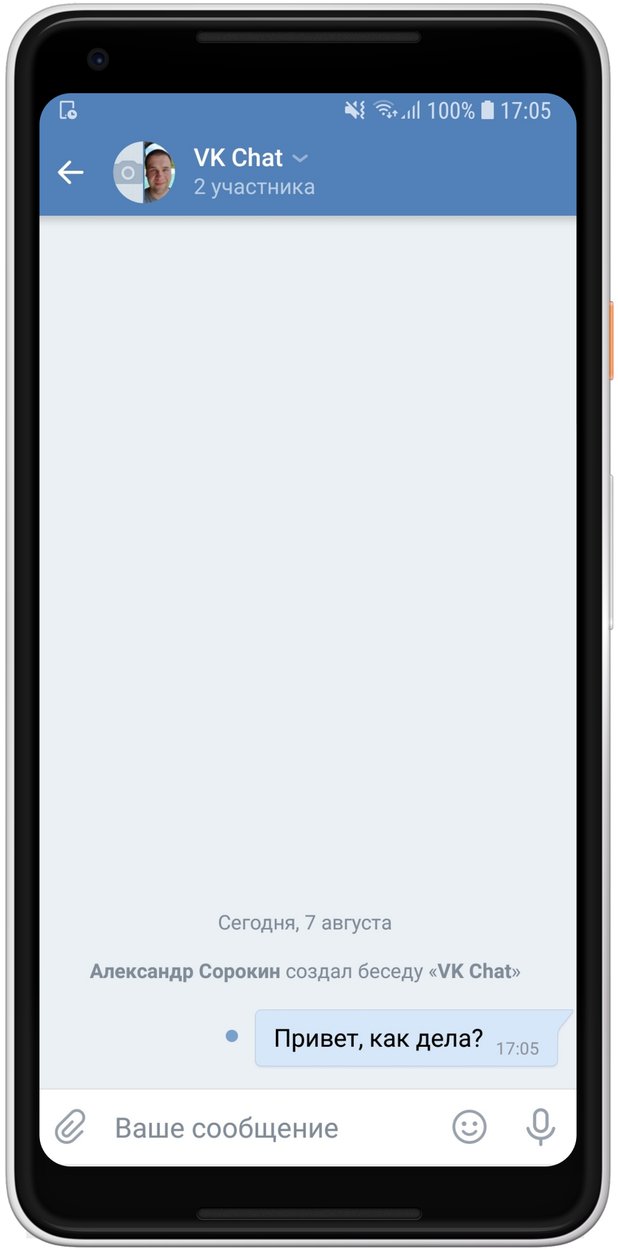
РЕНДЕРИНГ ЭКРАНА СООБЩЕНИЙ

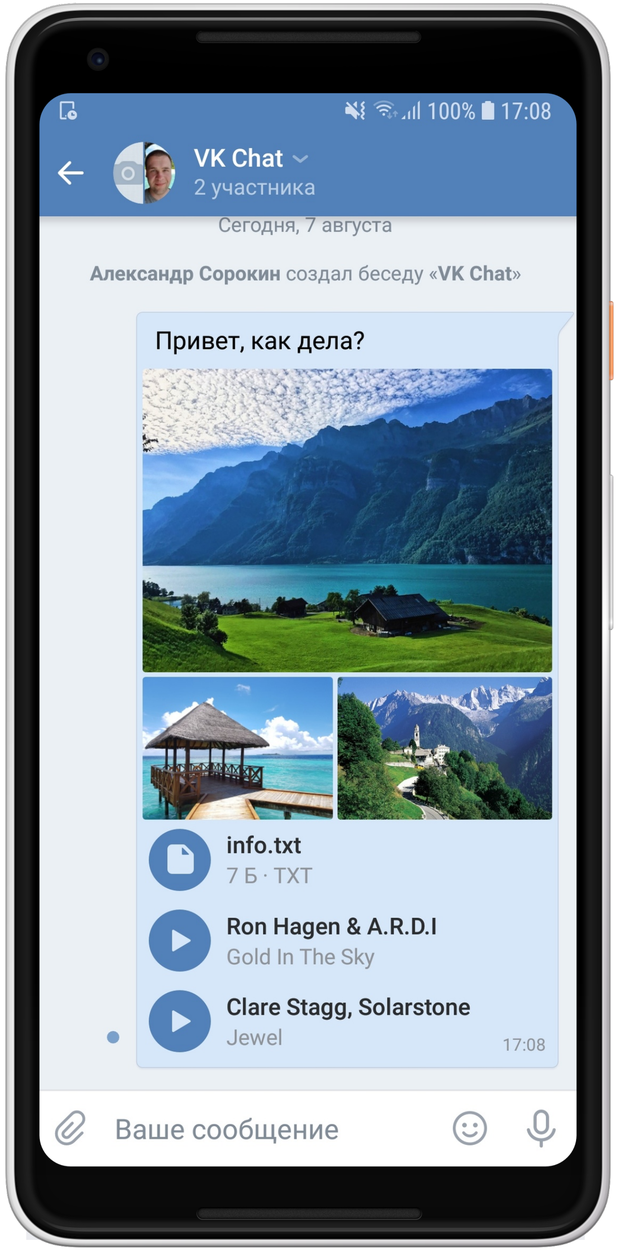
СОРОКИН АЛЕКСАНДР, ВКОНТАКТЕ

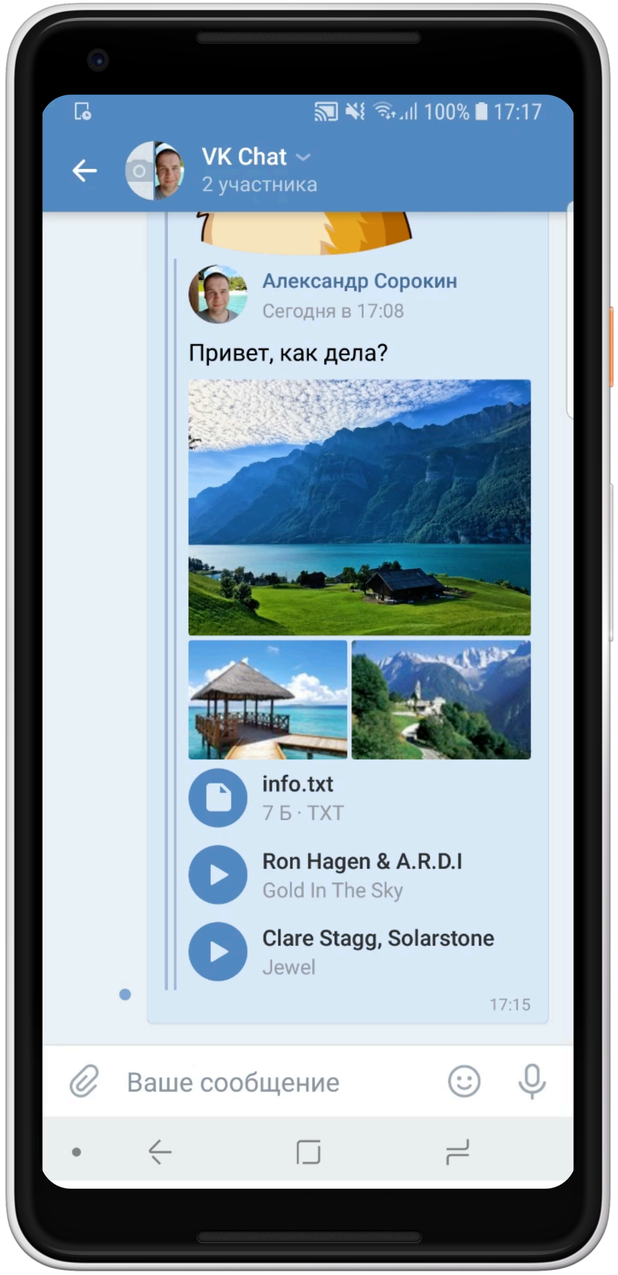
Principal Android Developer

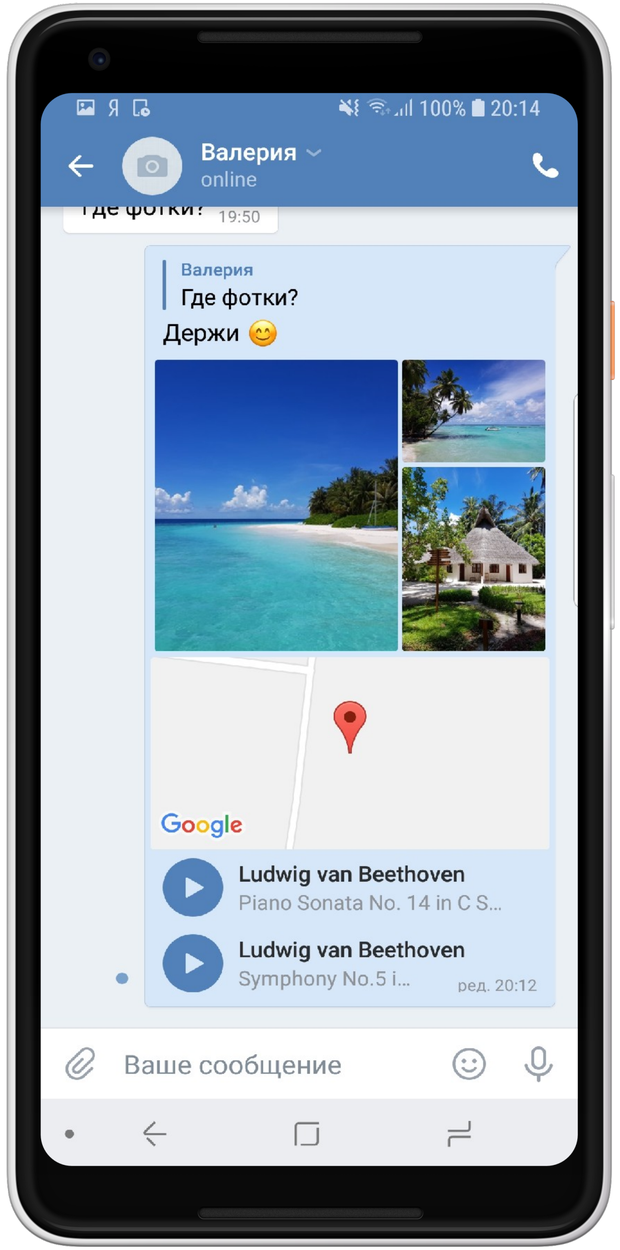
http://vk.com/alexoro

uas.sorokin@gmail.com


100% 17:05
VK Chat
2 участника
Сегодня, 7 августа
Александр Сорокин создал беседу «VK Chat»
Привет, как дела? 17:05
Ваше сообщение

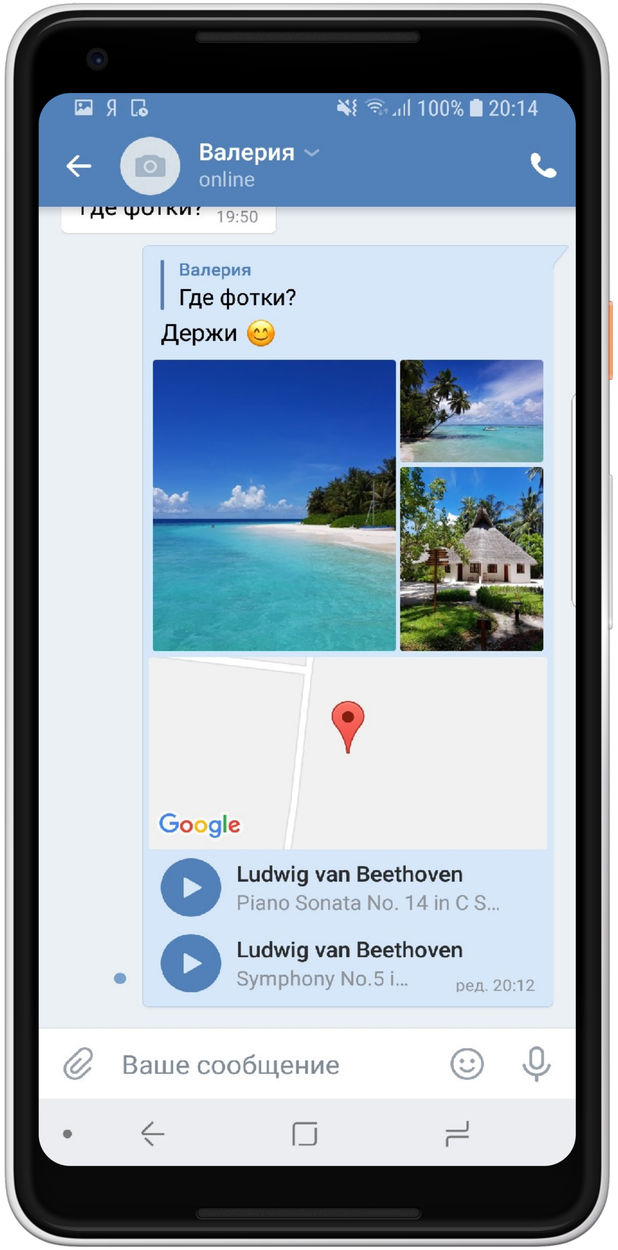

100% 17:08
VK Chat
2 участника
Сегодня, 7 августа
Александр Сорокин создал беседу «VK Chat»
Привет, как дела?
info.txt
7 Б · TXT
Ron Hagen & A.R.D.I
Gold In The Sky
Clare Stagg, Solarstone
Jewel
17:08
Ваше сообщение


100% 17:17
VK Chat
2 участника
Александр Сорокин
Сегодня в 17:08
Привет, как дела?
info.txt
7 Б · TXT
Ron Hagen & A.R.D.I
Gold In The Sky
Clare Stagg, Solarstone
Jewel
17:15
Ваше сообщение



СТРУКТУРА СООБЩЕНИЯ

- Текст
- Клавиатура ботов
- Список аттачей (~30 типов)
- Ответ на какое-либо сообщение
- Дерево пересланных сообщений
  - До 100 на одном уровне
  - До 40 в глубину
  - Не более ~300



ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ

- Даты / Индикаторы подгрузки
- Сервисные сообщения
- Текст (emoji, link, email, hashtags, ...)
- Клавиатура ботов
- ~40 способов отображения аттачей
- Дерево пересланных сообщений

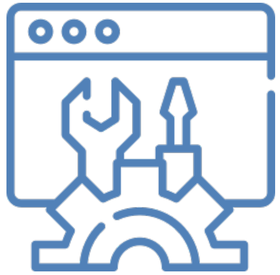
КАК ЭТО СДЕЛАТЬ?

