BDUI на 100%
Рисуем шимеры с бекенда

Михаил Бесхитров
Android-разработчик

Яндекс Маркет

Хто я?

> Теперь 17-летние миддлы.

А потом месяца через три можно спросить «А он все еще работает?», дабы посмотреть, к чему это привело.

О чем пойдет речь?

Это птица?
Это самолет?
Нет, это шиммер!

Яндекс Маркет

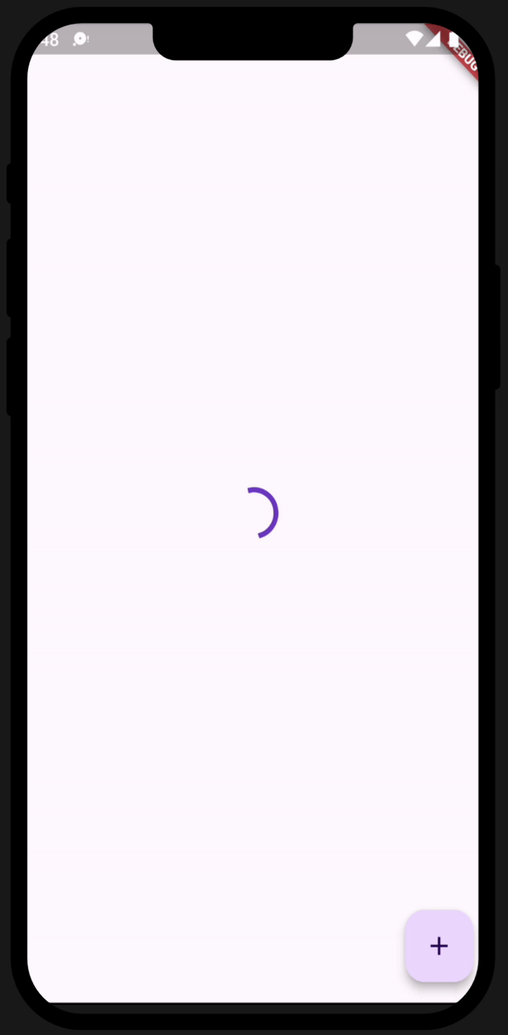
Что это?

Шиммер — это примитивный UI,
показываемый пользователю
во время загрузки основного контента

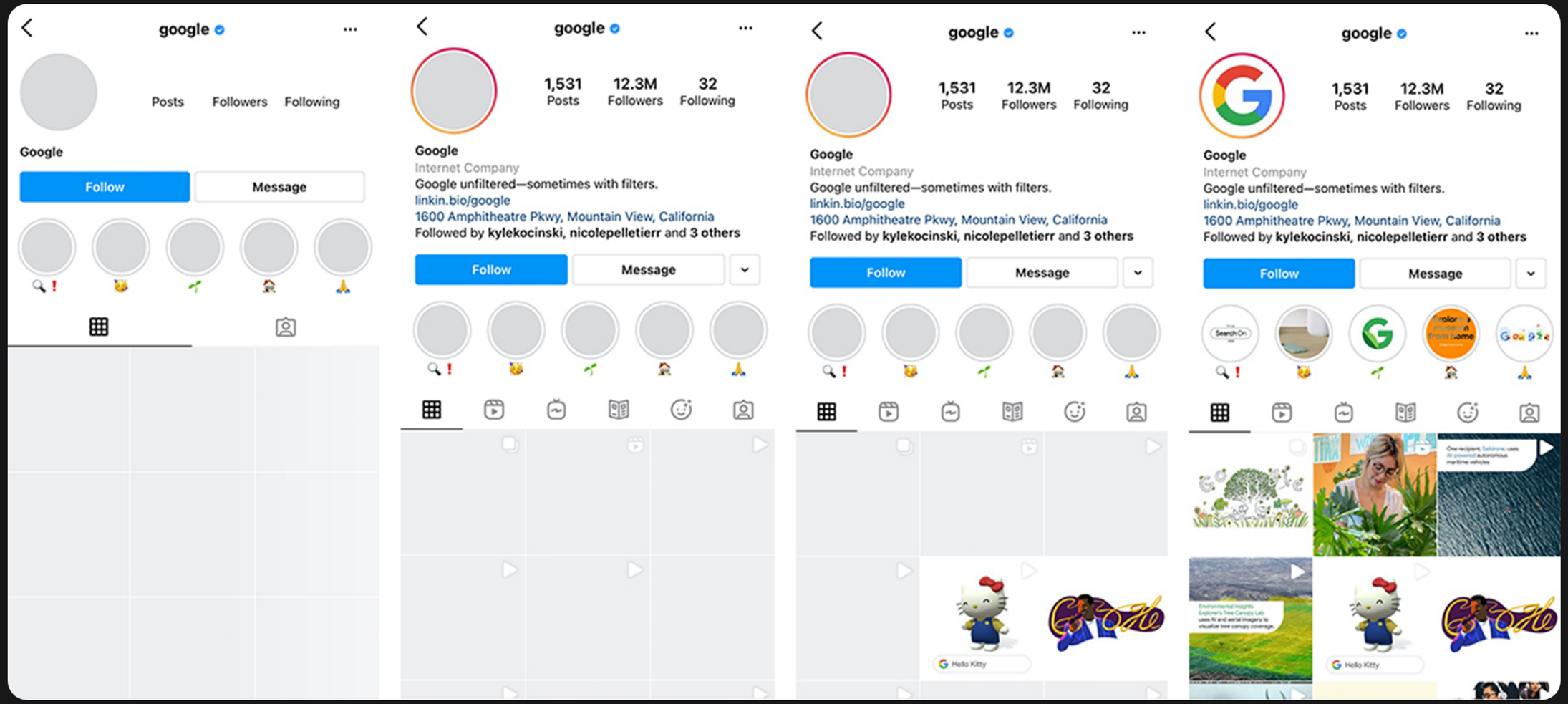
Зачем это нужно?

