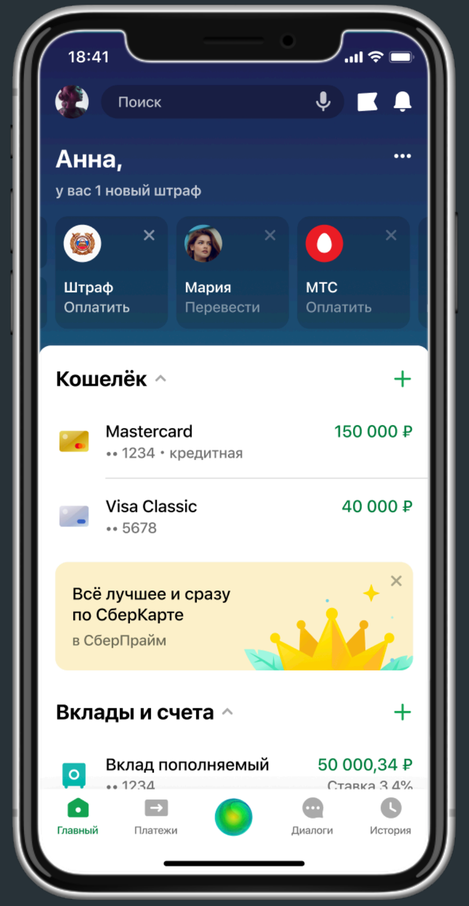
Оптимизируем размер приложения на практике

Оптимизируем размер приложения на практике

Каплан Дениз
Инженер платформенной команды Сбербанк Онлайн

СберБанк

- 200 iOS разработчиков
- 47 feature в спринт
- 200 таргетов
- 97 feature-модулей
- Более 3 млн. строк кода
- 1 core команда



В чем проблема?

- Недовольство пользователей
- Ограничение Apple
- Огромный прирост каждый релиз
- Отсутствие любых метрик
- Невозможность контролировать рост

В чем проблема?

Напичкали стольким количеством ерунды что слов нет от чего вес приложения стал большим. Сделайте отдельное приложение для тех кому нужен только сберБАНК! Я так понимаю за смену логотипа и добавление новых не связанный с банком функций вы решили содрать с обычных людей при помощи комиссии? За все стала комиссия скоро за вход в банк онлайн будете снимать! Я уже пробую другие банки, да и все мои друзья отказываются от вас из за комиссии и ненужных нововведений, сейчас 60% моих друзей перешли на тинькофф и Райффайзен когда раньше вашим тогда ещё банком

- Недовольство пользователей
- Раз
- Огромный прирост каждый релиз
-
- Невозможность контролировать рост

Занимает много памяти

Почему, ну почему приложение такое тяжелое?! И с каждым обновлением становится все тяжелее. Такими темпами на моем нищебродском айфоне 6s останется только одно приложение. А хочется оставить хотя бы калькулятор.

Расширьте загрузку более 200 мегабайт по обычной сети, а не только по wi-fi !!!!

Расширьте загрузку более 200 мегабайт по обычной сети, а не только по wi-fi !!!! Сейчас безлимитный интернет у 80% населения, а тарификация по гигабайтам!!!

Память

Вы серьезно? Как такое приложение может весить целых 225 мегабайт, в то время как обычные игры весят не больше 100 мегабайт

Что такое feature-модуль?

- Свой репозиторий
- Отдельное тестовое приложение
- Своя архитектура
- Могут поставляться в проект как исходниками, так и бинарниками
- Метафайл для работы с межмодульностью