КАК ЭТО СДЕЛАТЬ?
И ЧТОБЫ НЕ ЛАГАЛО

О ЧЕМ ПОЙДЕТ РЕЧЬ

Стандартные решения

Нестандартные решения



Альтернативы

О ЧЕМ ПОЙДЕТ РЕЧЬ

Стандартные решения

Нестандартные решения

Альтернативы

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView +
LinearLayoutManager

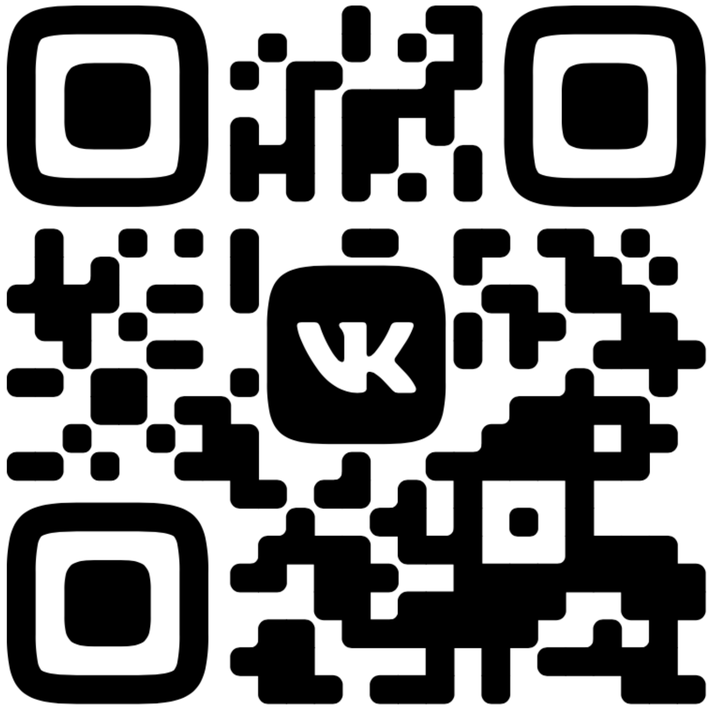
СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView +
LinearLayoutManager

RecyclerView: надстройки

RecyclerView: надстройки

- setHasFixedSize(<boolean>)
- setNestedScrollingEnabled(<boolean>)
- setItemViewCacheSize(<cache_size>)



RecyclerView:
как устроен кэш

https://vk.cc/9W1ztA

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView + LinearLayoutManager

RecyclerView: надстройки

Оптимизация onCreateViewHolder / onBindViewHolder

OnCreate / OnBind ViewHolder

- Легкая верстка (FrameLayout / CustomViewGroup).
- Тяжелые операции (парсинг ссылок / emoji) делаются асинхронно на этапе загрузки сообщений.
- StringBuilder для форматирования имени / даты / etc.
- И прочее..

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView + LinearLayoutManager

RecyclerView: надстройки

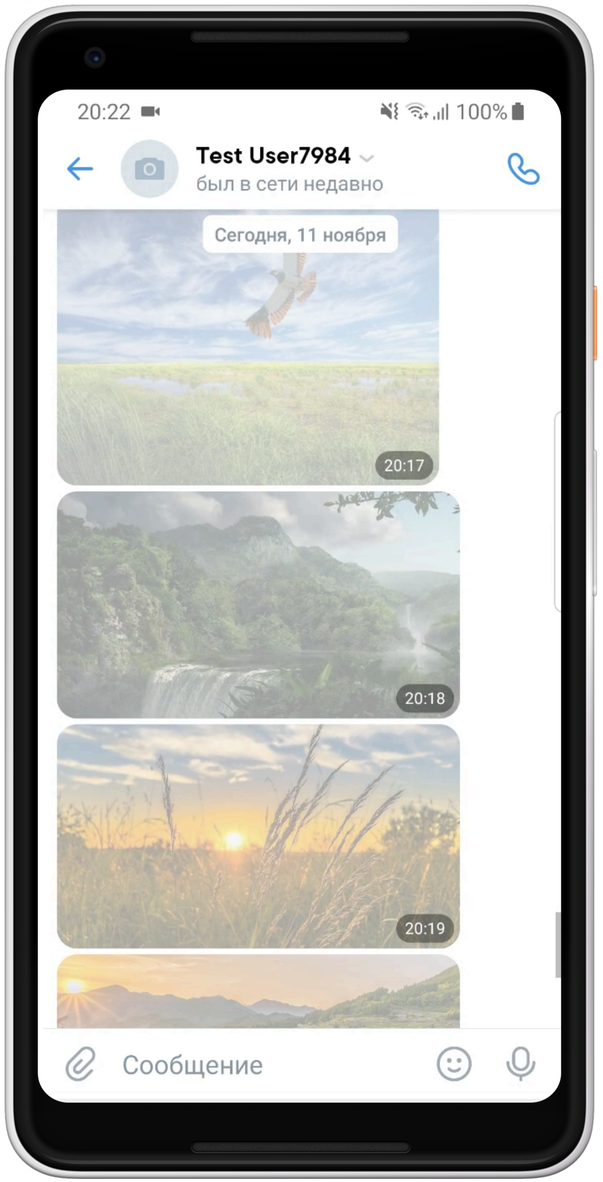
Оптимизация onCreateViewHolder / onBindViewHolder

Отслеживание Adapter.onFailedToRecycleView()

Adapter.onFailedToRecyclerView

Возникает, когда:

- View.hasTransientState == true (обычно, при анимациях).
- View в этот момент готова уйти в RecycledViewPool.



Adapter.onFailedToRecyclerView

Решение:

```
class Adapter : RecyclerView.Adapter {
    fun onViewDetachedFromWindow(holder: ViewHolder) {
        cancelAnimation()
    }



}
```

Adapter.onFailedToRecyclerView

Решение:

```
class Adapter : RecyclerView.Adapter {
    fun onViewDetachedFromWindow(holder: ViewHolder) {
        cancelAnimation()
    }
    fun onFailedToRecycleView(holder: ViewHolder) {
        return true // если лень фиксить анимацию
    }
}
```

СТАНДАРТНЫЕ РЕШЕНИЯ

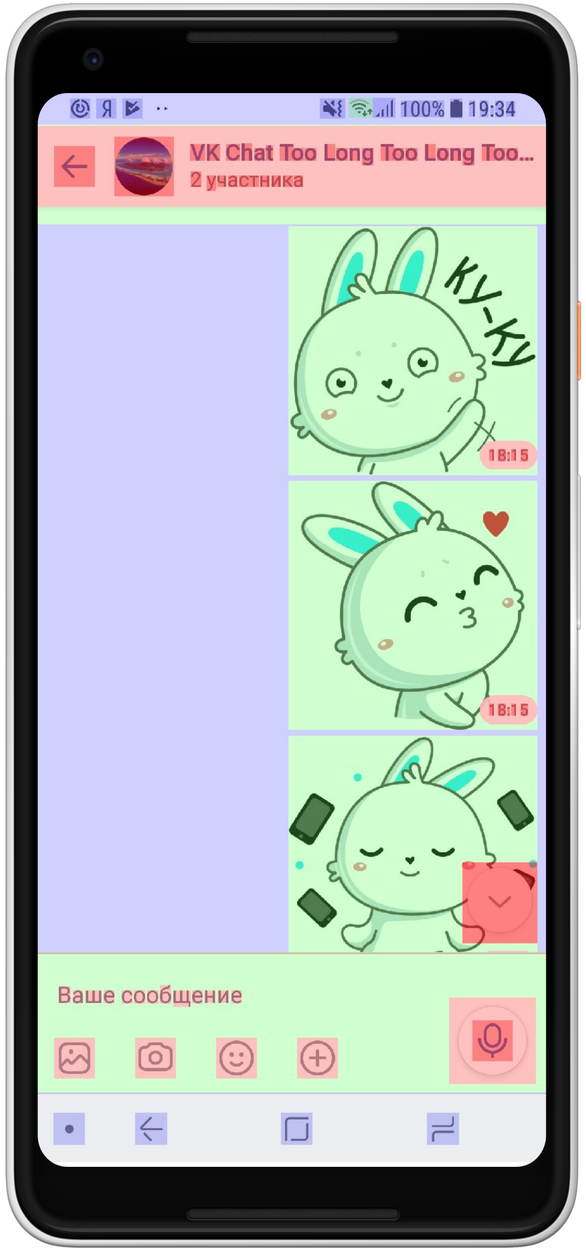
RecyclerView + LinearLayoutManager

RecyclerView: надстройки

Оптимизация onCreateViewHolder / onBindViewHolder

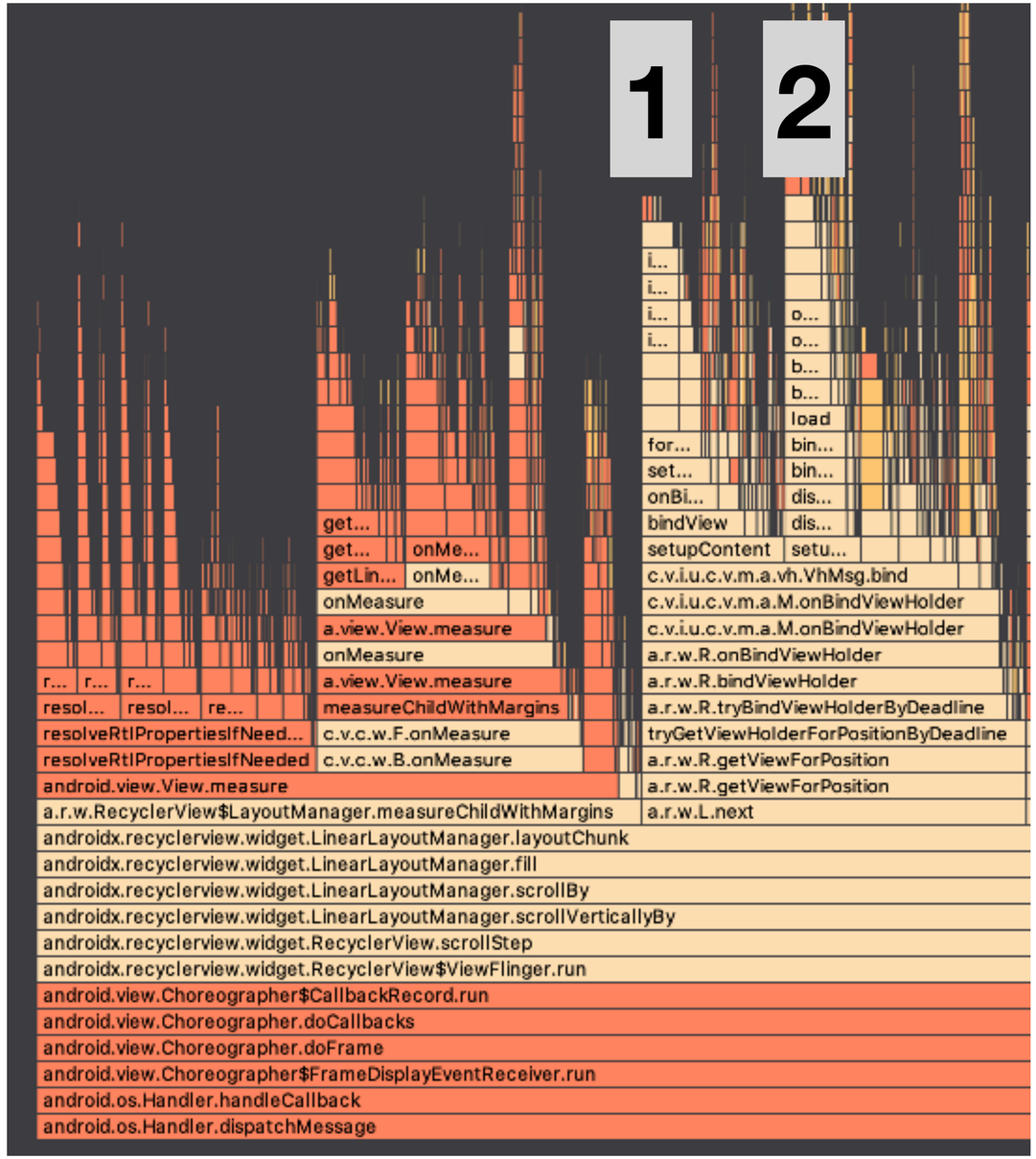
Отслеживание Adapter.onFailedToRecycleView()

Отслеживание overdraw / profiler



Overdraw

Один лишний background



Profiler

Скролл в списке сообщений:

1. Вызов регулярных выражений при форматировании данных.
2. Binding изображения в Fresco.

