods.configuration=DEBUG \
                    -Pkotlin.native.cocoapods.cflags="$OTHER_CFLAGS" \
                    -Pkotlin.native.cocoapods.paths.headers="$HEADER_SEARCH_PATHS" \
                    -Pkotlin.native.cocoapods.paths.frameworks="$FRAMEWORK_SEARCH_PATHS"
            SCRIPT
        }
    ]
```

ManyChat

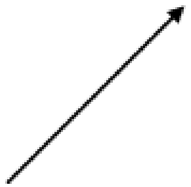
Как работает упрощенно Cocoapods

- Выкачивает зависимости
- Находит по пути vendored_frameworks библиотеку и перетаскивает ее в проект
- Выполняет дополнительные скрипты

ManyChat

Как работает упрощенно Cocoapods

- Выкачивает зависимости
- Находит по пути vendored_frameworks библиотеку и перетаскивает ее в проект
- Выполняет дополнительные скрипты



Генерится фрейморк

ManyChat

Как работает упрощенно Cocoapods

- Выкачивает зависимости
- Выполняются prepare_commands
- Находит по пути vendored_frameworks библиотеку и перетаскивает ее в проект
- Выполняет дополнительные скрипты

ManyChat

Своя реализация

```
outputFile.writeText(
    """
    |Pod::Spec.new do |spec|
    |    spec.name                     = '$specName'
    |    spec.version                  = '${settings.version}'
    |    spec.homepage                 = '${settings.homepage.orEmpty()}'
    |    spec.source                   = { :git => '${extendedSettings.githubRepo}', :tag => "v#{spec.version}" }
    |    spec.authors                  = '${settings.authors.orEmpty()}'
    |    spec.license                  = '${settings.license.orEmpty()}'
    |    spec.summary                  = '${settings.summary.orEmpty()}'
    |
    |    spec.static_framework         = true
    |    spec.vendored_frameworks      = "$frameworkDir/${frameworkNameProvider.get()}.framework"
    |    spec.libraries                = "c++"
    |    spec.preserve_paths           = "**/*.*"
    |    spec.module_name              = "#{spec.name}_umbrella"
    |
    $dependencies
    |
    |    spec.pod_target_xcconfig = {
    |        'KOTLIN_TARGET[sdk=iphonesimulator*]' => 'ios_x64',
    |        'KOTLIN_TARGET[sdk=iphoneos*]' => 'ios_arm',
    |    }
    |
    |    spec.prepare_command = <<-SCRIPT
    |       set -ev
    |       ./gradlew --no-daemon '${extendedSettings.prepeareCommandTask}' --stacktrace --info
    |    SCRIPT
    |end
""".trimMargin()
)
```

ManyChat

Теперь мы можем подключить библиотеку

```
target 'ManyChat' do
  use_frameworks!
  inhibit_all_warnings!

  pod 'SharedAnalytics', :git => 'https://github.com/.../app-analytics-shared.git', :branch => 'release/1.43.0'
```

Но если вы соберете в таком виде на устройство - вас ждет краш

- Не будет нужных символов
- Kotlin Native собирает под конкретную архитектуру
- Apple незаметно для нассобирает fat binaries
- Fat binaries — библиотеки с символами нескольких архитектур

ManyChat

К счастью мы тоже так можем

- Мы можем использовать библиотеку lipo для склеивания
- Более того, мы можем использовать плагин, который упрощает процесс

К счастью мы тоже так можем

```
import org.jetbrains.kotlin.gradle.tasks.FatFrameworkTask


project.tasks.register<FatFrameworkTask>(releaseFatFrameworkTaskName) {
    baseName = artifactId
    group = "Create Universal framework"
    destinationDir = buildDir.resolve(cocoapodsFrameworkDirName)
    from(iosArm64.binaries.getFramework("RELEASE"), iosX64.binaries.getFramework("RELEASE"))
}
```

ManyChat

К счастью мы тоже так можем