План доклада

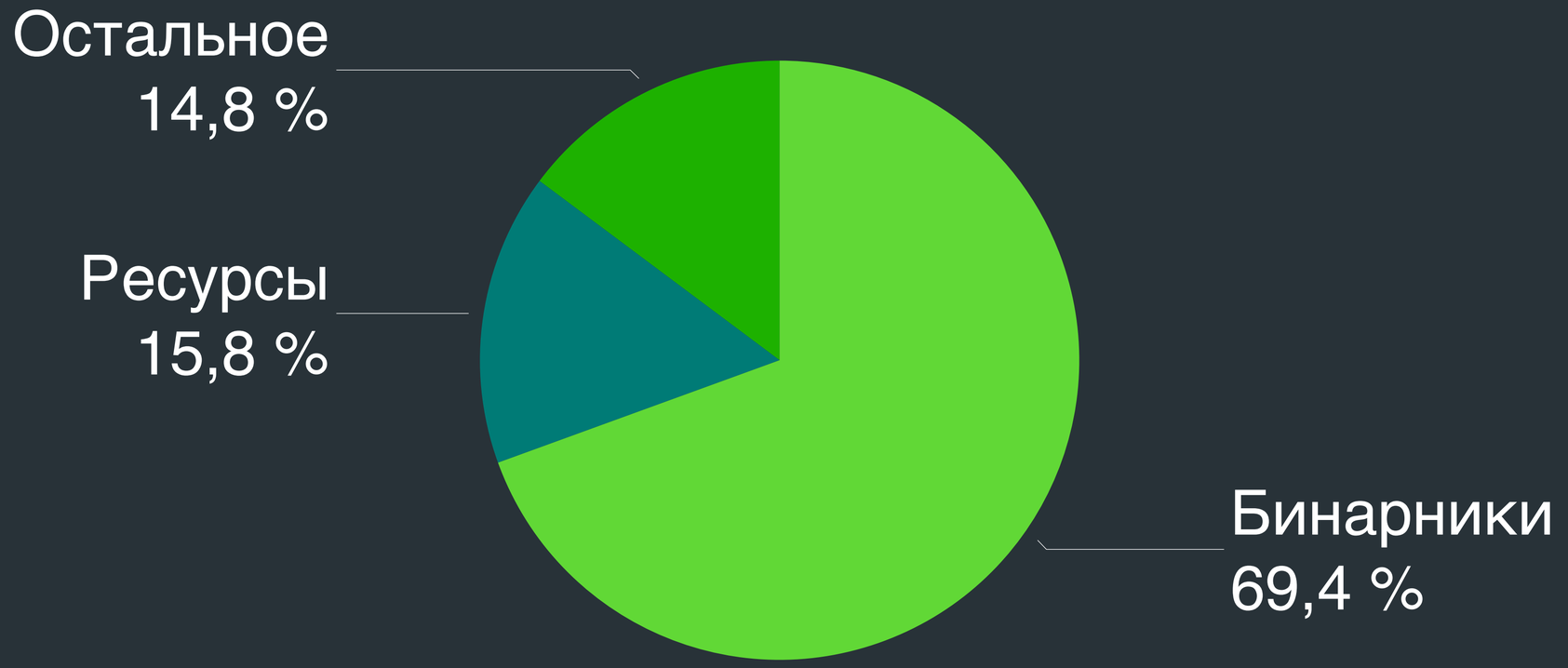
- Что внутри приложений?
- Как можно оптимизировать размер?
- Оптимизируем размер СберБанка
- Что делать после?

Что внутри?

Файлы приложения