Яндекс Маркет

Шиммеры «ускоряют» приложение



Активное ожидание



Де-факто стандарт индустрии

БДУЙ грядет, бегите, глупцы

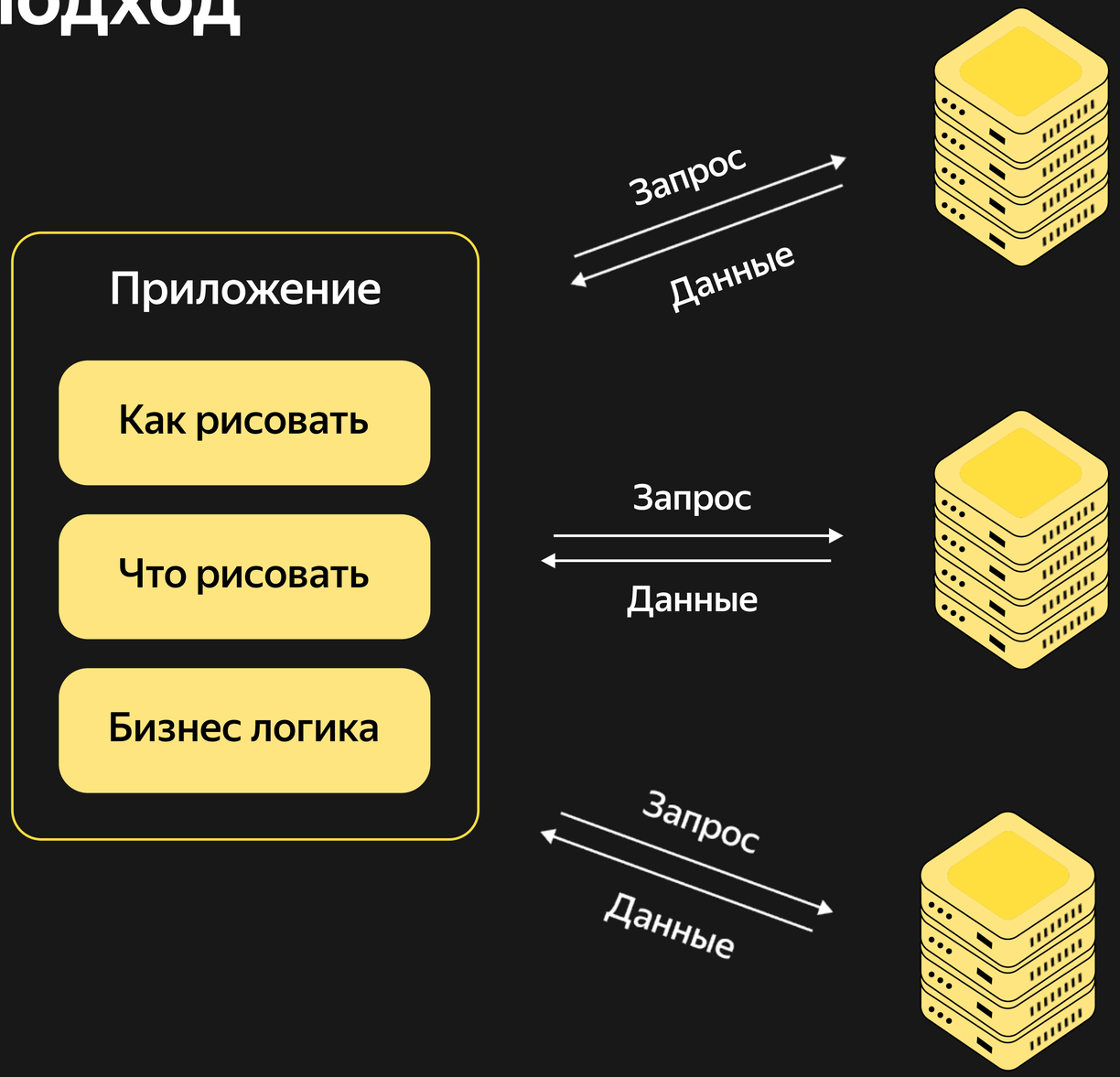
Яндекс Маркет

Кто такой этот ваш BDUI?

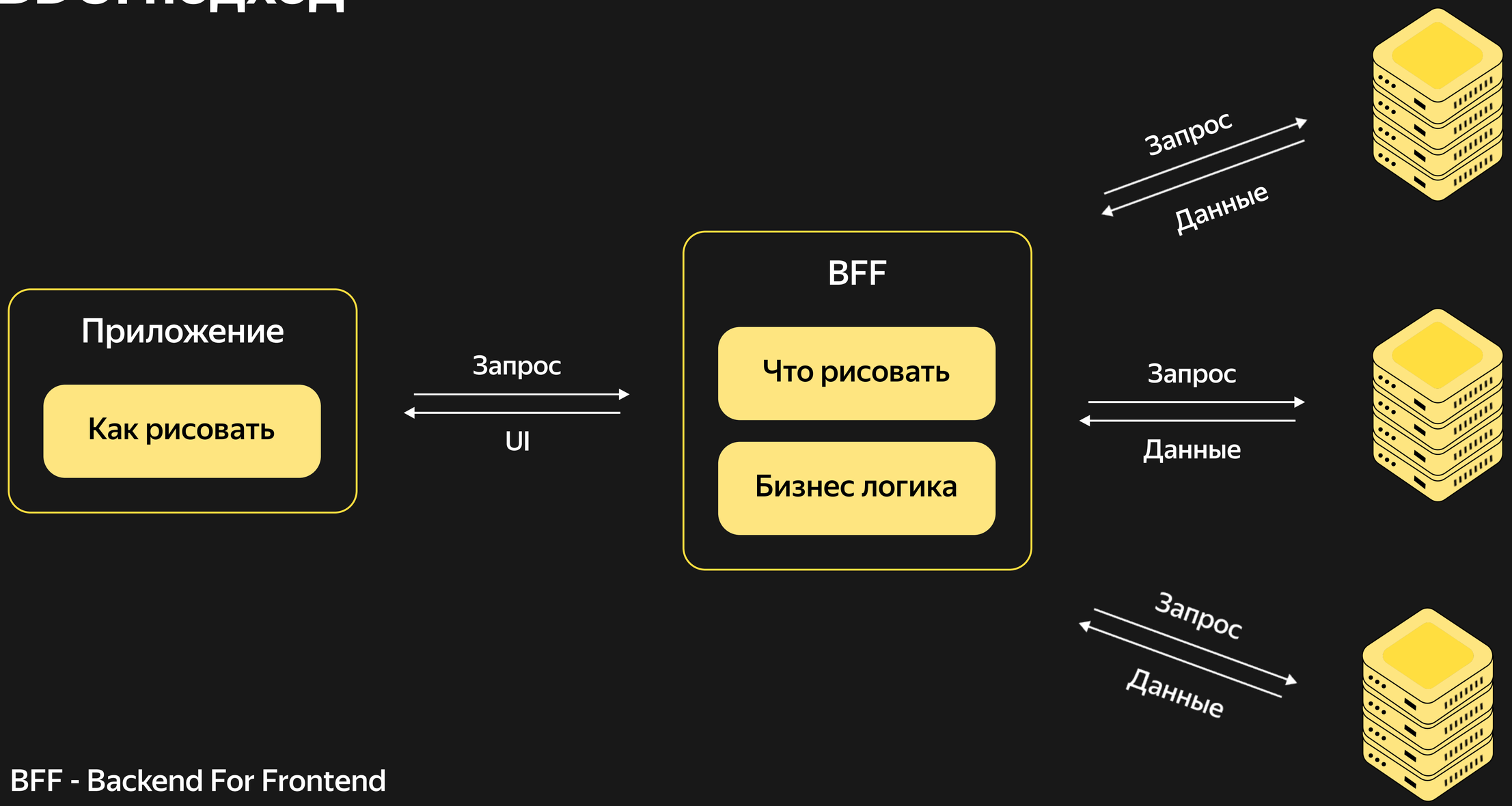
BDUI
приложение

обычное
приложение

Стандартный подход



BDUI подход



BFF - Backend For Frontend

Почему это лучше?

1

Релизы за час

2

Меньше TTM

3

Нет хвоста версий

4

Тонкий клиент

Шимеры против BDUI

Яндекс Маркет

```
object BannersSkeletonItem : Item()

class BannersSkeletonItemAdapter : ItemAdapter<BannersSkeletonItem, ViewHolder>() {
    override fun createViewHolder(parent: ViewGroup): ViewHolder {
        val res = R.layout.section_skeleton_banners_carousel
        return ViewHolder(createItemView(parent, res ))
    }
}
```

```
object BannersSkeletonItem : Item()

class BannersSkeletonItemAdapter : ItemAdapter<BannersSkeletonItem, ViewHolder>() {
    override fun createViewHolder(parent: ViewGroup): ViewHolder {
        val res = R.layout.section_skeleton_banners_carousel
        return ViewHolder(createItemView(parent, res ))
    }
}
```

```
object BannersSkeletonItem : Item()

class BannersSkeletonItemAdapter : ItemAdapter<BannersSkeletonItem, ViewHolder>() {
    override fun createViewHolder(parent: ViewGroup): ViewHolder {
        val res = R.layout.section_skeleton_banners_carousel
        return ViewHolder(createItemView(parent, res ))
    }
}
```

```
object BannersSkeletonItem : Item()

class BannersSkeletonItemAdapter : ItemAdapter<BannersSkeletonItem, ViewHolder>() {
    override fun createViewHolder(parent: ViewGroup): ViewHolder {
        val res = R.layout.section_skeleton_banners_carousel
        return ViewHolder(createItemView(parent, res ))
    }
}
```