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView +
LinearLayoutManager

RecyclerView: надстройки

Оптимизация
onCreateViewHolder /
onBindViewHolder

Отслеживание
Adapter.onFailedToRecycleView()

Отслеживание overdraw / profiler

DiffUtil*

DiffUtil

Как и когда используем:

ListAdapter /
AsyncListDiffer

DiffUtil

Как и когда используем:

ListAdapter / AsyncListDiffer

Только для структурных изменений (insert / update / delete).

DiffUtil

Как и когда используем:

ListAdapter / AsyncListDiffer

Только для структурных изменений (insert / update / delete).

Не подходит для быстроизменяемых данных (upload progress / audio progress).

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView + LinearLayoutManager

RecyclerView: надстройки

Оптимизация onCreateViewHolder / onBindViewHolder

Отслеживание Adapter.onFailedToRecycleView()

Отслеживание overdraw / profiler

DiffUtil*

Анимации

Анимации

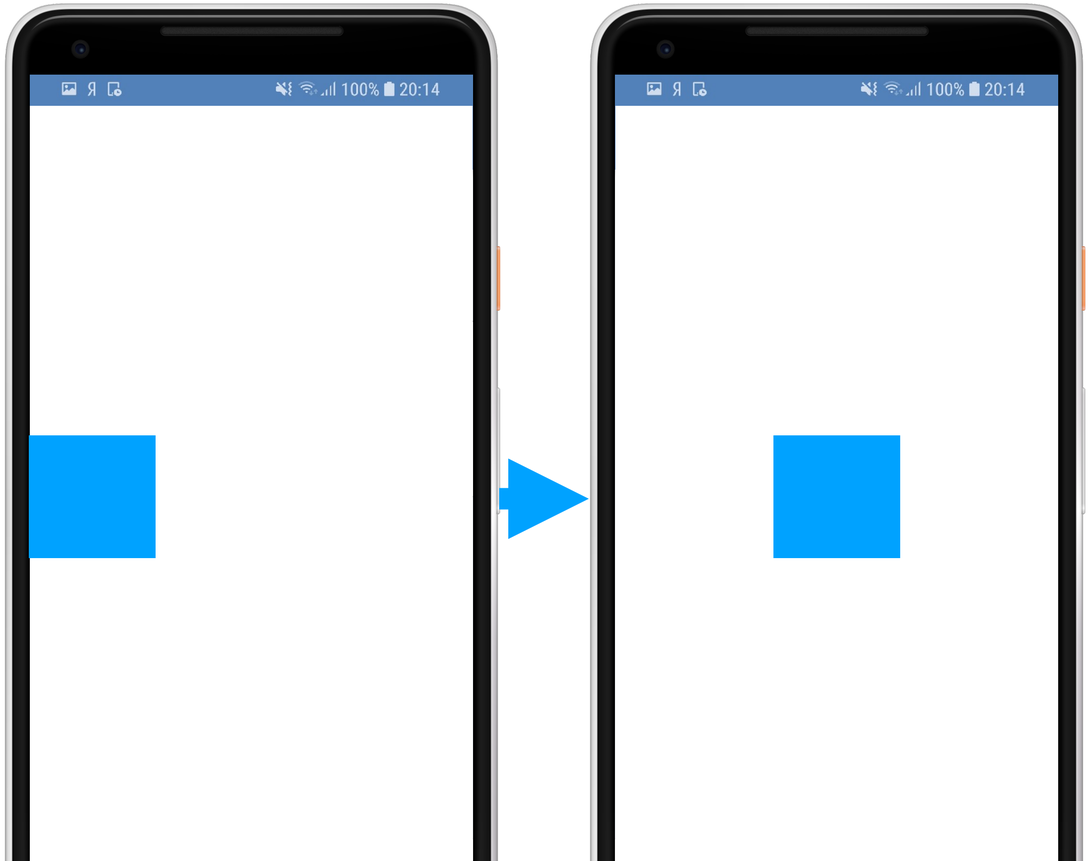
- Animation

Animation

```
view.startAnimation(TranslateAnimation(fromX = 0, toX = 300))
```

Animation

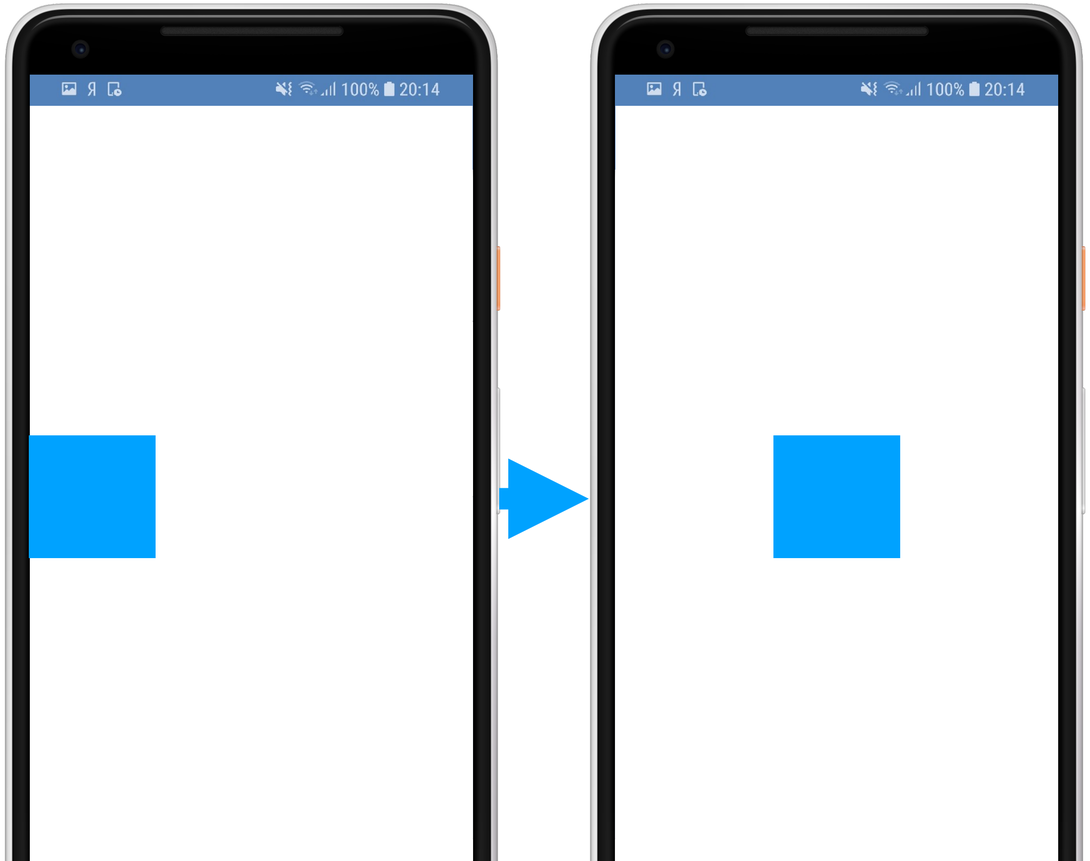
```
view.startAnimation(TranslateAnimation(fromX = 0, toX = 300))
```



Animation

```
view.startAnimation(TranslateAnimation(fromX = 0, toX = 300))
```

getTranslationX() = 0



getTranslationX() = 0

Анимации

- Animation
- Animator

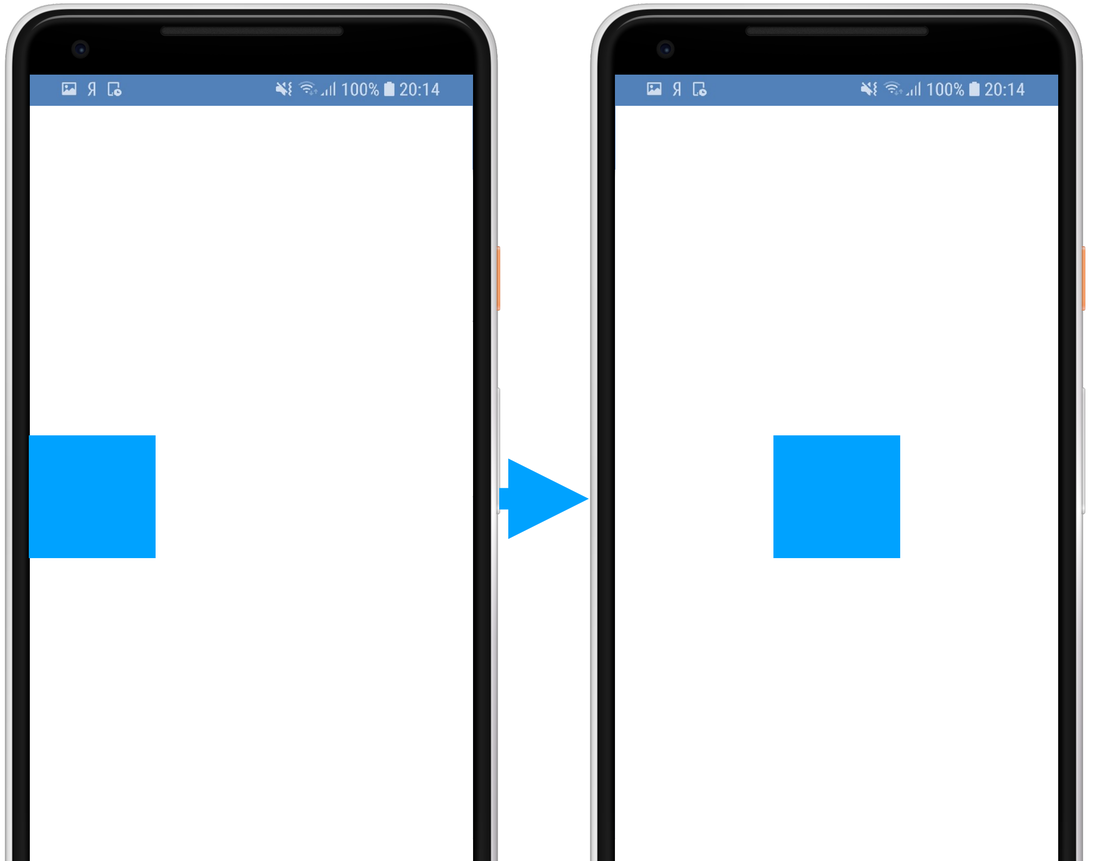
Animator

```
ObjectAnimator.ofFloat(view, “translationX”, 300f).start()
```

Animator

ObjectAnimator.ofFloat(view, “translationX”, 300f).start()

getTranslationX() = 0



getTranslationX() = 300

Анимации

- Animation
- Animator
- ViewPropertyAnimator

ViewPropertyAnimator

```
val animX = ObjectAnimator.ofFloat(view, “translationX”, 100f)
val animY = ObjectAnimator.ofFloat(view, “translationY”, 200f)
AnimatorSet().apply {
    playTogether(animX, animY)
    start()
}
```

ViewPropertyAnimator

```
val animX = ObjectAnimator.ofFloat(view, “translationX”, 100f)
val animY = ObjectAnimator.ofFloat(view, “translationY”, 200f)
AnimatorSet().apply {
    playTogether(animX, animY)
    start()
}
```

```
view.animate().translationX(100f).translationY(200f)
```