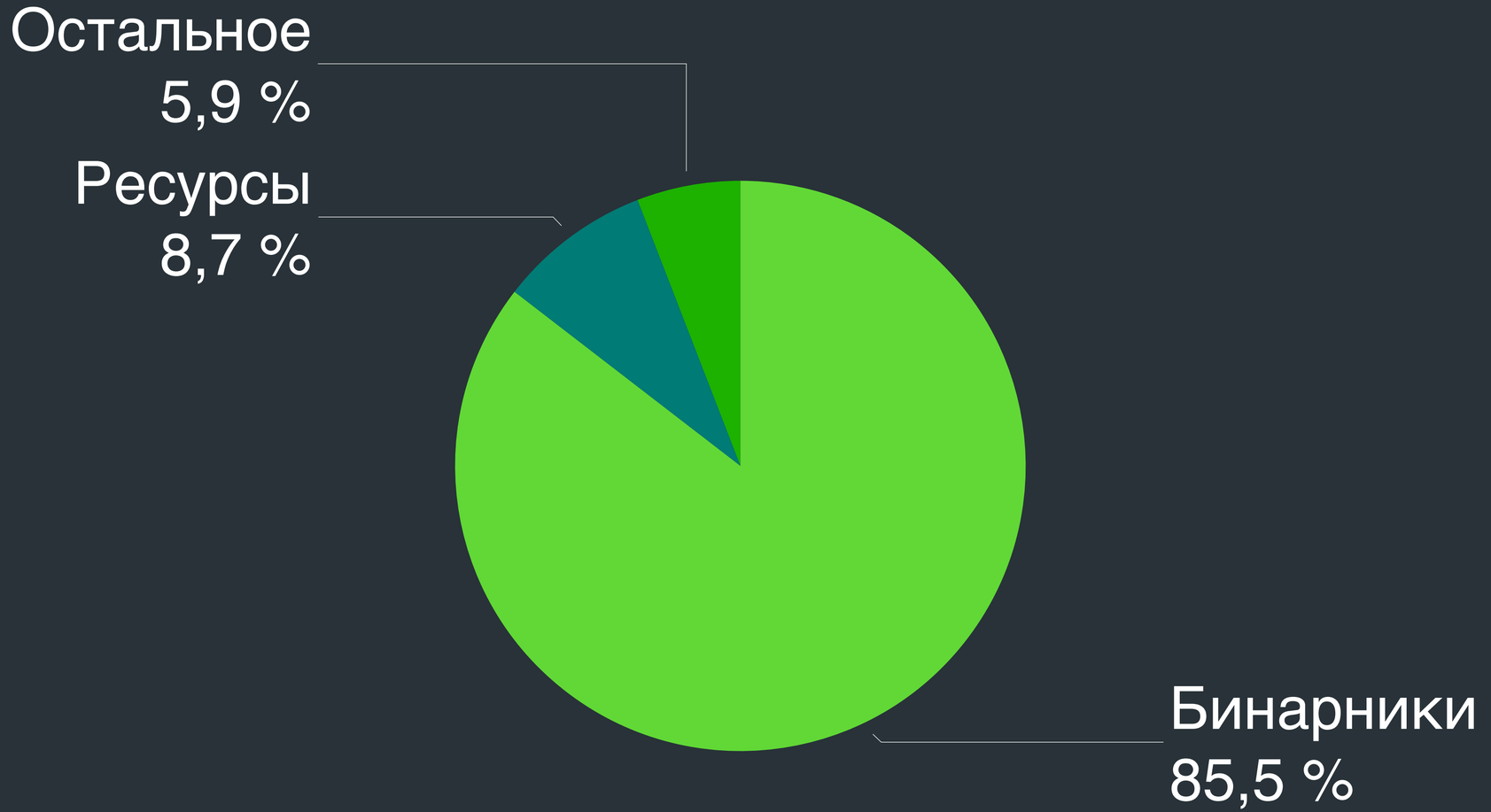
- Бинарники → dynamic library, static library, executable
- Ресурсы → car, mp3, caf, pdf, json, plist
- Остальное → xib, nib, storyboards, momd