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    documentEngine.setLoadingIndicatorController(
       DefaultLoadingIndicatorController(
           bindings = listOf(
               BannersSkeletonItemAdapter() forType BannersSkeletonItem::class,
           ),
           items = listOf(
                BannersCarouselSkeletonItem,
           )
       )
    )
}
```

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    documentEngine.setLoadingIndicatorController(
       DefaultLoadingIndicatorController(
           bindings = listOf(
               BannersSkeletonItemAdapter() forType BannersSkeletonItem::class,
           ),
           items = listOf(
                BannersCarouselSkeletonItem,
           )
       )
    )
}
```

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    documentEngine.setLoadingIndicatorController(
        DefaultLoadingIndicatorController(
            bindings = listOf(
                BannersSkeletonItemAdapter() forType BannersSkeletonItem::class,
            ),
            items = listOf(
                 BannersCarouselSkeletonItem,
            )
        )
    )
}
```

```
override fun onAttach(context: Context) {
    super.onAttach(context)
    documentEngine.setLoadingIndicatorController(
        DefaultLoadingIndicatorController(
            bindings = listOf(
                BannersSkeletonItemAdapter() forType BannersSkeletonItem::class,
            ),
            items = listOf(
                 BannersCarouselSkeletonItem,
            )
        )
    )
}
```

Почему не устраивает

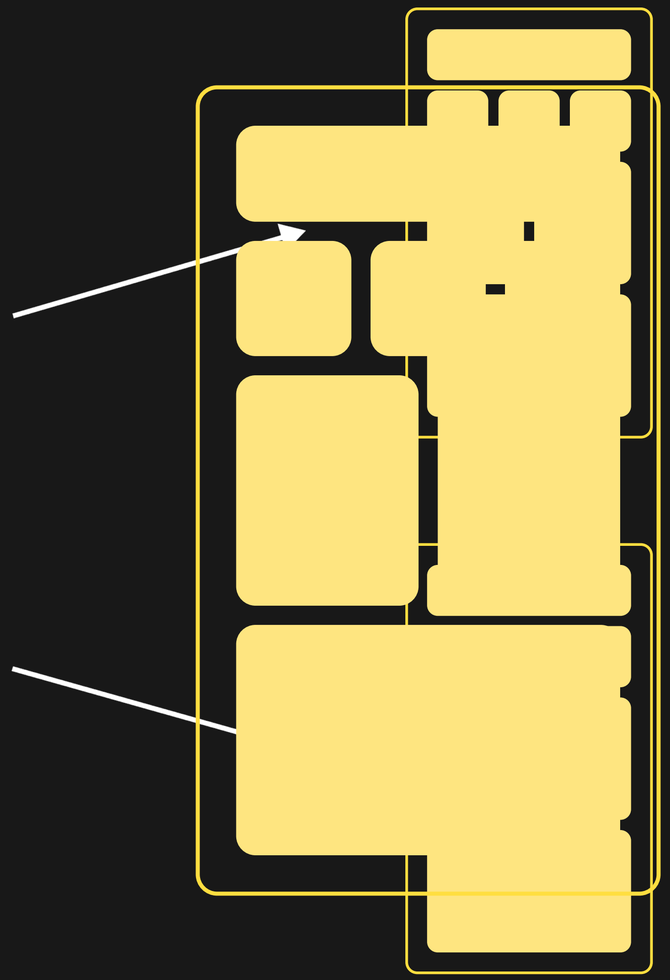
1. Зависимость от релизов
2. Нет никакой переиспользуемости
3. Экран существует на трех платформах
4. Какой-то костыль в движке

Не нравится технология?

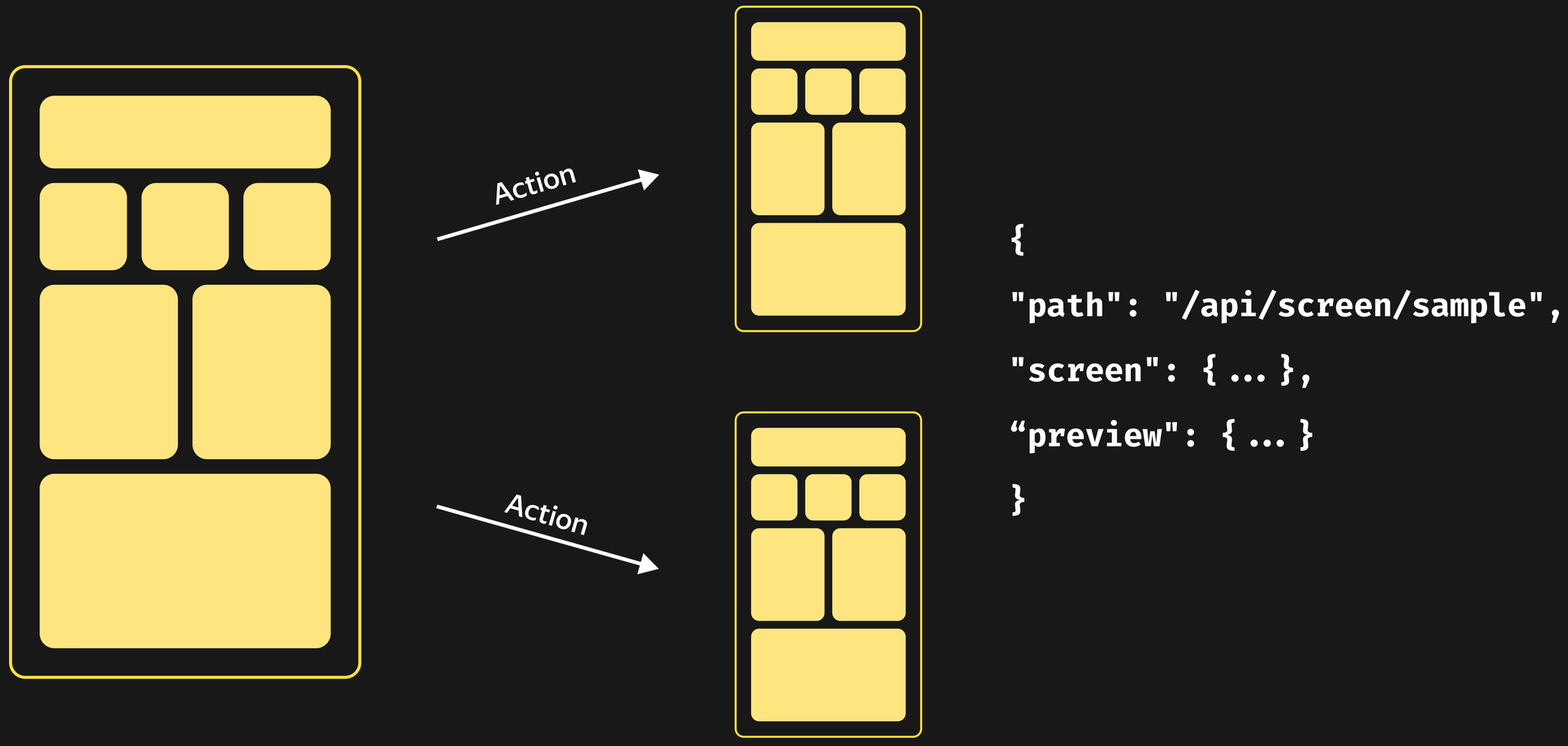
Сделай свой велосипед!

Яндекс Маркет

Концепция



Концепция

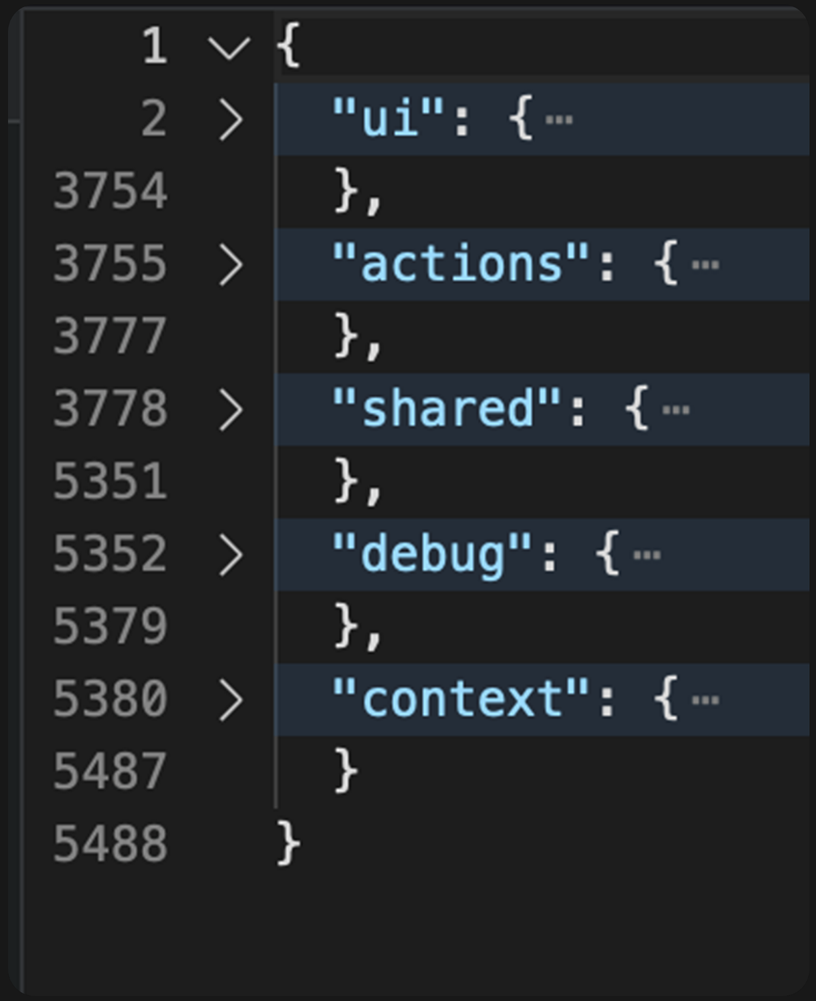


Требования

1

Минимальное место
в ответе бекенда


{
"shared": {…
},
"actions": {…
},
"debug": {…
},
"context": {…
},
"ui": {…
},
"scaffold": {…
}
}


{
"ui": {…
},
"actions": {…
},
"shared": {…
},
"debug": {…
},
"context": {…
}
}

Требования

1. Минимальное место в ответе бекенда
2. Быстрый парсинг
3. Простота использования

Делаем велосипед

Яндекс Маркет

Json

Знакомый формат

Уже общаемся везде через json

Занимает много места

Долго парсится

Протобуфы

Скорость парсинга

Маленький размер

Никакой инфры

Никакой интеграции в фреймворк

Команде будет сложно понимать это

Свое решение

Можем сделать оптимально по размеру

Можем сделать быстрое решение

Можем выступить на мобиусе

Надо писать с нуля

Skeletor

Делаем свой формат данных

Яндекс Маркет

Json схема