ViewPropertyAnimator

- Проще читать и поддерживать

ViewPropertyAnimator

- Проще читать и поддерживать
- Batch операций анимации

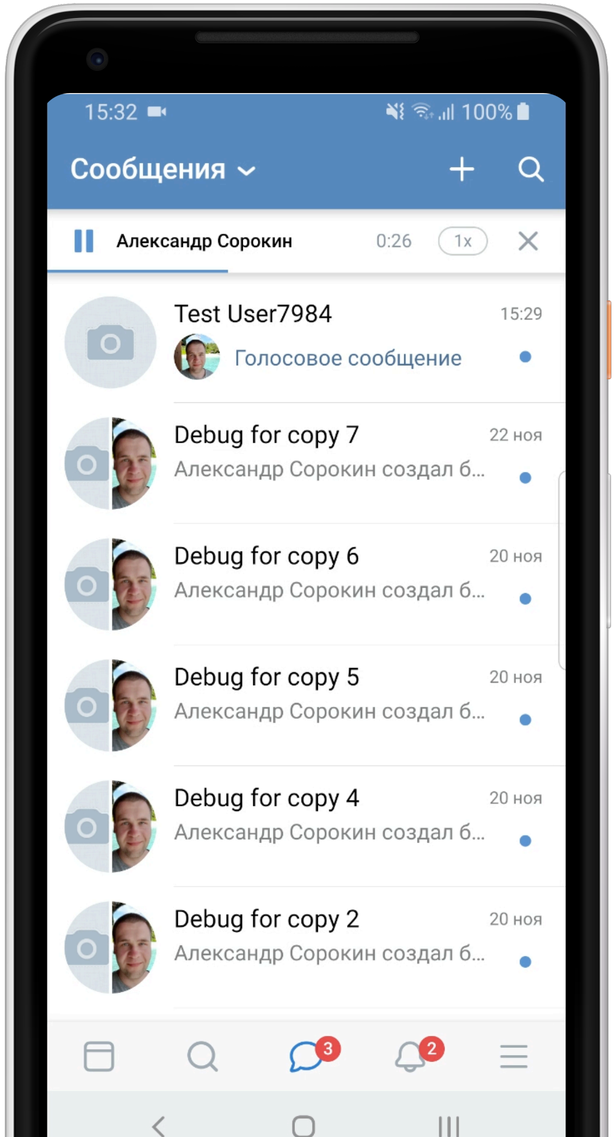
ViewPropertyAnimator

- Проще читать и поддерживать
- Batch операций анимации
- Performance (влияют только на RenderNode)

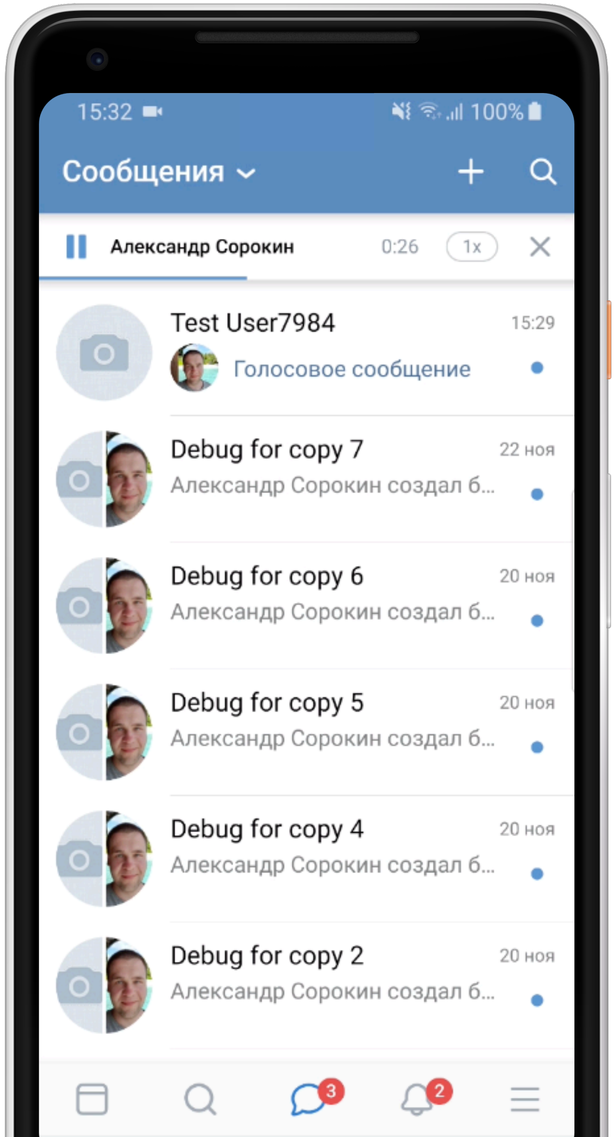
Анимации

- Animation
- Animator
- ViewPropertyAnimator
- TransitionManager

TransitionManager

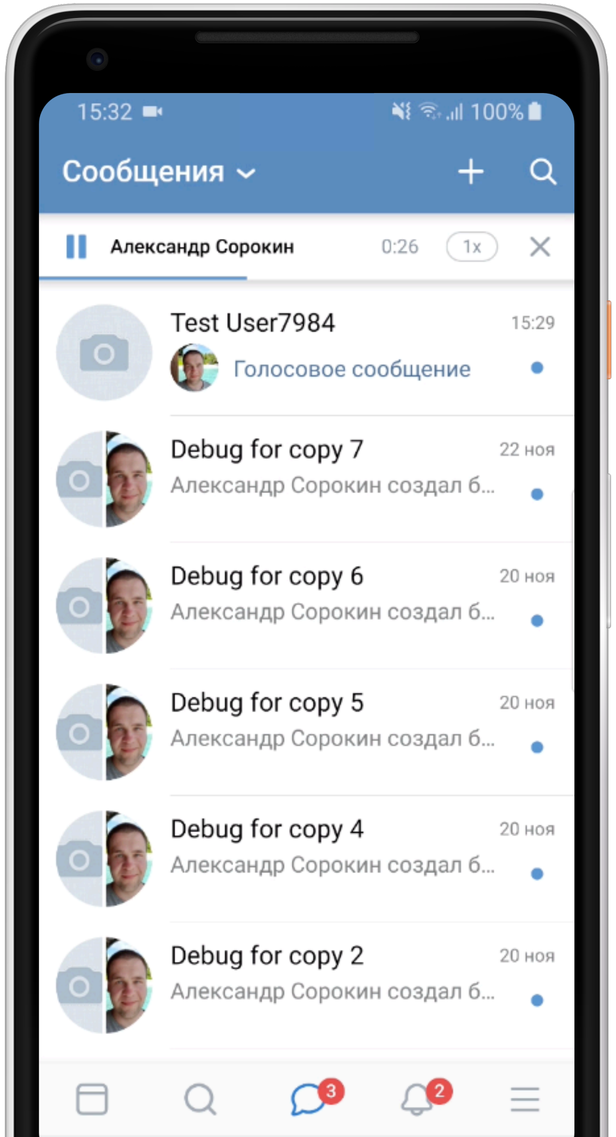


TransitionManager



LinearLayoutManager

TransitionManager

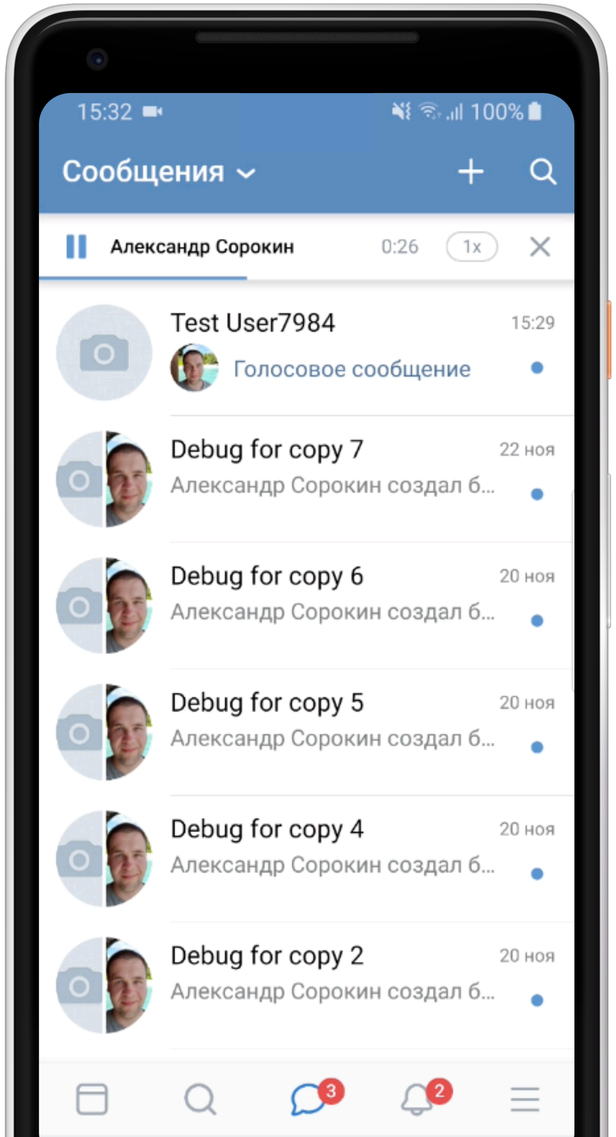



```
val buttonsTranslationStart = if (animateToShow) 0f else BUTTON_TRANSLATION.toFloat()
val buttonsTranslationFinish = if (animateToShow) BUTTON_TRANSLATION.toFloat() else 0f
val buttonsTranslationAnimator = ValueAnimator.ofFloat(buttonsTranslationStart, buttonsTranslationFinish).also {
    it.addUpdateListener { animator ->
        val translationX = animator.animatedValue as Float
        setPosterButtonAnimationTranslation(translationX)
    }
}

if (animateToShow) {
    posterButton.let {
        it.scaleX = 0.0f
        it.scaleY = 0.0f
        it.isVisible = true
    }
}
val posterButtonScaleStart = if (animateToShow) 0.0f else 1.0f
val posterButtonScaleFinish = if (animateToShow) 1.0f else 0.0f
val posterButtonScaleAnimator = ValueAnimator.ofFloat(posterButtonScaleStart, posterButtonScaleFinish).also {
    it.addUpdateListener { animator ->
        val scale = animator.animatedValue as Float
        posterButton.scaleX = scale
        posterButton.scaleY = scale
    }
    it.addListener(object : Animator.AnimatorListener {
        override fun onAnimationStart(animation: Animator?) {}
        override fun onAnimationCancel(animation: Animator?) {}
        override fun onAnimationRepeat(animation: Animator?) {}
        override fun onAnimationEnd(animation: Animator?) {
            val scale = if (animateToShow) 1.0f else 0.0f
            posterButton.let { v ->
                v.scaleX = scale
                v.scaleY = scale
                if (!animateToShow) {
                    v.isVisible = false
                }
            }
        }
    })
}

posterButtonAnimatorSet = AnimatorSet().also { animatorSet ->
    animatorSet.play(buttonsTranslationAnimator)
    posterButtonScaleAnimator?.let { animatorSet.play(it) }
    animatorSet.duration = POSTER_BUTTON_ANIMATION_DURATION_MS
    animatorSet.start()
}
```

TransitionManager




```
TransitionManager.beginDelayedTransition(
        viewGroup = <LinearLayoutManager>,
        transition = AutoTransition())
playerView.visibility = View.GONE
```

СТАНДАРТНЫЕ РЕШЕНИЯ

RecyclerView + LinearLayoutManager

RecyclerView: надстройки

Оптимизация onCreateViewHolder / onBindViewHolder

Отслеживание Adapter.onFailedToRecycleView()