Соотношение файлов в СберБанк



Первый замер, актуальная версия на апрель 2020 года

Соотношение файлов в СберБанк

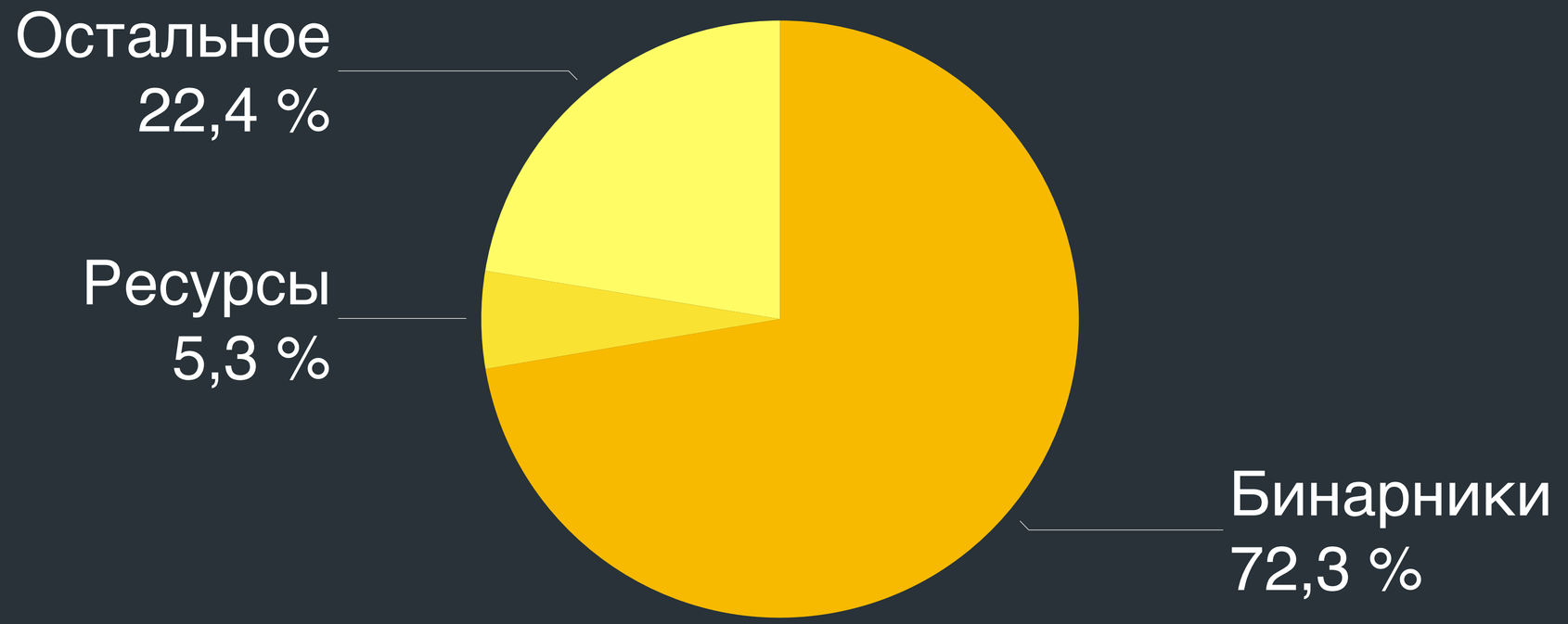


Последний замер, актуальная версия

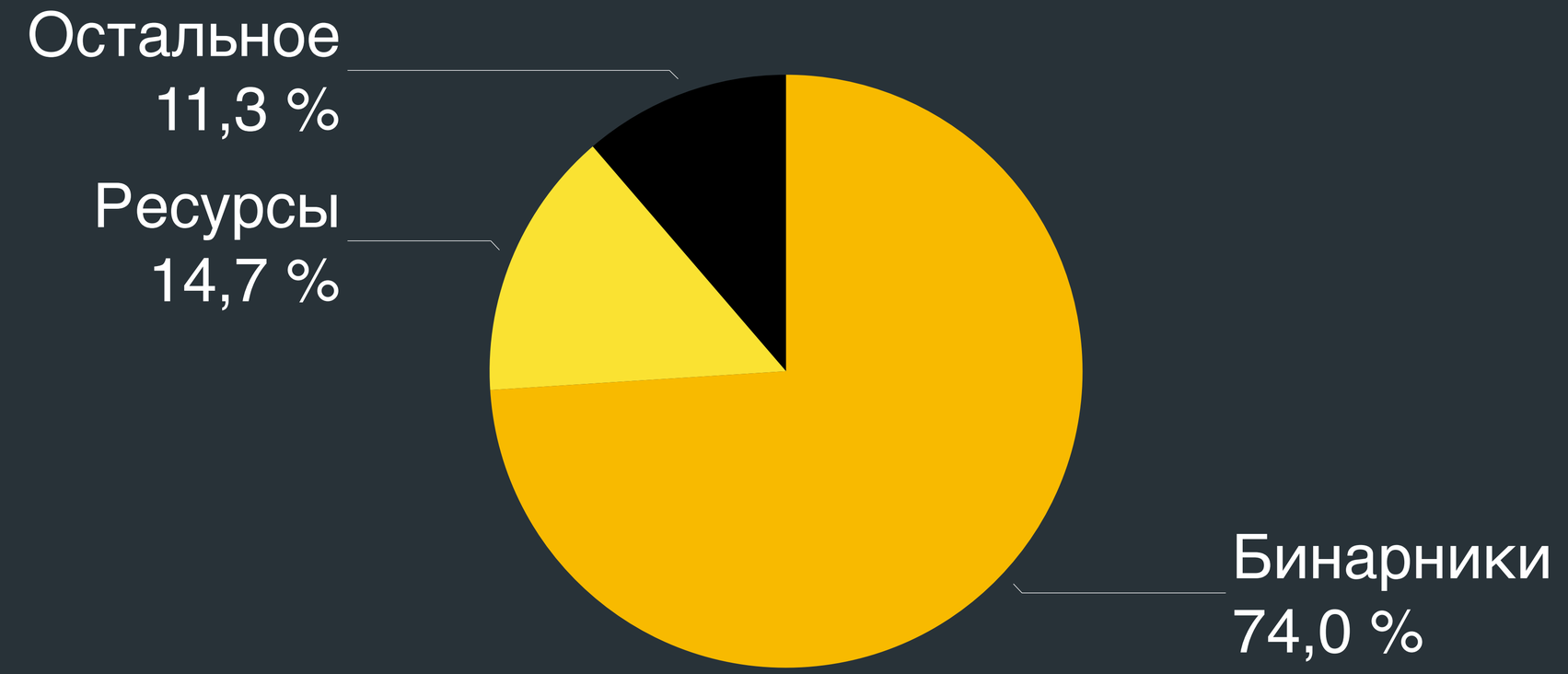
на март 2021 года

Что с другими?