```
{
     "type": "container",
     "orientation": "vertical",
     "items": [
        {
             "type": "rect",
             "width": 40,
             "height": 60,
             "corners": 20
        },
        ...
      ]
}
```

```
{
     "type": "column",
     "items": [
        {
             "type": "rect",
             "width": 40,
             "height": 60,
             "corners": 20
        },
        ...
     ]
}
```

Json схема

```
{
     "type": “column",
     "items": [
        {
             "type": “rect",
             "width": 40,
             "height": 60,
             "corners": 20
        },
        ...
     ]
}
```

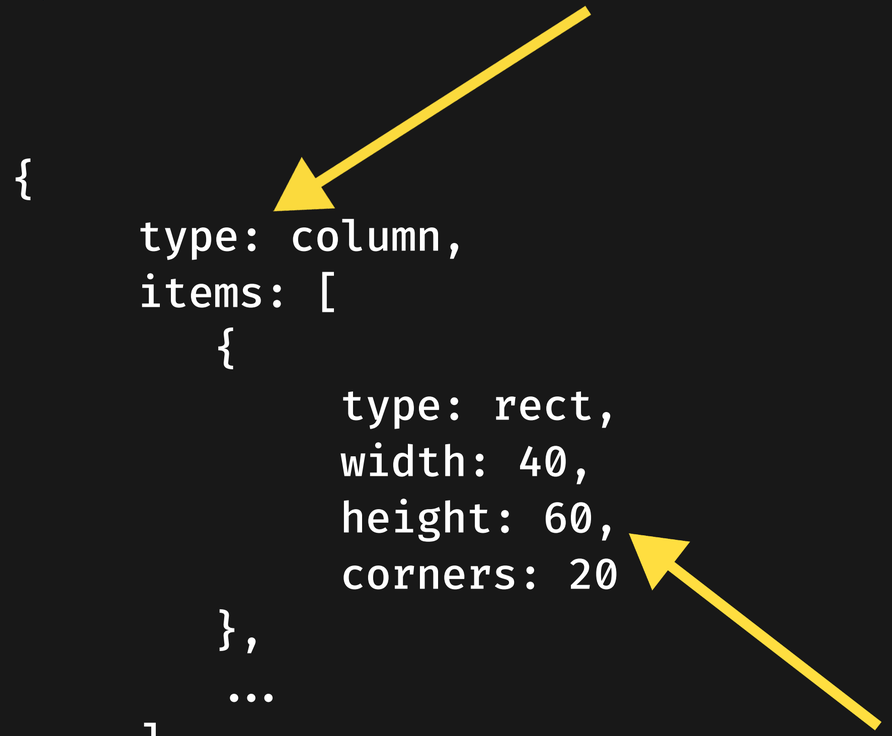
Скелетон — это **строчка** в JSON’е

Json схема

```
{
    "type": “column",
    "items": [
        {
            "type": “rect",
            "width": 40,
            "height": 60,
            "corners": 20
        },
        ...
    ]
}
```

```
{
    type: column,
    items: [
        {
            type: rect,
            width: 40,
            height: 60,
            corners: 20
        },
        ...
    ]
}
```

~~Json~~ схема



```
{
      type: column,
      items: [
         {
              type: rect,
              width: 40,
              height: 60,
              corners: 20
         },
          ...
      ]
}
```

~~Json~~ схема

```
{type:column,items:[{type:rect,width:40,height:
60,corners:20}, ... ]}
```

~~Json~~ схема

```
{
      type: column,
      items: [
         {
            type: rect,
            width: 40,
            height: 60,
            corners: 20
         },
          ...
      ]
}
```

```
{type column items
[{type rect width
40 height 60
corners 20}…]}
```

Json схема

```
{type column items [{type rect width 40 height 60
corners 20}…]}
```

Json схема

```
{
      type: column,
      items: [
         {
            type: rect,
            width: 40,
            height: 60,
            corners: 20
         },
          ...
      ]
}
```

```
{type column items
[{type rect width
40 height 60
corners 20}…]}
```

Json схема

```
{type column items
[{type rect width
40 height 60
corners 20}…]}
```

```
{column [{rect 40
60 20}…]}
```

Json схема

```
{[выражение] [аргумент1] [аргумент2] ... [аргументN]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

Json схема

```
{[выражение] [аргумент1] [аргумент2] ... [аргументN]}
```

~~Json~~ схема

```
{column [{rect 40 60 20}…]}
```

Json схема

```
{column {rect 40 60 20}…}
```

Json схема

```
{
   "type": "container",
   "orientation": "vertical",
   "items": [
      {
         "type": "rect",
         "width": 40,
         "height": 60,
         "corners": 20
      },
      ...
   ]
}
```