Отслеживание overdraw / profiler

DiffUtil*

Анимации

О ЧЕМ ПОЙДЕТ РЕЧЬ

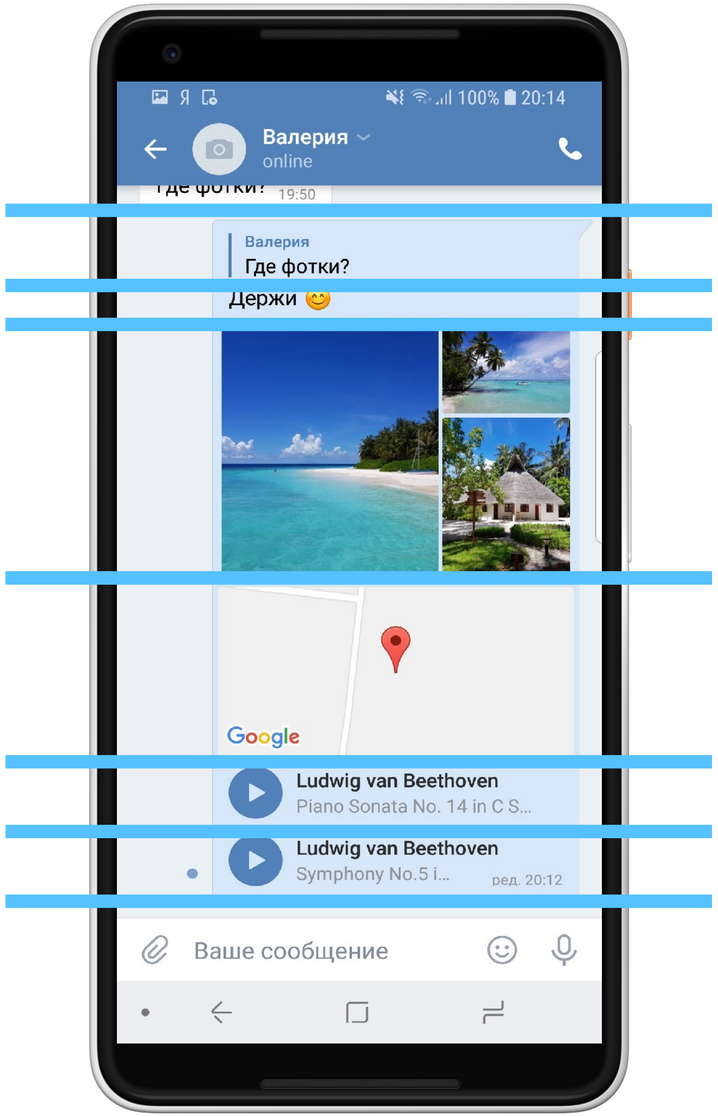
Стандартные решения

Нестандартные решения

Альтернативы

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder



Разбиение сообщения на ViewHolder-ы

Как организованы ViewHolder

Как организованы ViewHolder

Простые ViewHolder:

Дата, смена названия чата
и т.д. (TextView)

Смена аватарки чата
(TextView + ImageView)

Как организованы ViewHolder

Простые ViewHolder:

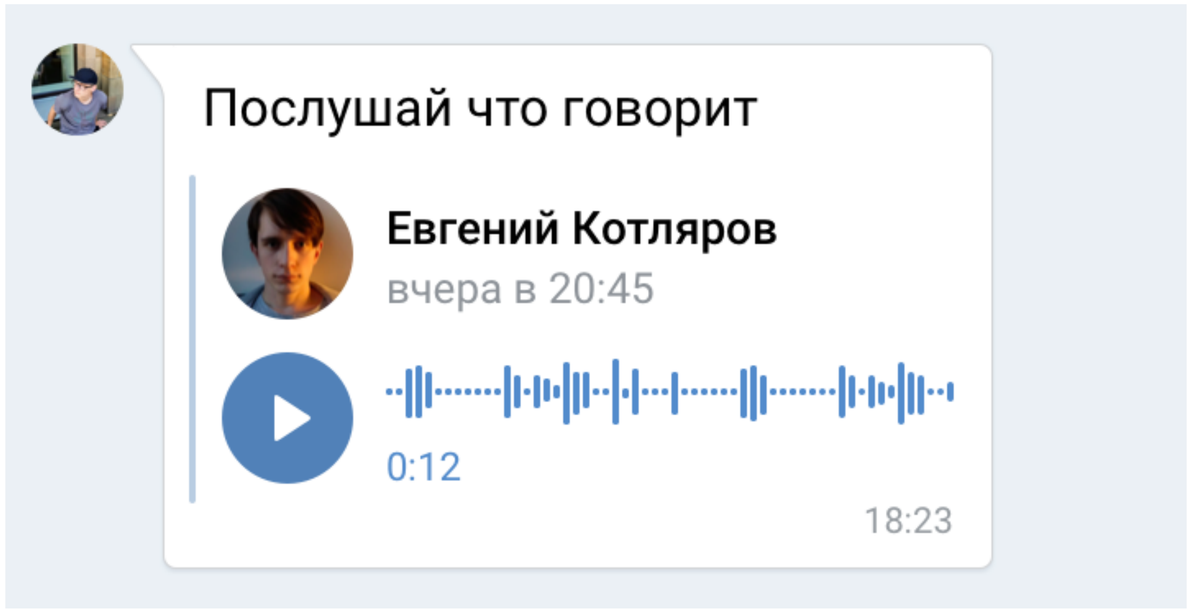
Дата, смена названия чата
и т.д. (TextView)

Смена аватарки чата
(TextView + ImageView)

ViewHolder-ы сообщений:

BaseViewHolder (abstract)
- TextViewHolder
- PhotoViewHolder
- AudioViewHolder
- ReplyViewHolder
- ...