```
project.tasks.register<PodspecTask>("createPodSpec") {
    …
    extendedSettings.prepeareCommandTask = releaseFatFrameworkTaskName
}
```

ManyChat

Как теперь выглядит процесс подтягивания зависимости

- Тянется исходный код из репозитория
- Собираются бинарными под симулятор и устройства
- Склеиваются в один
- Фреймворк линкуется в проект

ManyChat

Добавим туда события

- Положим все базовые классы в commonMain
- Рядом положим скрипт и укажем путь для сгенеренных файлов (тоже commonMain)
- Напишем еще маленький скрипт, который подтягивает все зависимости и запускается одной кнопкой

Добавим туда события

- Положим все базовые классы в commonMain
- Рядом положим скрипт и укажем путь для сгенеренных файлов (тоже commonMain)
- Напишем еще маленький скрипт, который подтягивает все зависимости и запускается одной кнопкой
- Но пока что это все запускается на локальной машине : (

ManyChat

План

- Выбираем формат хранилища событий ✅
- Выкачиваем данные из хранилища событий ✅
- Превращаем в код ✅
- Заворачиваем в библиотеку(-ки), доступные на обеих платформах ✅
- Подключаем привычным для платформы способом

ManyChat

Нужен CI

ManyChat

Скорей всего он у вас уже есть

- В случае GitHub у вас есть GithubActions
- Github Packages как Maven Repository
- Actions запускаются по триггерам

ManyChat

Как выглядит наш алгоритм

- Как триггер выбрали создание ветки (например release/1.10.1)
- Запускается CI
- Подтягиваются все зависимости (JDK, Python)
- Генерация событий
- Генерация подфайла
- Коммитновых файлов в ветку
- Публикация в Maven репозиторий

ManyChat

Сколько занимает

- Коммит новых файлов — примерно минуту
- Публикация в Maven репозиторий — примерно 3 минуты

Ссылка на проект

ManyChat

Выводы

- Генерация кода заняла всего неделю одного человека
- KMM классно справляется с ролью черной коробки
- Продукт доволен: прекратились ошибки/забытые параметры в событиях
- Разработчики довольны: аналитика стала удобной в использовании
- Работает стабильно уже 10 месяцев

ManyChat

Всякая полезна всячина

- Работа с таблицей
- Пример таблицы
- Gradle
- Плагин для создания KMM проекта
- Github actions

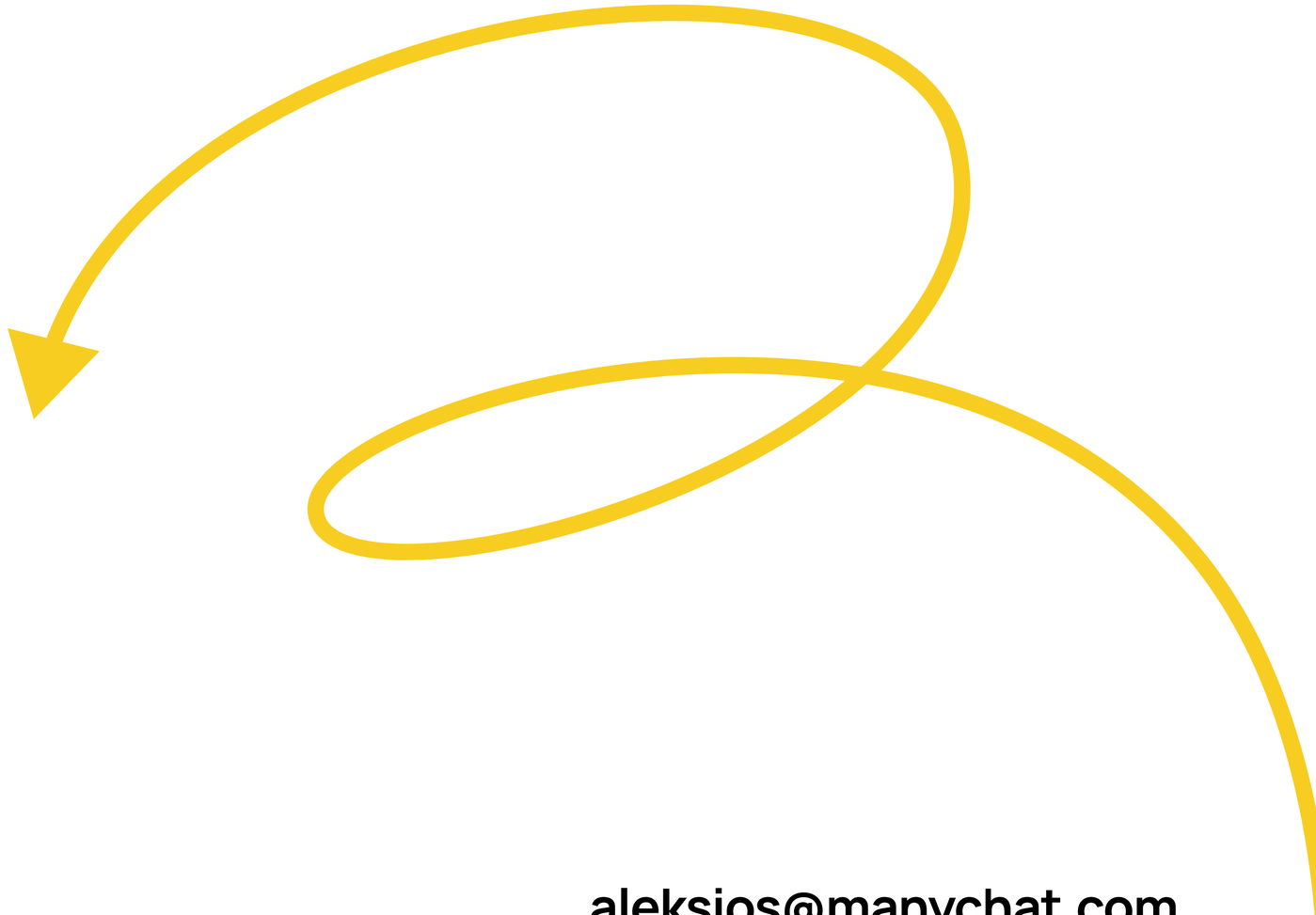
ManyChat

Спасибо!

Вопросы?

Александр Лавриненко

aleksios@manychat.com



ManyChat