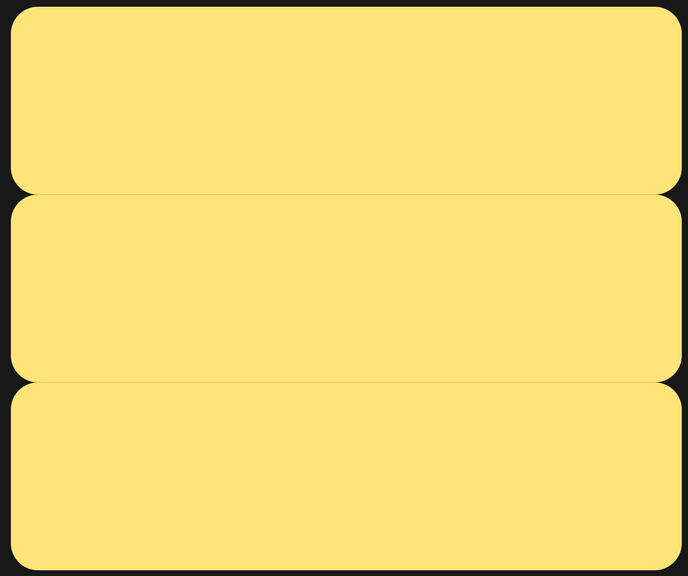
```
{column {rect 40
60 20}…}
```

~~Lisp~~ SkeletorLang

```
(col (bone 40 60 20) … )
```

SkeletorLang

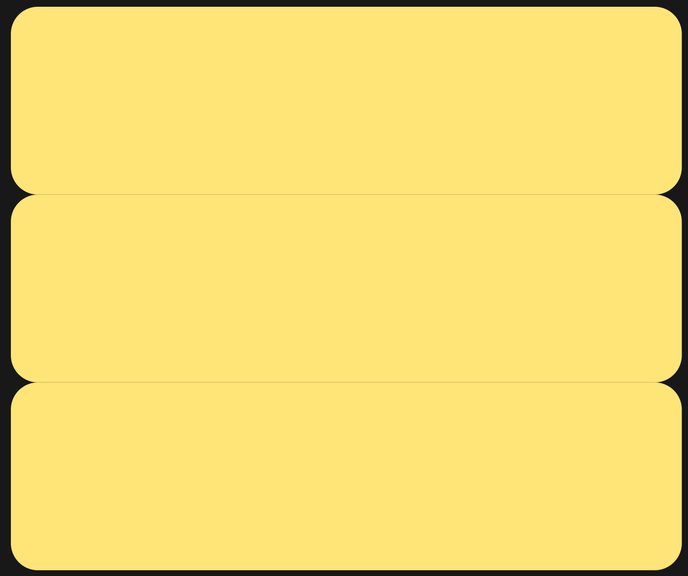
```
(col (bone 100 10 20) (bone 100 10 20) (bone 100 10 20))
```



Space

```
(col (bone 100 10 20) (space 10) (bone 100 10 20) (space
10) (bone 100 10 20))
```

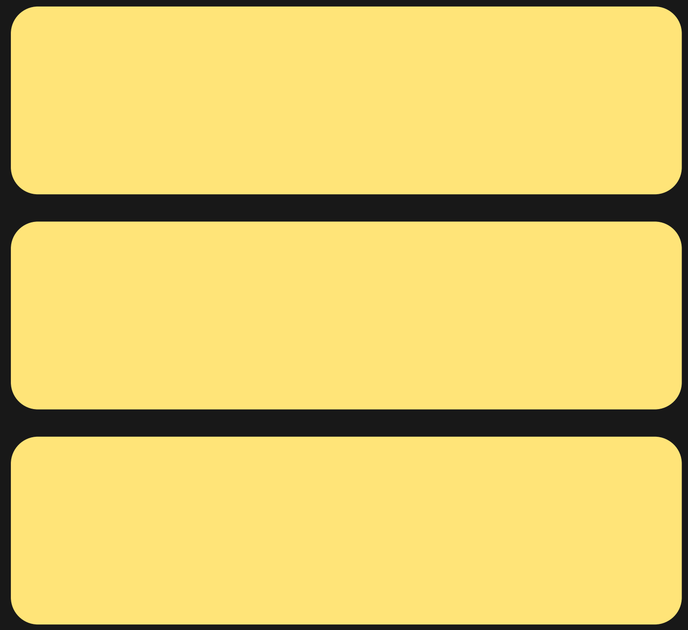
(space <значение>)



Repeat

```
(col (repeat 3 (col (bone 100 10 20) (space 10))))
```

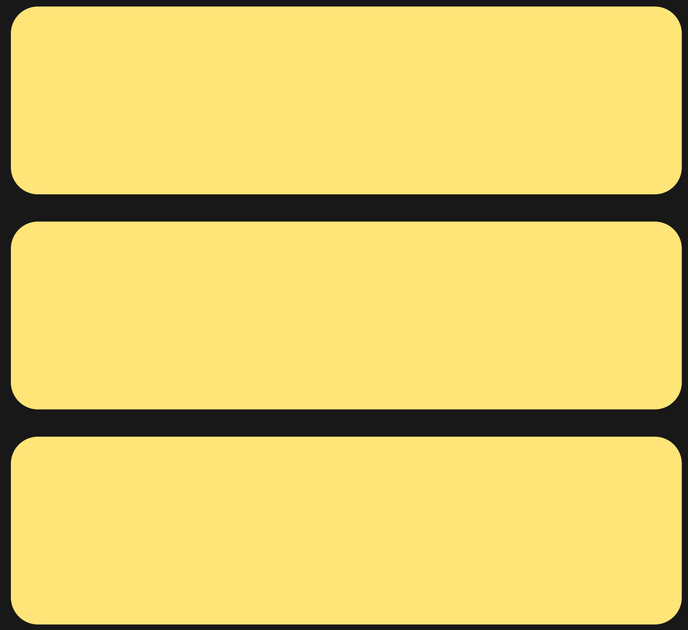
(repeat <сколько> <выражение>)



Padded

```
(padded <все> <выражение>)
(padded <вертикаль> <горизонталь> <выражение>)
(padded <лево> <верх> <низ> <право> <выражение>)
```

```
(col (repeat 3 (padded 10 (hbox 100 10 20))))
```