Зоны ответственности



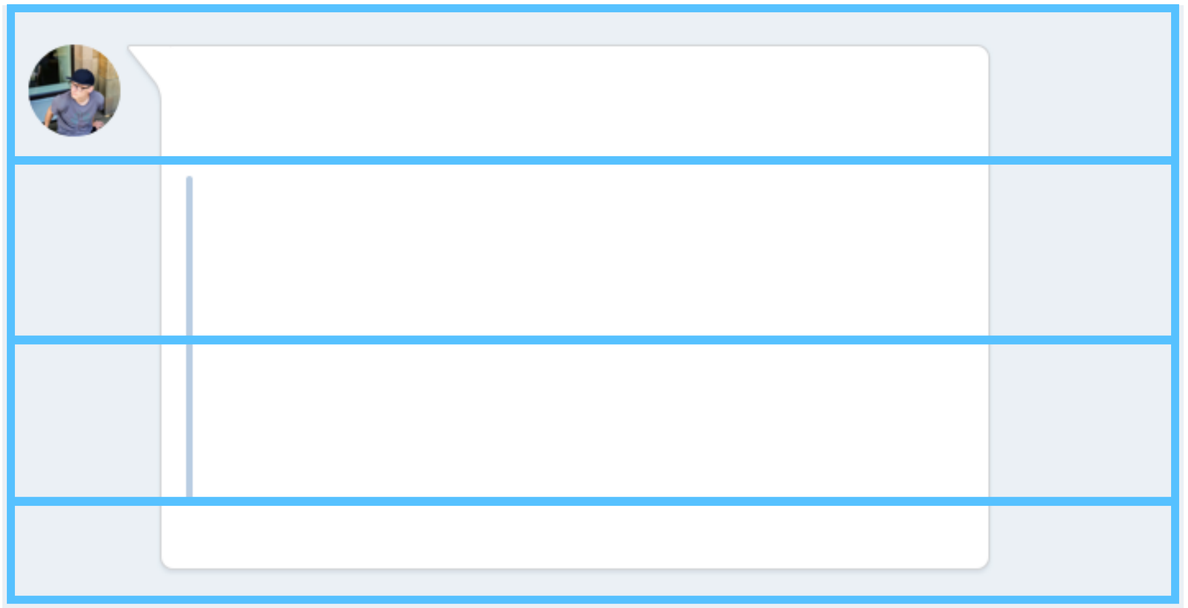
Зоны ответственности

BaseViewHolder

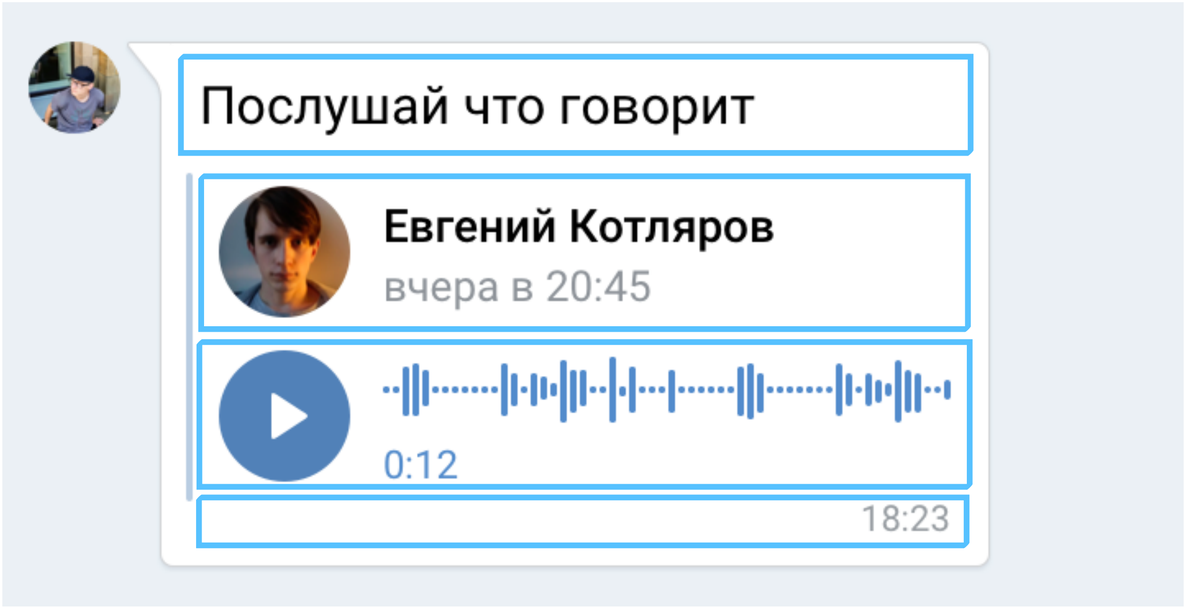
BaseViewHolder

BaseViewHolder

BaseViewHolder



Зоны ответственности



TextViewHolder

FwdSenderViewHolder

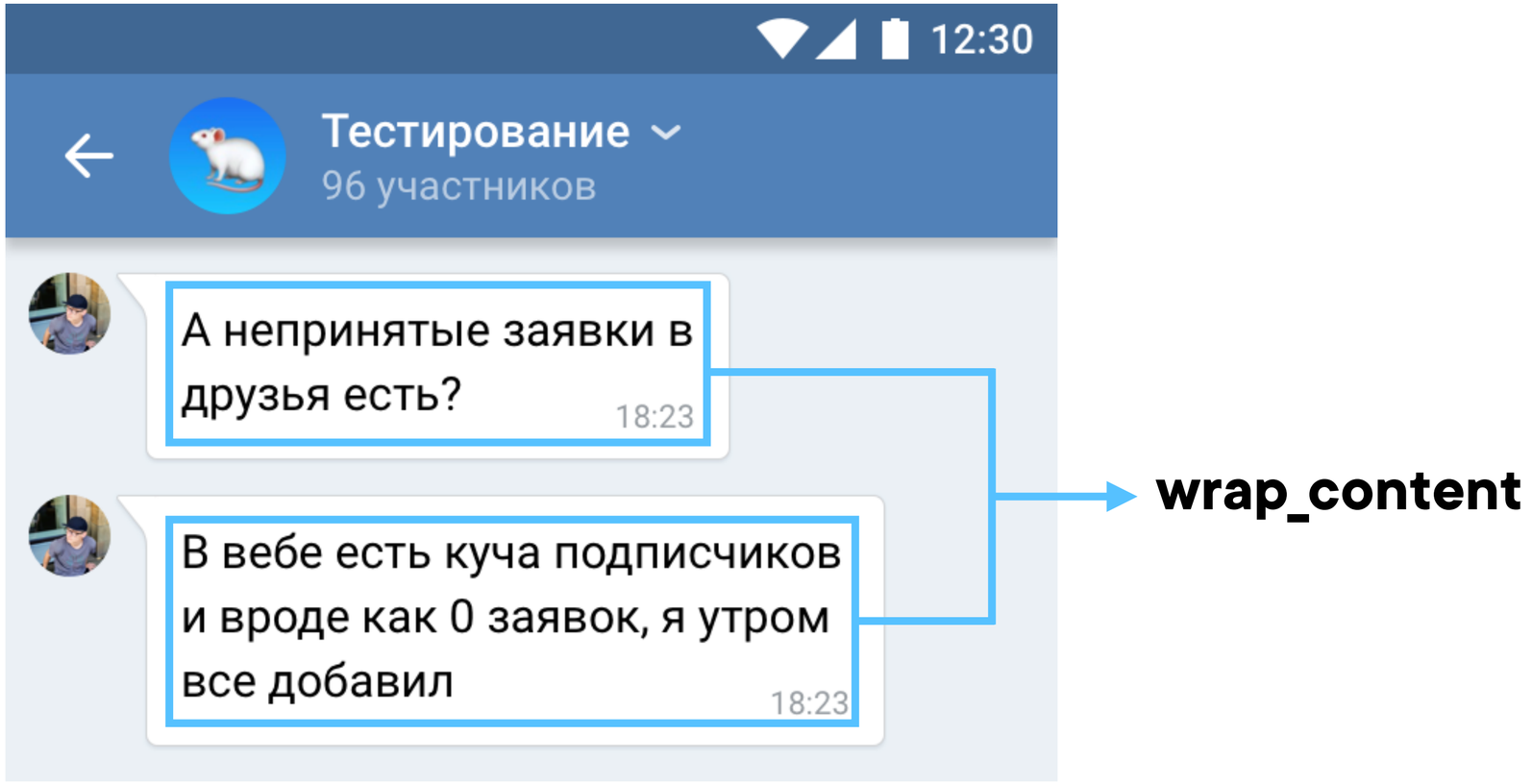
AudioMsgViewHolder

TimeViewHolder

Настройка ширины

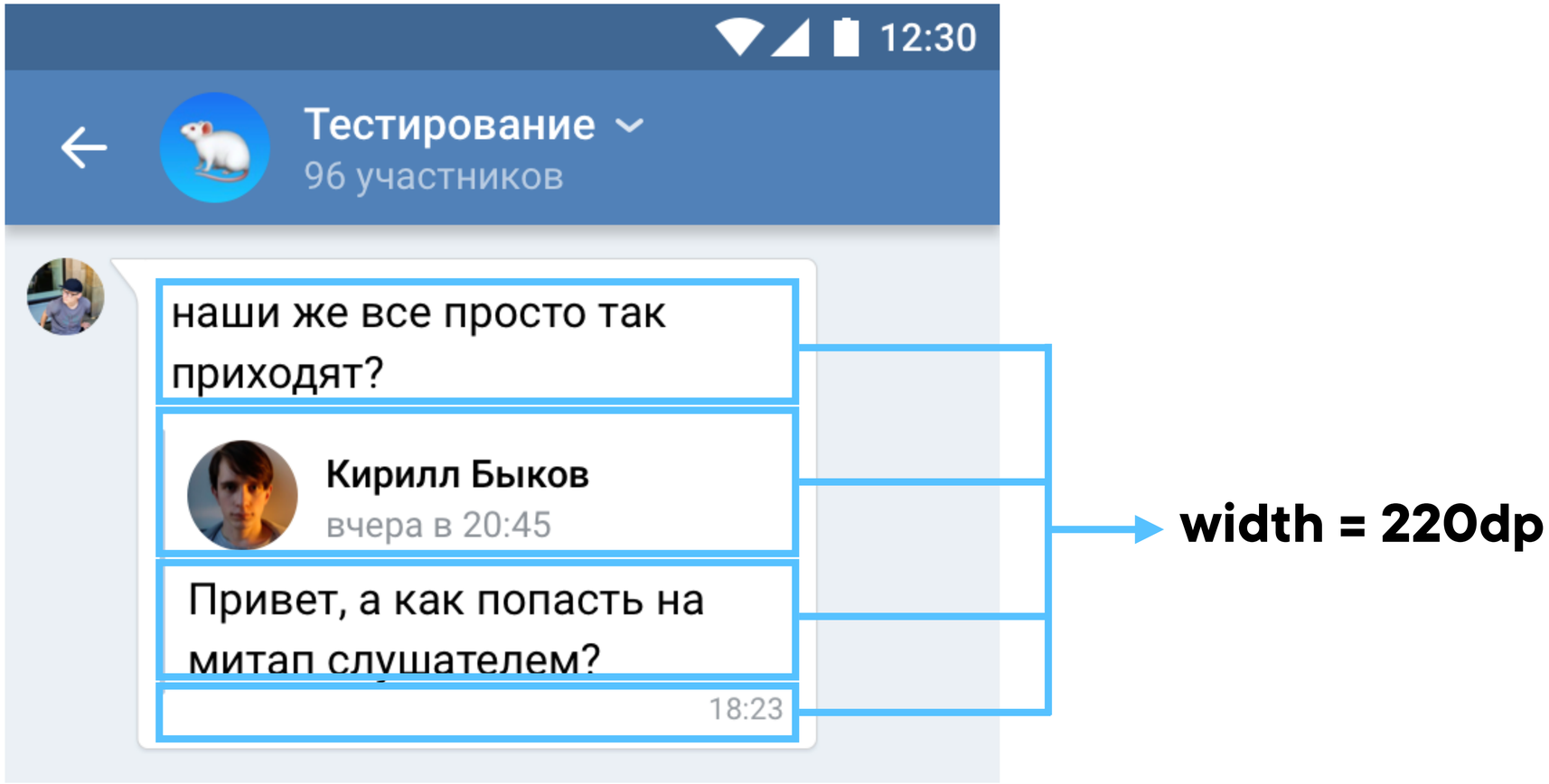
Настройка ширины

Простое сообщение (1 тип содержимого)



Настройка ширины

Сложное сообщение (несколько типов содержимого)



Настройка ширины

Сложное сообщение (reply + sticker)

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Когда разбиваем на ViewHolder?

На этапе загрузки сообщений (примерный pipeline):

01

Запускаем
асинхронную загрузку

1. Грузим List<Msg>
   из сети / кэша
2. Преобразуем List<Msg>
   -> List<MsgListItem>

Когда разбиваем на ViewHolder?

На этапе загрузки сообщений (примерный pipeline):

01 ---------- 02

Запускаем
асинхронную загрузку

1. Грузим List<Msg>
   из сети / кэша
2. Преобразуем List<Msg>
   -> List<MsgListItem>

Доставляем
результат на @UiThread

Когда разбиваем на ViewHolder?

На этапе загрузки сообщений (примерный pipeline):

01 ---------- 02 ---------- 03

Запускаем асинхронную загрузку

1. Грузим List<Msg> из сети / кэша
2. Преобразуем List<Msg> -> List<MsgListItem>

Доставляем результат на @UiThread

Устанавливаем в RecyclerView

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

Глобальный RecycledViewPool сообщений

- [x] В рамках Application живет RecycledViewPool как синглтон

Глобальный RecycledViewPool сообщений

- [x] В рамках Application живет RecycledViewPool как синглтон
- [x] Переиспользуется на экране сообщений

Глобальный RecycledViewPool сообщений

- ☑ В рамках Application живет RecycledViewPool как синглтон
- ☑ Переиспользуется на экране сообщений
- ☑ Устанавливается как RecyclerView.setRecycledViewPool(pool)

Глобальный RecycledViewPool сообщений

ВАЖНО:

Глобальный RecycledViewPool сообщений

ВАЖНО:

LinearLayoutManager.recycleChildrenOnDetach = true

Глобальный RecycledViewPool сообщений

ВАЖНО:

LinearLayoutManager.recycleChildrenOnDetach = true

Настройка размера RecycledViewPool:

Глобальный RecycledViewPool сообщений

ВАЖНО:

LinearLayoutManager.recycleChildrenOnDetach = true

Настройка размера RecycledViewPool:

- RecycledViewPool.setMaxRecycledViews(viewType, size)

Глобальный RecycledViewPool сообщений

ВАЖНО:

LinearLayoutManager.recycleChildrenOnDetach = true

Настройка размера RecycledViewPool:

- RecycledViewPool.setMaxRecycledViews(viewType, size)
- NoLimitRecycledViewPool (https://vk.cc/9DV7Xr)

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

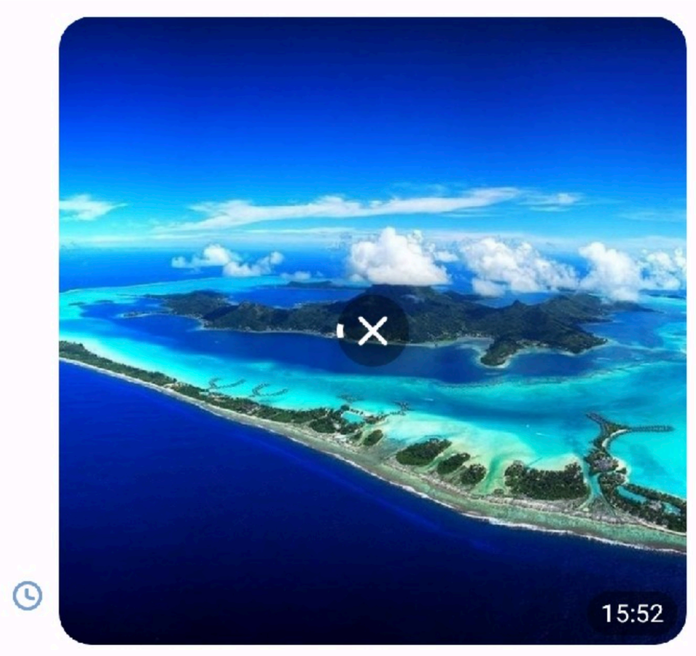
Разбиение сообщения
на несколько ViewHolder

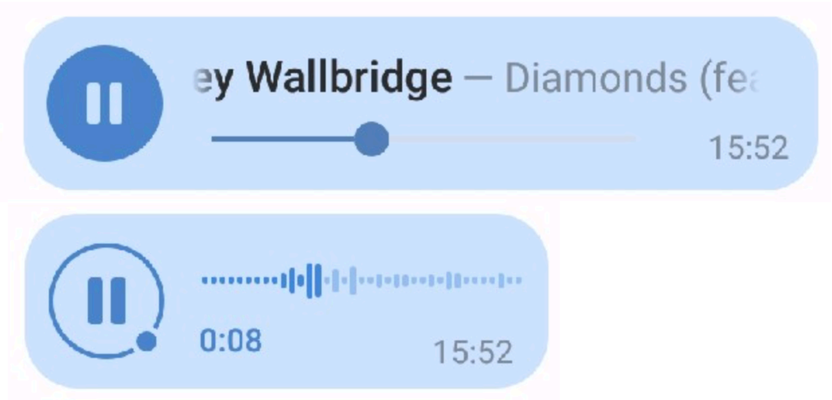
Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

DiffUtil - не всегда полезен

DiffUtil — не всегда полезен





Слишком частые события, спам вызовов DiffUtil

Модифицируем ViewHolder напрямую

```
abstract class BaseViewHolder : ViewHolder {
    ...
    fun updateUploadProgress(attachId: Int, progress: Float)
    ...
}
```

Модифицируем ViewHolder напрямую

```
fun onUploadProgress(attachId: Int, progress: Float) {
    forEachActiveViewHolder {
        it.updateUploadProgress(attachId, progress)
    }
}
```

Поиск активных ViewHolder

```
val firstVisiblePosition = <...>
val lastVisiblePosition = <...>
for (i in firstVisiblePosition.. lastVisiblePosition) {
    val viewHolder = recyclerView.findViewHolderForAdapterPosition(i)
    viewHolder.updateUploadProgress(...)
}
```

Поиск активных ViewHolder

```
val firstVisiblePosition = <...>
val lastVisiblePosition = <...>
for (i in firstVisiblePosition.. lastVisiblePosition) {
    val viewHolder = recyclerView.findViewHolderForAdapterPosition(i)
    viewHolder.updateUploadProgress(...)
}
```

Не включает ViewHolder из промежуточного кэша:

RecyclerView.setItemViewCacheSize

Поиск активных ViewHolder

```
class Adapter : RecyclerView.Adapter {

    val activeViewHolders = WeakSet<ViewHolder>()

    fun onBindViewHolder(holder: ViewHolder, position: Int) {
        activeViewHolders.add(holder)
    }
    fun onViewRecycled(holder: ViewHolder) {
        activeViewHolders.remove(holder)
    }

}
```