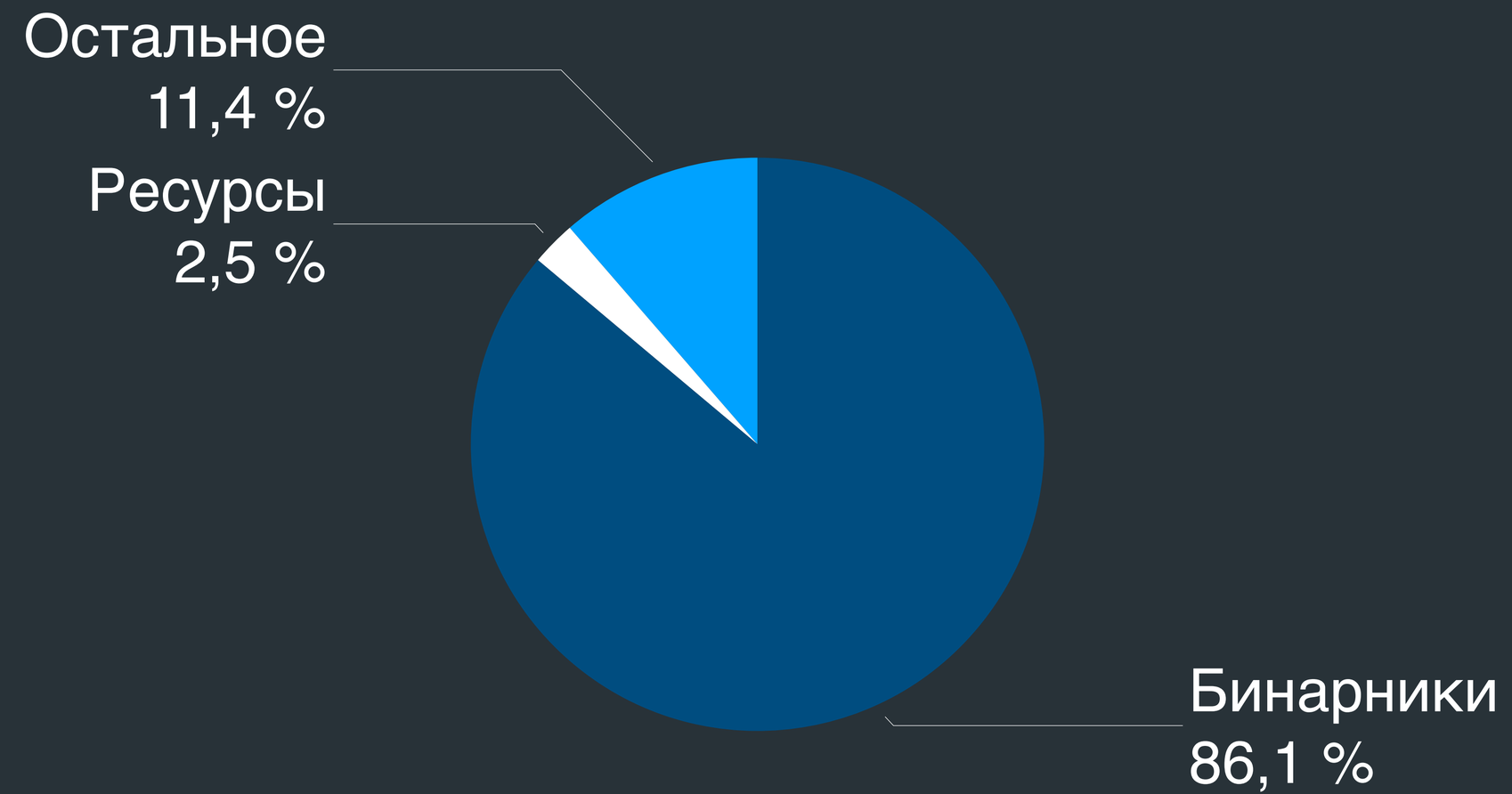
Соотношение файлов в банковском приложении А