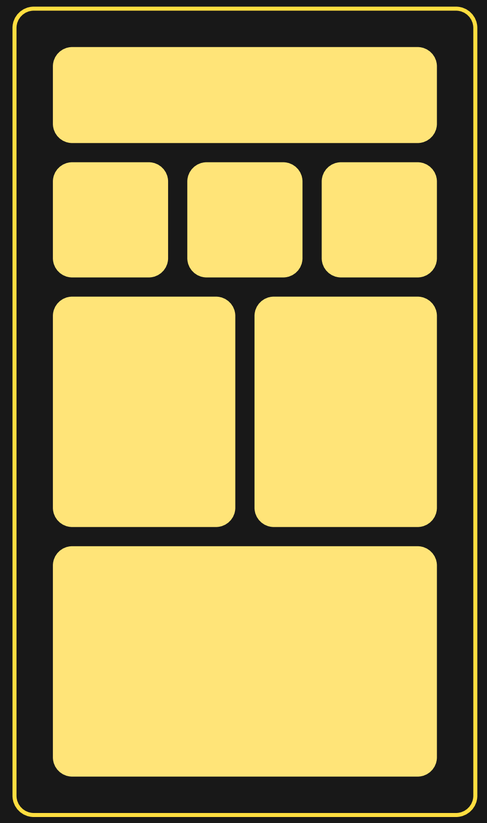


Цвет

```
(col (repeat 3 (padded 10 (bone 100 10 20 #BBBBBB))))
```

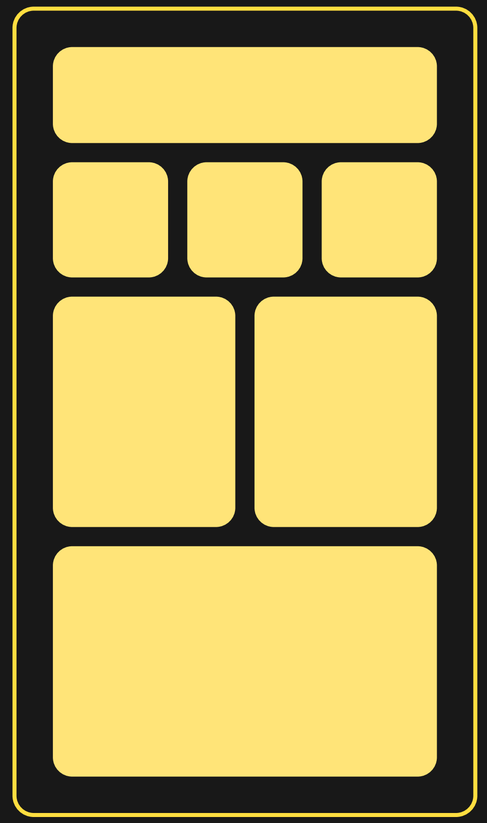
Цвет

```
(padded 10 (col (bone 400 20 10 #BBBBBB) (row (repeat 3
(padded 5 (bone 40 40 10 #BBBBBB)))) (row (repeat 2
(padded 5 (bone 100 150 10 #BBBBBB)))) (space 10) (bone
400 20 10 #BBBBBB)))
```



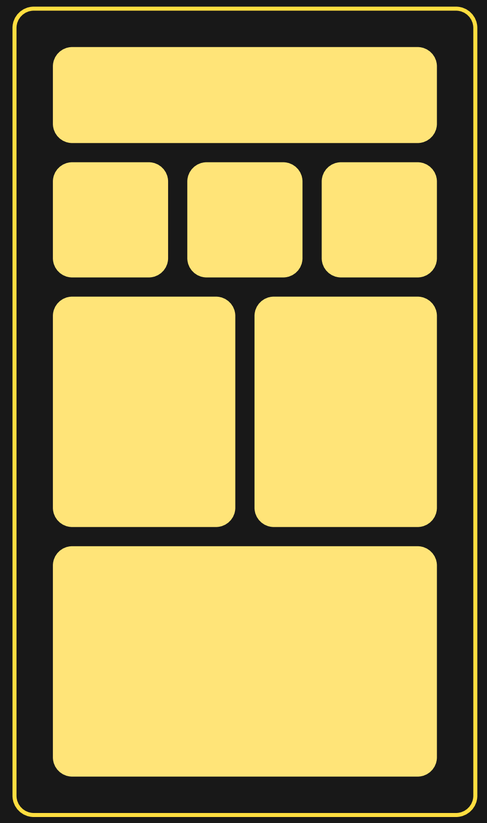
Стиль

```
(padded 10 (col (bone 400 20 10 #BBBBBB) (row (repeat 3
(padded 5 (bone 40 40 10 #BBBBBB)))) (row (repeat 2
(padded 5 (bone 100 150 10 #BBBBBB)))) (space 10) (bone
400 20 10 #BBBBBB)))
```



Theme

```
(theme <цвет> <закругления> <выражение>)
(theme #BBBBBB 10 (padded 10 (col(bone 400 20) (row
(repeat 3 (padded 5 (bone 40 40)))) (row (repeat 2
(padded 5 (bone 100 150)))) (space 10) (bone 400 20))))
```



Учимся парсить

Яндекс Маркет

Вдохновение

20. Valid Parentheses

Hint

Easy 22.5K 1.5K

Companies

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Главные идеи

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Определение типа выражения

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Определение типа выражения

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Определение типа выражения

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Определение типа выражения

```
token = skeleton[i]
if token == '(' {
    sb = ""
    while ++i < skeleton.lastIndex and skeleton[i] != ' ' {
        sb += skeleton[i]
    }
    expression = expressionFromStr(sb)
}
```

Определение типа выражения

```
token = skeleton[i]
if token == '(' {
    sb = ""
    while ++i < skeleton.lastIndex and skeleton[i] != ' ' {
        sb += skeleton[i]
    }
    expression = expressionFromStr(sb)
}
```

Определение типа выражения

```
token = skeleton[i]
if token == '(' {
    sb = ""
    while ++i < skeleton.lastIndex and skeleton[i] != ' ' {
        sb += skeleton[i]
    }
    expression = expressionFromStr(sb)
}
```

Определение типа выражения

```
token = skeleton[i]
if token == '(' {
    sb = ""
    while ++i < skeleton.lastIndex and skeleton[i] != ' ' {
        sb += skeleton[i]
    }
    expression = expressionFromStr(sb)
}
```

Аргументы

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Аргументы

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Аргументы

```
(theme #BBBBBB 10 (padded 10 (col (bone 400 20))))
```

Аргументы

```
token = skeleton[i]
if token == '(' {…}
elif token == ')' {…}
else {
   sb = ''
   args = []
   while i < skeleton.lastIndex and skeleton[i] ≠ ' ' and skeleton[i] ≠ ')' {
      sb += skeleton[i++]
   }
   args.add(sb)
}
```

Аргументы

```
token = skeleton[i]
if token == '(' {…}
elif token == ')' {…}
else {
   sb = ''
   args = []
   while i < skeleton.lastIndex and skeleton[i] ≠ ' ' and skeleton[i] ≠ ')' {
      sb += skeleton[i++]
   }
   args.add(sb)
}
```

Аргументы

```
token = skeleton[i]
if token == '(' {…}
elif token == ')' {…}
else {
   sb = ''
   args = []
   while i < skeleton.lastIndex and skeleton[i] ≠ ' ' and skeleton[i] ≠ ')' {
      sb += skeleton[i++]
   }
   args.add(sb)
}
```

Аргументы

```
token = skeleton[i]
if token == '(' {…}
elif token == ')' {…}
else {
   sb = ''
   args = []
   while i < skeleton.lastIndex and skeleton[i] ≠ ' ' and skeleton[i] ≠ ')' {
      sb += skeleton[i++]
   }
   args.add(sb)
}
```

Финализация

```
expressions = stack()
args = stack()
if token == ‘(‘ {
   …
   expressions.add(expression)
   args.add([])
}
elif token == ‘)’ {…}
else {
   …
   args[len(args) - 1].add(sb)
}
```

Финализация

```
expressions = stack()
args = stack()
if token == ‘(‘ {
   …
   expressions.add(expression)
   args.add([])
}
elif token == ‘)’ {…}
else {
   …
   args[len(args) - 1].add(sb)
}
```

Финализация

```
expressions = stack()
args = stack()
if token == '(' {
   …
   expressions.add(expression)
   args.add([])
}
elif token == ')' {…}
else {
   …
   args[len(args) - 1].add(sb)
}
```

Финализация

```
ready = stack()
if token == '(' {…}
elif token == ')' {
   expression = expressions.pop()
   args = args.pop()
   used = finalize(expression, args, ready)
   if used {
      ready = stack()
   }
   ready.add(expression)
}
```

Финализация

```
ready = stack()
if token == '(' {…}
elif token == ')' {
   expression = expressions.pop()
   args = args.pop()
   used = finalize(expression, args, ready)
   if used {
      ready = stack()
   }
   ready.add(expression)
}
```

Финализация

```
ready = stack()
if token == ‘(‘ {…}
elif token == ‘)’ {
   expression = expressions.pop()
   args = args.pop()
   used = finalize(expression, args, ready)
   if used {
      ready = stack()
   }
   ready.add(expression)
}
```

Финализация

```
ready = stack()
if token == ‘(‘ {…}
elif token == ‘)’ {
   expression = expressions.pop()
   args = args.pop()
   used = finalize(expression, args, ready)
   if used {
      ready = stack()
   }
   ready.add(expression)
}
```

Пишем код?

Яндекс Маркет

Генерируем код!