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

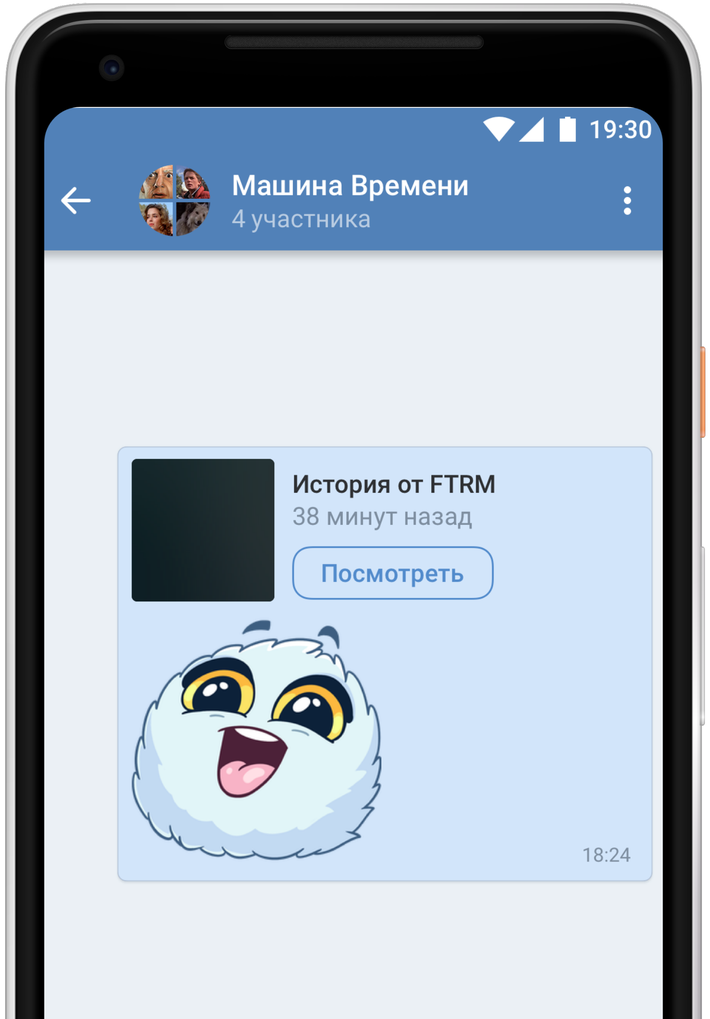
DiffUtil - не всегда полезен

Наложение ViewHolder

Наложение ViewHolder: Истории

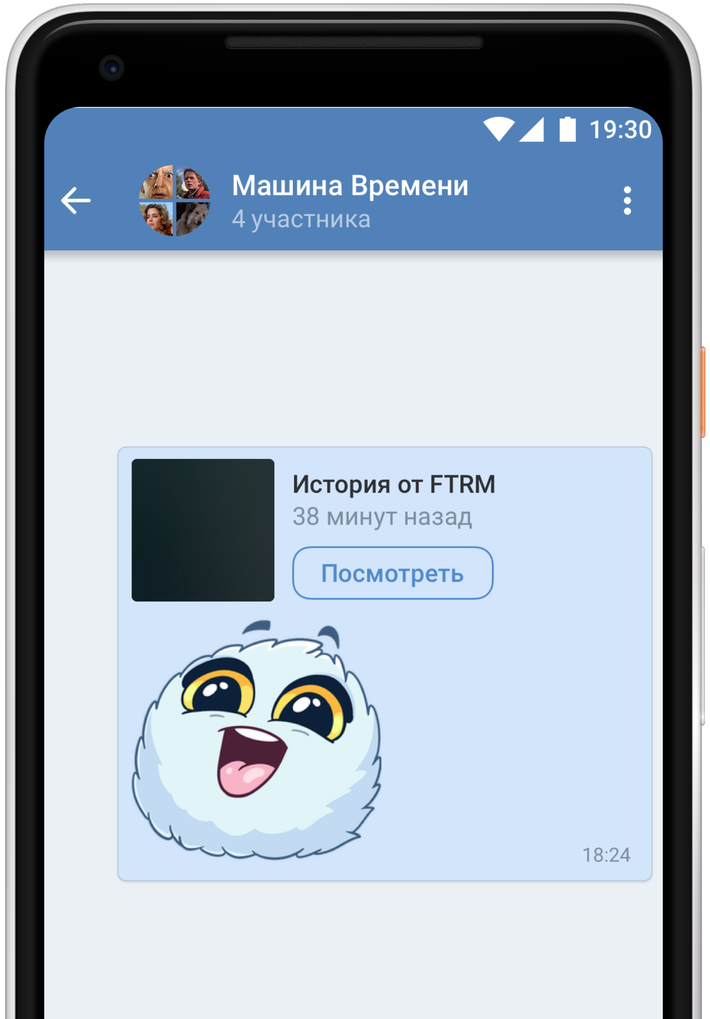
Наложение ViewHolder: Истории

До

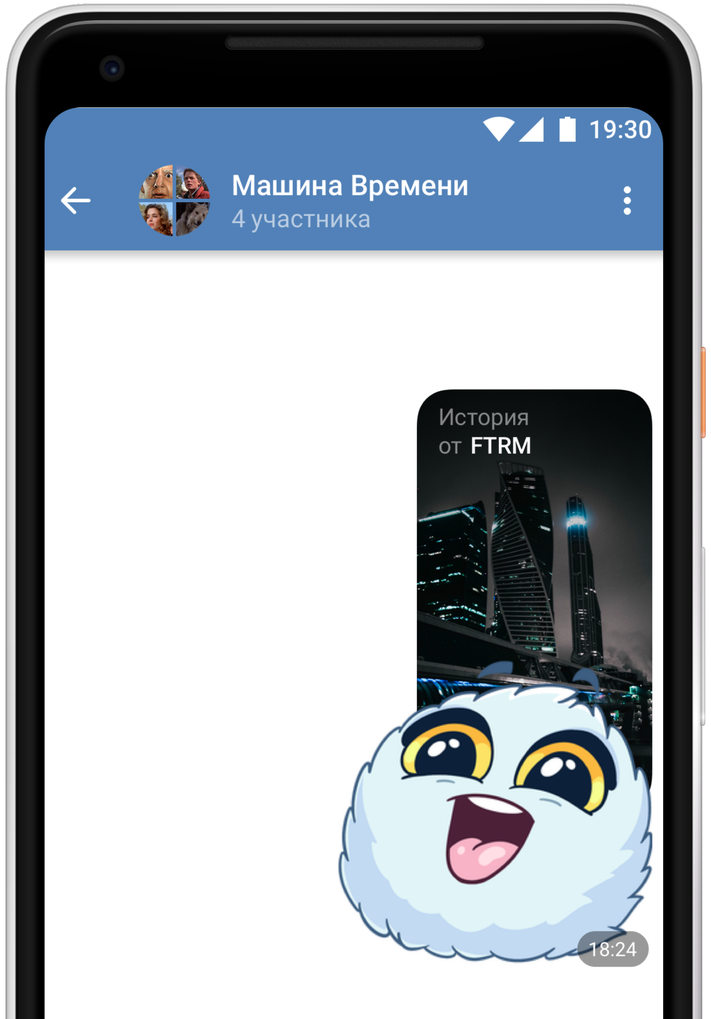


Наложение ViewHolder: Истории

До

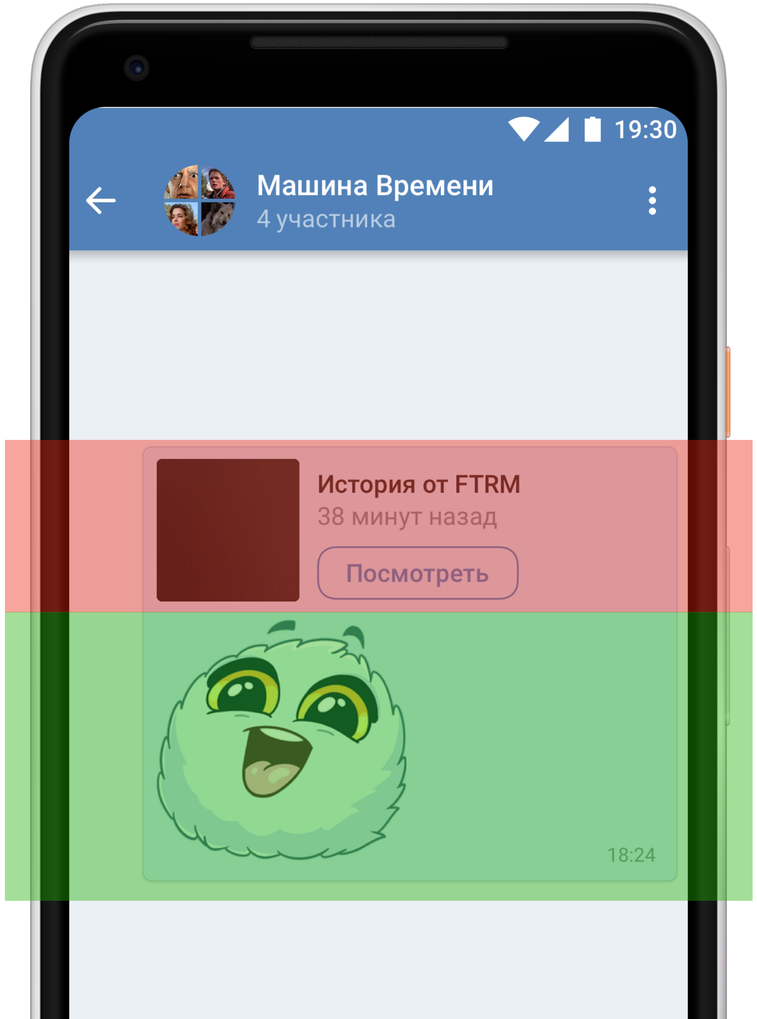


После

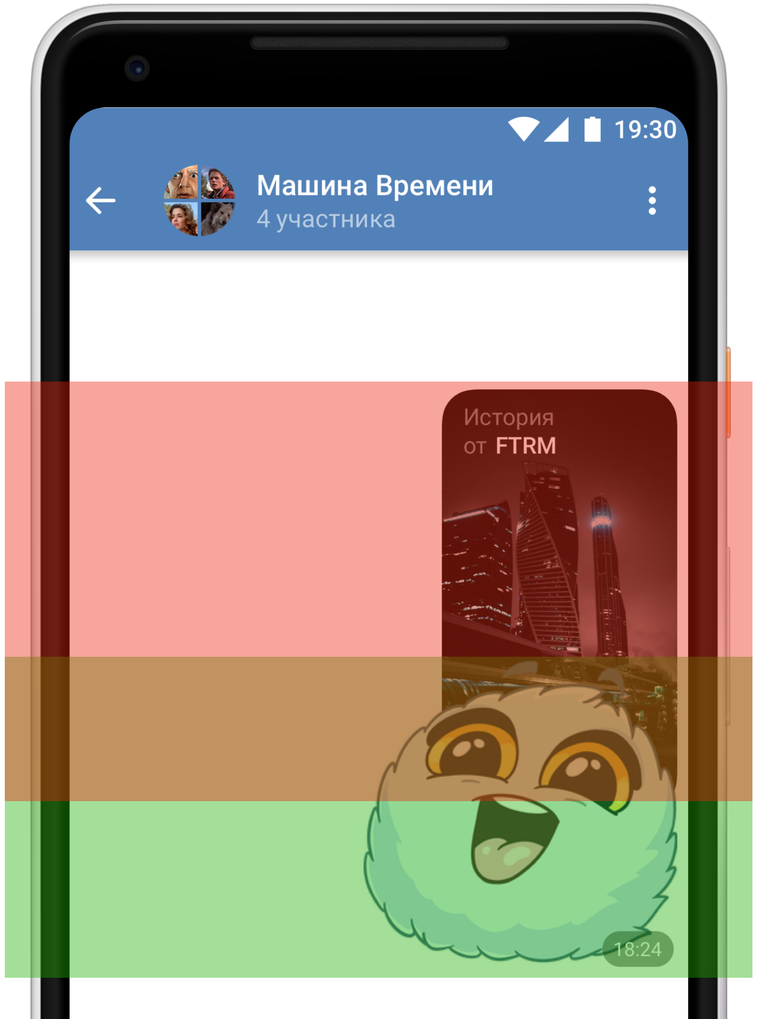


Наложение ViewHolder: Истории

До



После



Наложение ViewHolder: Истории

```
class OffsetItemDecoration : RecyclerView.ItemDecoration() {

    override fun getItemOffsets(offset: Rect, …) {
        offset.top = -100dp
    }

}
```

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

DiffUtil - не всегда полезен

Наложение ViewHolder

IdleHandler*

Упрощенный принцип работы @UiThread

```
val tasks = Queue<Runnable>()

while (true) {
    if (tasks.isEmpty()) {
        awaitNewTasks()
        continue;
    } else {
        tasks.forEach { it.run() }
    }
}
```

Упрощенный принцип работы @UiThread

Handler

```
val tasks = Queue<Runnable>()

while (true) {
    if (tasks.isEmpty()) {
        awaitNewTasks()
        continue;
    } else {
        tasks.forEach { it.run() }
    }
}
```

Looper

IdleHandler

Looper.myQueue().addIdleHandler()

IdleHandler

```
@AnyThread
class IdleHandler {

    private val handler = Handler(Looper.getMainLooper())

    fun post(task: Runnable) {
        handler.post {
            Looper.myQueue().addIdleHandler {
                task.run()
                return@addIdleHandler false
            }
        }
    }

}
```