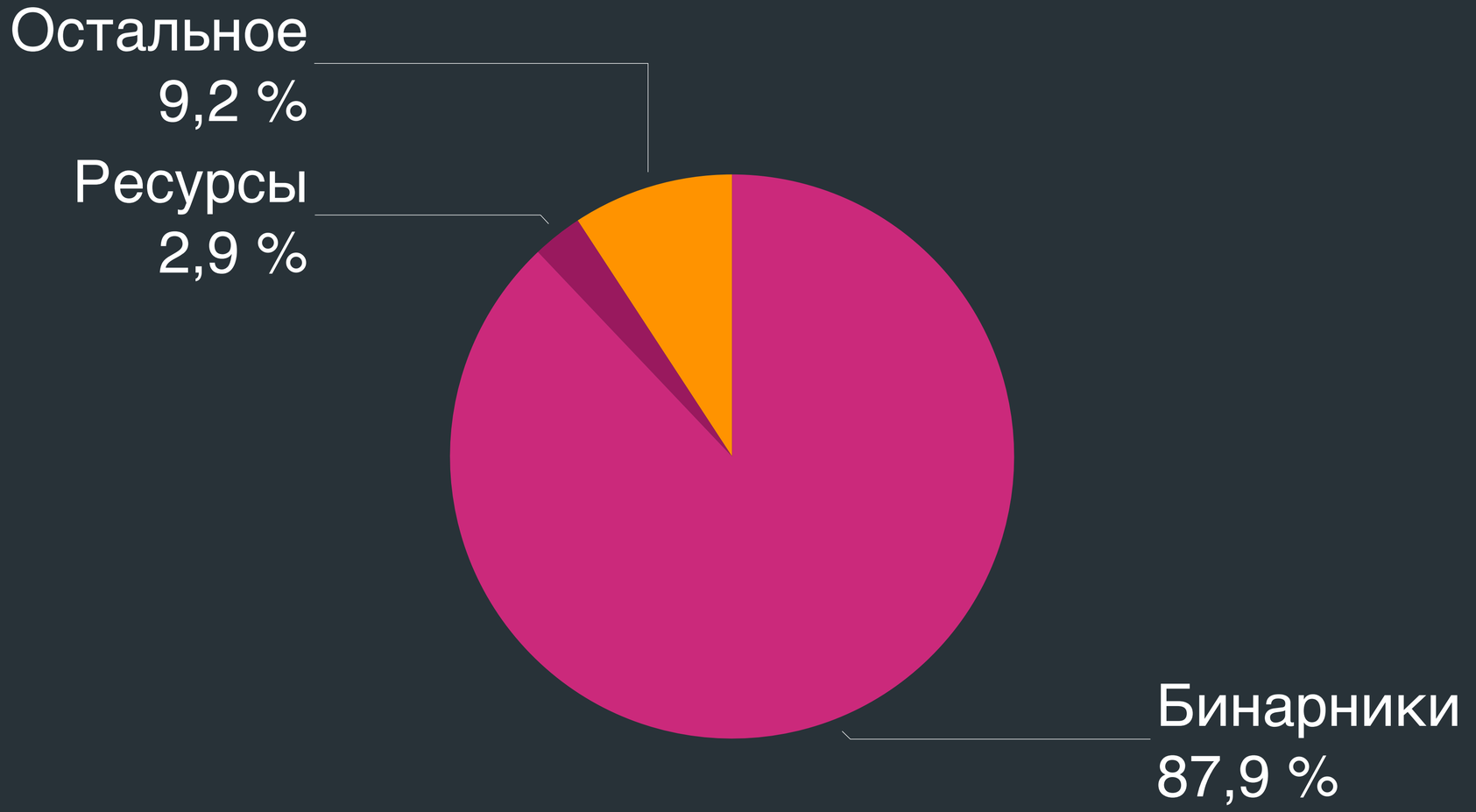
Соотношение файлов в банковском приложении Б



Соотношение файлов в соцсети А



Соотношение файлов в соцсети Б



Как можно оптимизировать размер?

Оптимизации размера, которые мы рассмотрим

- За счёт кода
- За счёт ресурсов

Оптимизация размера за счёт кода

- Язык
- Оптимизации компилятора
- Архитектура
- Mach-O type

Swift или Objective-C

- Optionals, value types, generics, codegen
- Swift runtime до iOS 12.2
- Protocol conformance
- Swift binary весит более чем в 2 раза больше

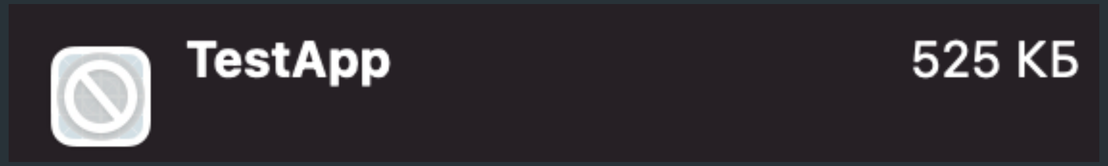
Сравнение размеров static framework

	Строки кода	Язык	Размер
Framework A	6564	Objective-C	276 KB
Framework B	7045	Swift	1346 KB
Framework C	4532	Swift	673 KB