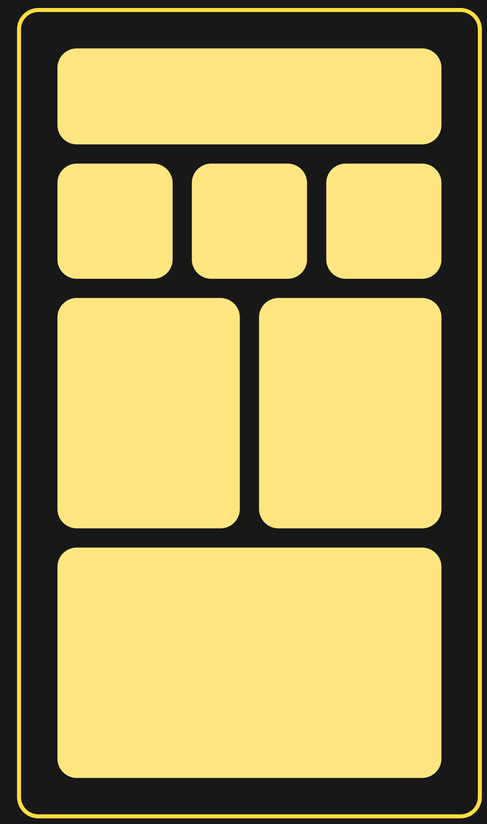
Яндекс Маркет

Почему? За что?

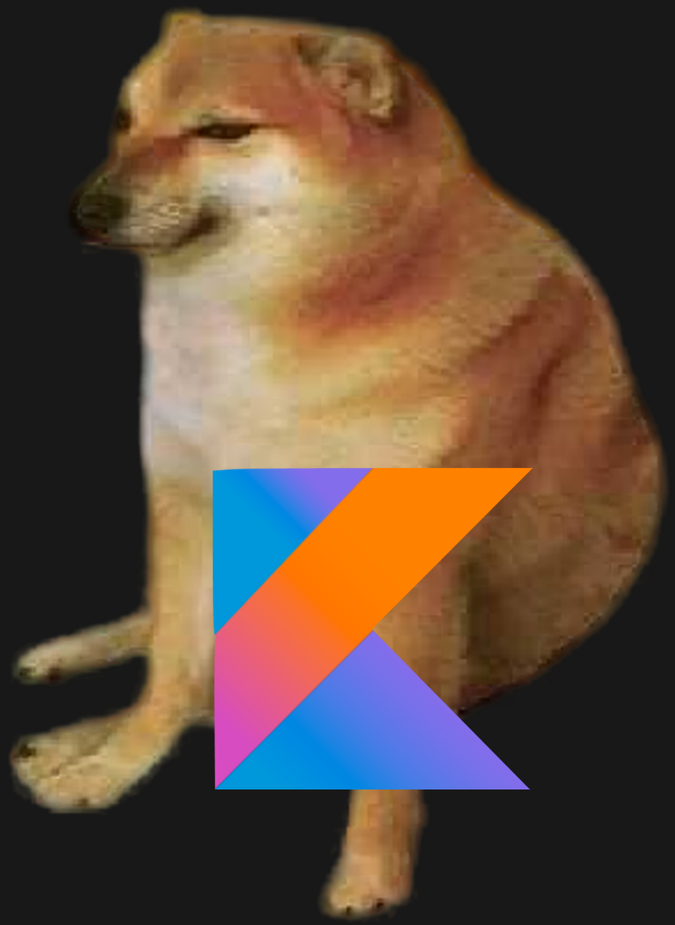
1. Много одинакового кода
2. Независимость от платформы
3. Мы хотим превью!

Превью

```
(theme #BBBBBB 10 (padded 10 (col (bone 400
20) (row (repeat 3 (padded 5 (bone 40 40))))
(row (repeat 2 (padded 5 (bone 100 150))))
(space 10) (bone 400 20))))
```