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

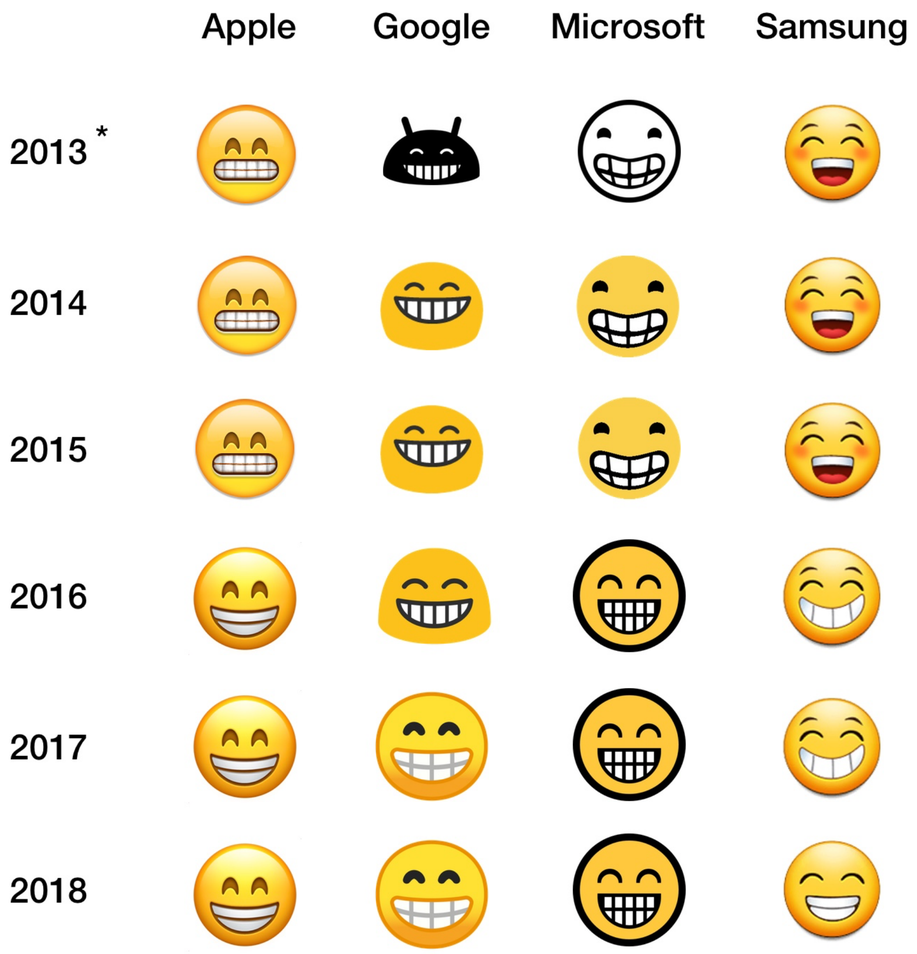
DiffUtil - не всегда полезен

Наложение ViewHolder

IdleHandler*

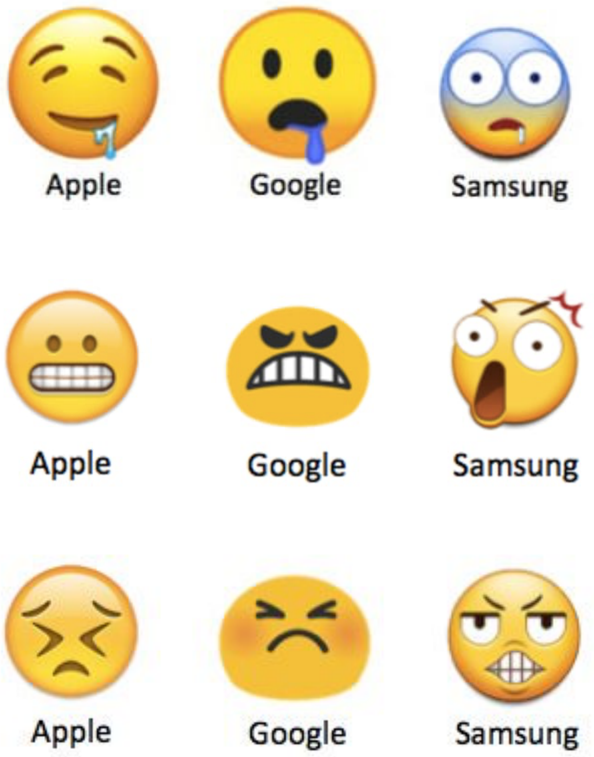
Emoji

Emoji: зачем?



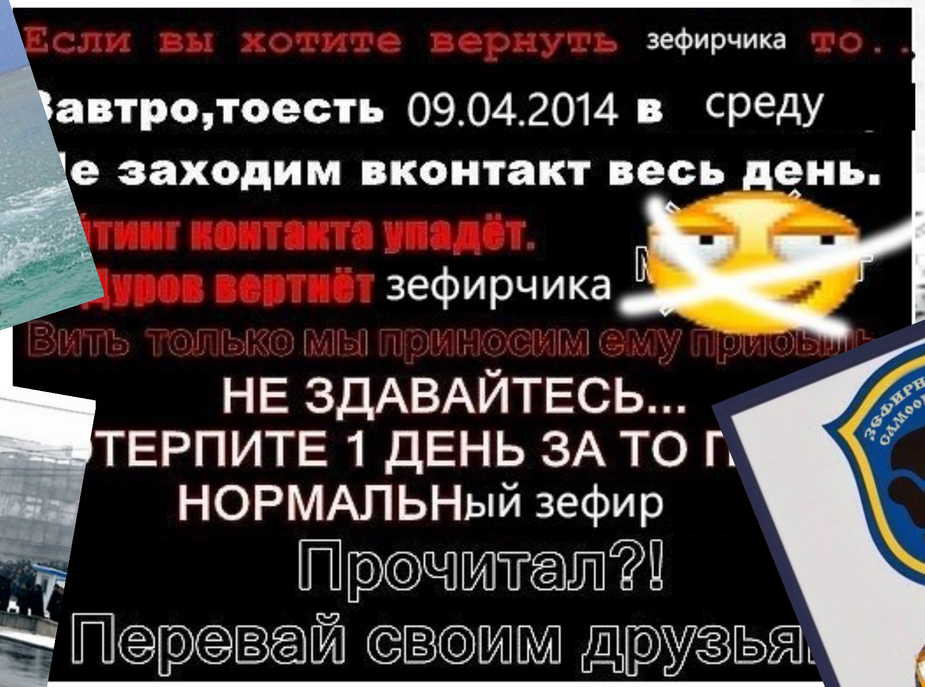
* 2012 for Google and Microsoft

Emoji: зачем?


Apple
Google
Samsung
Apple
Google
Samsung
Apple
Google
Samsung

Emoji: когда-то в VK в 2014

Emoji: когда-то в VK в 2014



Emoji: реализация

Emoji: реализация

Emoji: реализация

```
class EmojiSpan : ReplacementSpan {
    val emoji: Char,
    val map: Bitmap,
    val location: Rect,
}
```

Emoji: реализация

```
class EmojiSpan : ReplacementSpan {
    val emoji: Char,
    val map: Bitmap,
    val location: Rect,
}
```

“ok👌“ -> “ok [EmojiSpan]👌[/EmojiSpan]”

👌

НЕСТАНДАРТНЫЕ РЕШЕНИЯ

Разбиение сообщения
на несколько ViewHolder

Когда разбиваем на ViewHolder?

Глобальный RecycledViewPool

DiffUtil - не всегда полезен

Наложение ViewHolder

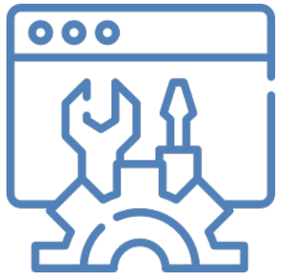
IdleHandler*

Emoji

О ЧЕМ ПОЙДЕТ РЕЧЬ

Стандартные решения

Нестандартные решения



Альтернативы

АЛЬТЕРНАТИВЫ

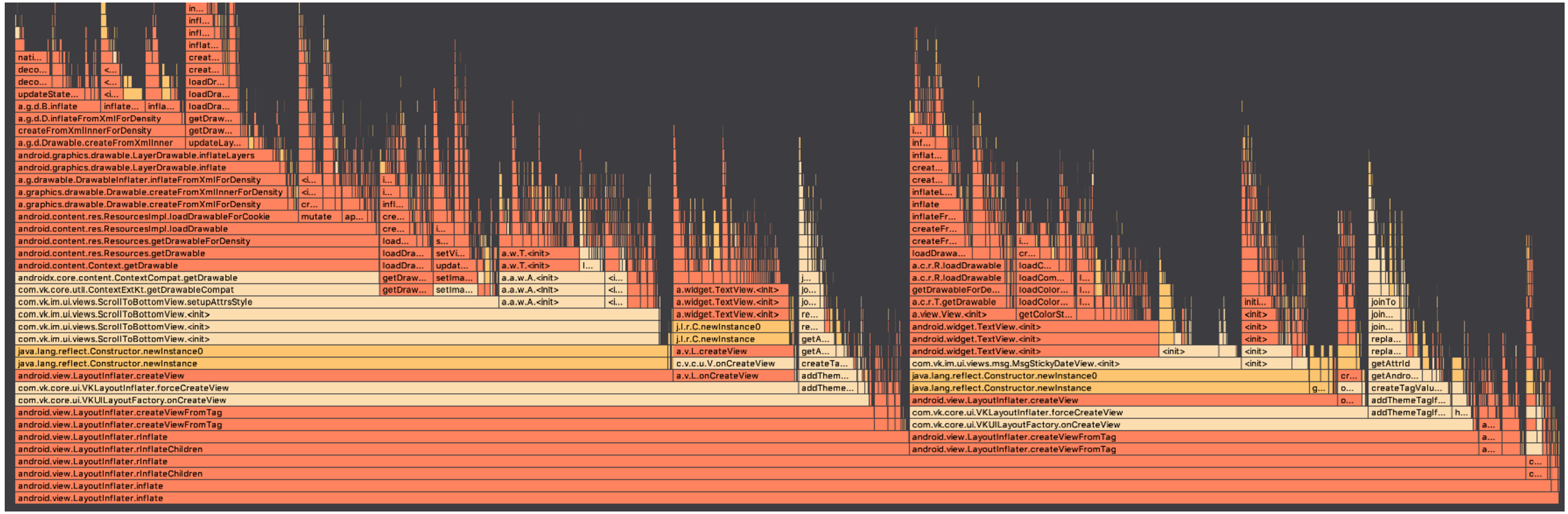
inflate -> ручное создание

inflate -> ручное создание

```
Debug.startMethodTracing(“trace”)
inflate(...)
Debug.stopMethodTracing()
```

inflate -> ручное создание

setupAttrsStyle		loadDrawable loadColor loadColorStateList	etc
loadDrawable	loadColor loadColorStateList		



АЛЬТЕРНАТИВЫ

inflate -> ручное создание

Anko

АЛЬТЕРНАТИВЫ

inflate -> ручное создание

Anko

Litho / Flutter

АЛЬТЕРНАТИВЫ

inflate -> ручное создание

Anko

Litho / Flutter

Jetpack compose

АЛЬТЕРНАТИВЫ

inflate -> ручное создание

Anko

Litho / Flutter

Jetpack compose

Одна большая Custom View

Одна большая CustomView

- ☑ Сложно поддерживать

Одна большая CustomView

- ☑ Сложно поддерживать
- ☑ Не имеет смысла в текущих реалиях

Одна большая CustomView

- [x] Сложно поддерживать
- [x] Не имеет смысла в текущих реалиях:
  - Кэш прошлого onMeasure() внутри View.measure()

Одна большая CustomView

- ☑ Сложно поддерживать
- ☑ Не имеет смысла в текущих реалиях:
  - Кэш прошлого onMeasure() внутри View.measure()
  - RenderNode/DisplayList внутри View.draw()

АЛЬТЕРНАТИВЫ

inflate -> ручное создание

Anko

Litho / Flutter

Jetpack compose

Одна большая Custom View

ИТОГИ

ИТОГИ

RecyclerView + стандартные оптимизации

ИТОГИ

RecyclerView + стандартные оптимизации

Разбиение сообщения на ViewHolder-ы

ИТОГИ

RecyclerView + стандартные оптимизации

Разбиение сообщения на ViewHolder-ы

Пре-процессинг на @WorkerThread

ИТОГИ

RecyclerView + стандартные оптимизации

Разбиение сообщения на ViewHolder-ы

Пре-процессинг на @WorkerThread

Глобальный RecycledViewPool

ИТОГИ

RecyclerView + стандартные оптимизации

Разбиение сообщения на ViewHolder-ы

Пре-процессинг на @WorkerThread

Глобальный RecycledViewPool

UI составляющие: ItemDecoration / Emoji

ИТОГИ

RecyclerView + стандартные оптимизации

Разбиение сообщения на ViewHolder-ы

Пре-процессинг на @WorkerThread

Глобальный RecycledViewPool

UI составляющие: ItemDecoration / Emoji

Альтернативы

РЕАЛИЗАЦИЯ КЭША СООБЩЕНИЙ

MOBIUS, SPB 2019

https://youtu.be/S7_hdLtDwcA

Q&A

СОРОКИН АЛЕКСАНДР,
ВКОНТАКТЕ

Principal Android Developer

http://vk.com/alexoro
uas.sorokin@gmail.com