Оптимизации компилятора

- Swift compiler optimization level: -Onone, -O, -Osize
- Apple clang optimization level: -O0 .. -O3, -Os, -Ofast, -Oz

optimization level -O



optimization level -Osize

Стоит индивидуально подходить к выбору уровня оптимизации для вашего приложения

Оптимизация размера за счёт ресурсов

- Реорганизация работы с ресурсами
- Оптимизация компилятора
- Удаления дубликатов
- On Demand Resources
- Сторонние зависимости

Реорганизация работы с ресурсами

- ~~.bundle~~
- .xcassets + NSDataAsset

Оптимизируем размер СберБанка

Как приступить к оптимизации

- Сбор данных о приложении
- Работы по оптимизации

Сбор данных о приложении

- Размер всего приложения
- Размер каждого таргета
- Размер ресурсов каждого таргета
- Количество строк кода в каждом таргете

Работы по оптимизации

- Замена legacy иконок на аналоги из DesignSystem
- Актуализация сторонних зависимостей
- Внедрение On Demand Resources
- Удаление мёртвого кода

СберБанк 11.1.0

218 Мб - 292 Мб

Как локально получить такой же размер как в AppStore

- Сборка в разных конфигурациях с bitcode и без
- Сборка с App Thinning
- Изучение AppStore Connect API

- Apple что-то скрывает и делает server side оптимизации

Что такое App Thinning

- xcodebuild … thinning <variant>
- Slicing
- On-Demand Resources
- Bitcode

App Thinning size report

- App Thinning Size report.txt
- app-thinning.plist

App Thinning Size Report for All Variants of Sberbank Beta Enterprise

Variant: SberbankApp-4C7355DD-1272-460A-9D7C-D9CDBB70565A.ipa

Supported variant descriptors: [device: iPhone9,3, os-version: 12.2], [device: iPhone9,1, os-version: 12.2], [device: iPhone10,1, os-version: 12.2], and [device: iPhone10,4, os-version: 12.2]

App + On Demand Resources size: 133,5 MB compressed, 341,5 MB uncompressed

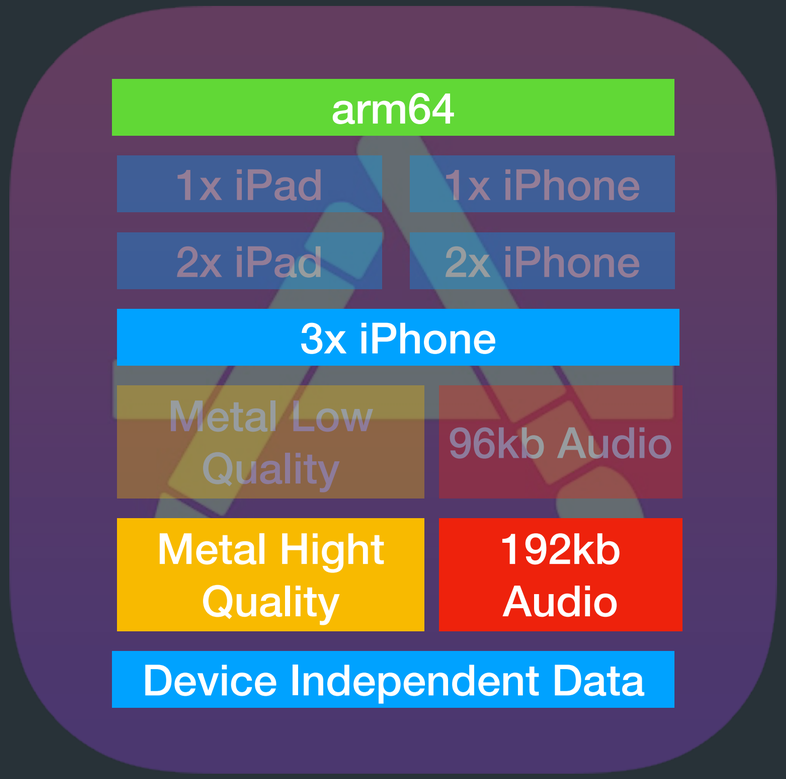
App size: 334,6 MB uncompressed

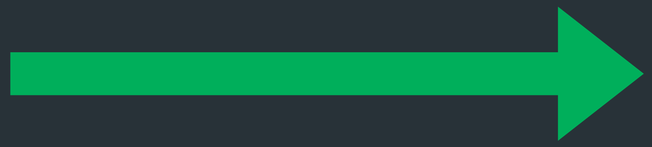
On Demand Resources size: 7 MB uncompressed

Slicing


arm64
1x iPad
1x iPhone
2x iPad
2x iPhone
3x iPhone
Metal Low Quality
96kb Audio
Metal Hight Quality
192kb Audio
Other data