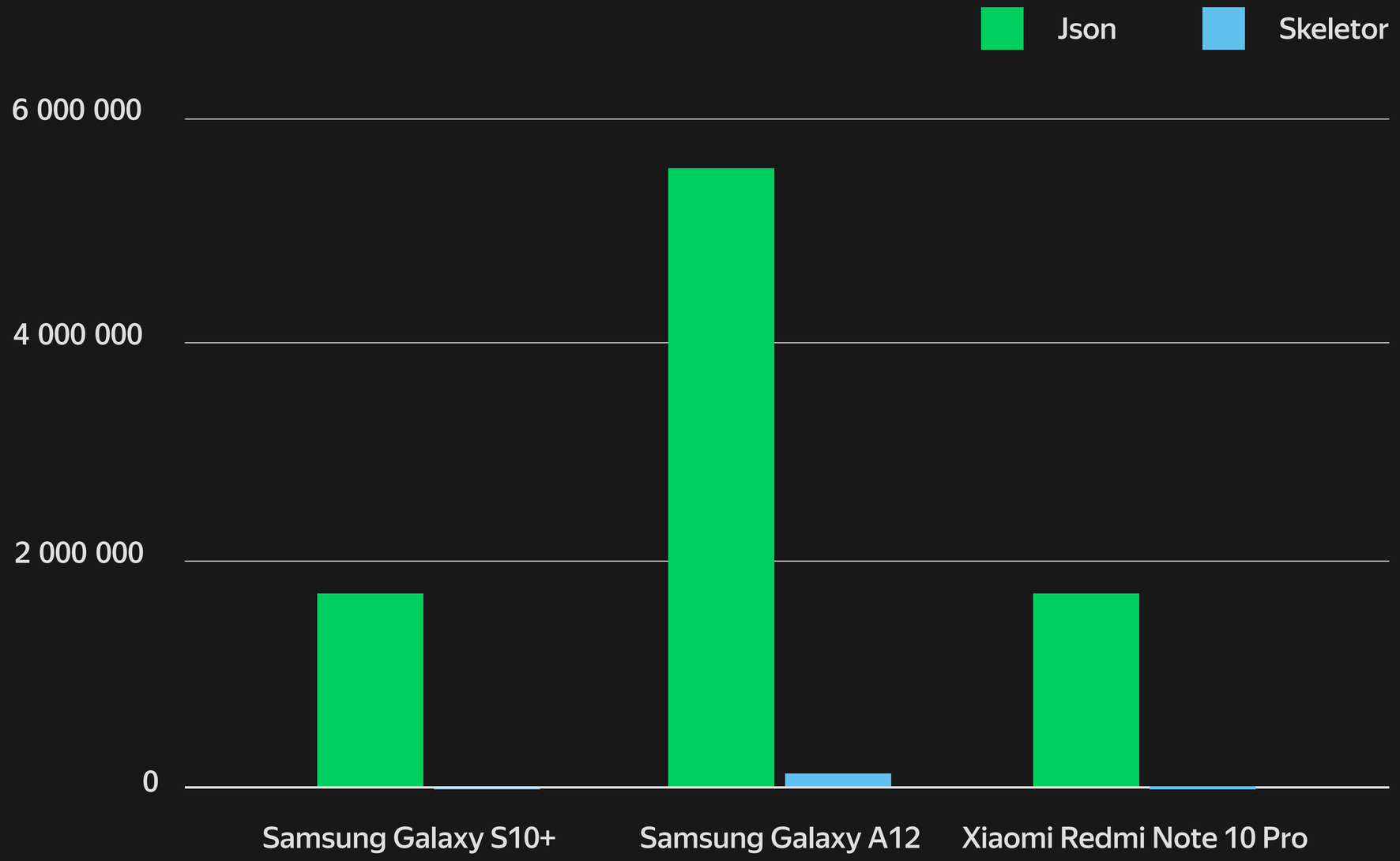

Python


Kotlin

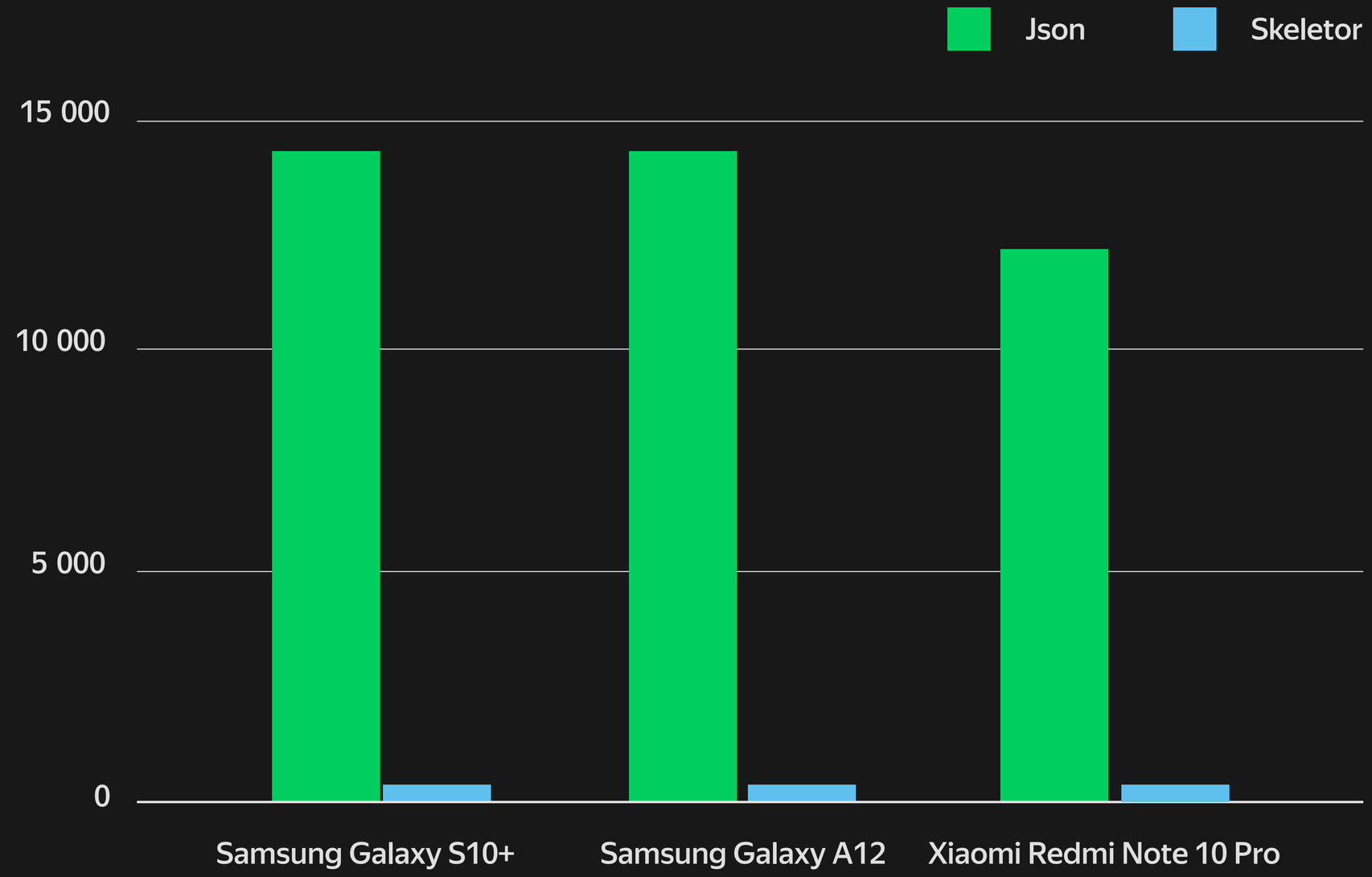
Эффективность решения

Яндекс Маркет

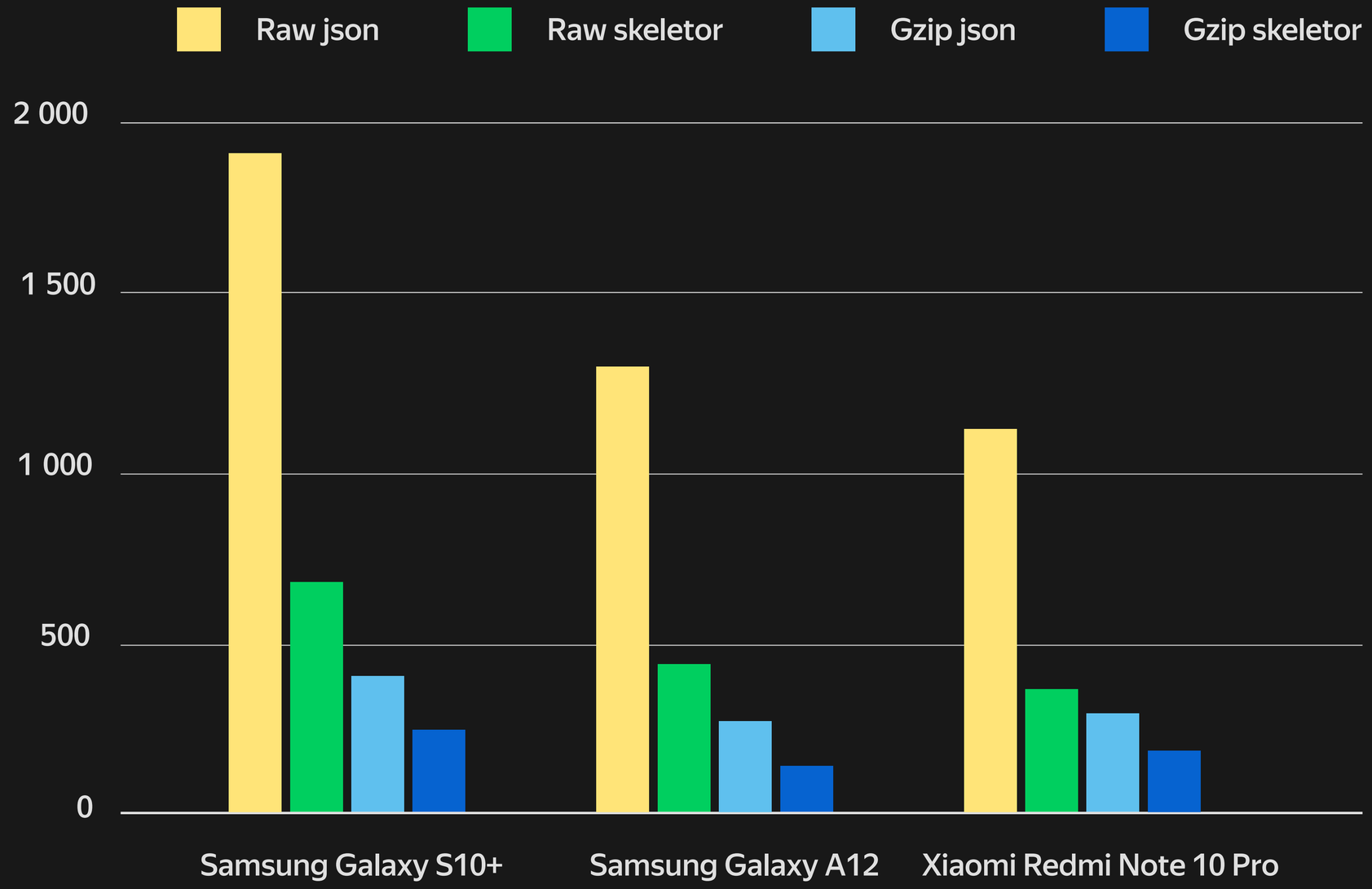
Скорость парсинга



Количество алокаций



Размер


Raw json
Raw skeletor
Gzip json
Gzip skeletor
2 000
1 500
1 000
500
0
Samsung Galaxy S10+
Samsung Galaxy A12
Xiaomi Redmi Note 10 Pro

Спасибо!

Михаил Бесхитров

@eshendo