Slicing


arm64
1x iPad
1x iPhone
2x iPad
2x iPhone
3x iPhone
Metal Low Quality
96kb Audio
Metal Hight Quality
192kb Audio
Device Independent Data

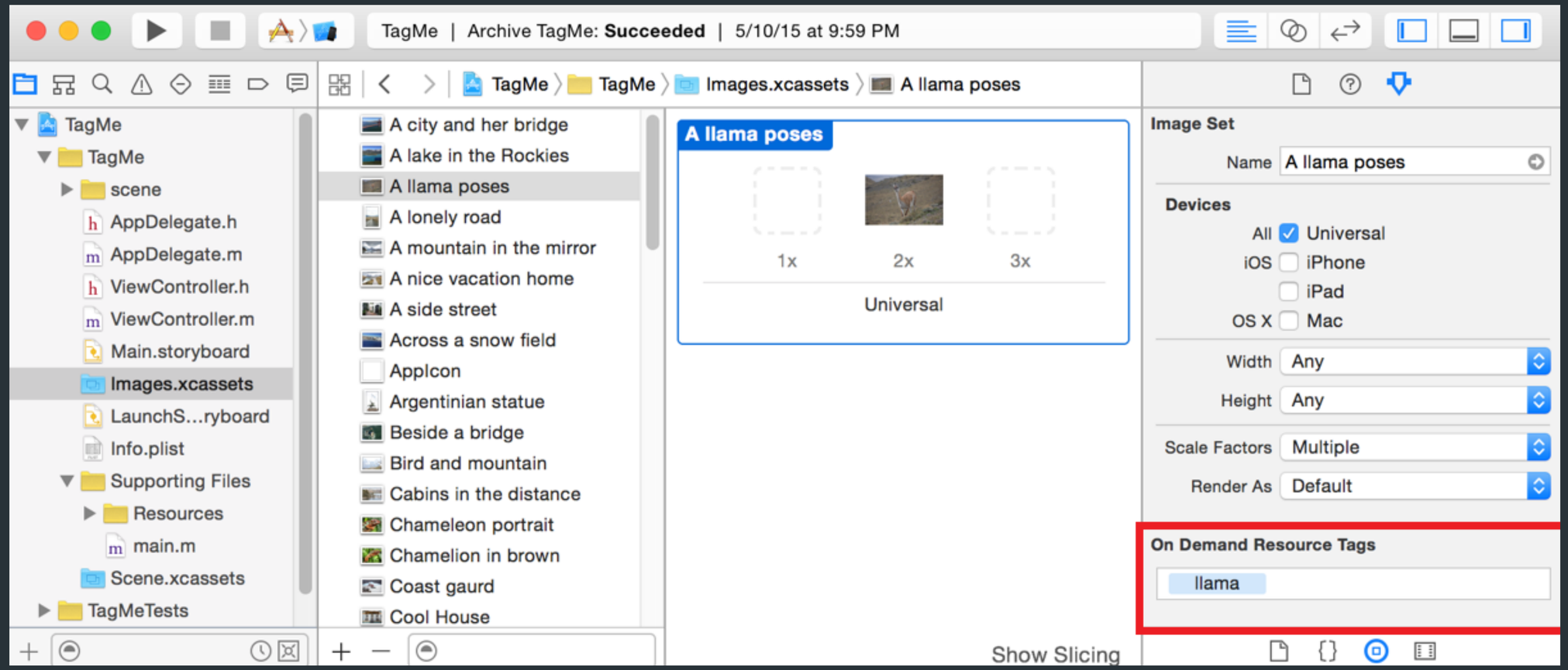



9:41 AM
100%
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Wallet
Notes
Reminders
Stocks
iTunes Store
App Store
iBooks
News
Health
Settings
Phone
Safari
Mail
Music

On-Demand Resources

- Ресурсы располагаются в облаке
- Разные варианты загрузки: Initial, Prefetched, On Demand
- Простой интерфейс

On-Demand Resources



Bitcode

- Позволяет перекомпилировать ваш бинарник
- Является прослойкой между LLVM backend и frontend
- Развязывает руки Apple и позволяет оптимизировать размеры вашего приложения

App Thinning - Итоги

- Можно собирать приложение под любое устройство
- Удобный отчет в формате .plist
- Приближенные размеры к реальным
- Apple говорит, что так вы можете измерить размер вашего приложения

Удаление legacy иконок

- LSUnusedResources - приложение для поиска неиспользуемых ресурсов
- Ручное удаление дубликатов
- Сложности в коммуникациях с feature-командами
- Неопределенность с баннерами и крупными картинками

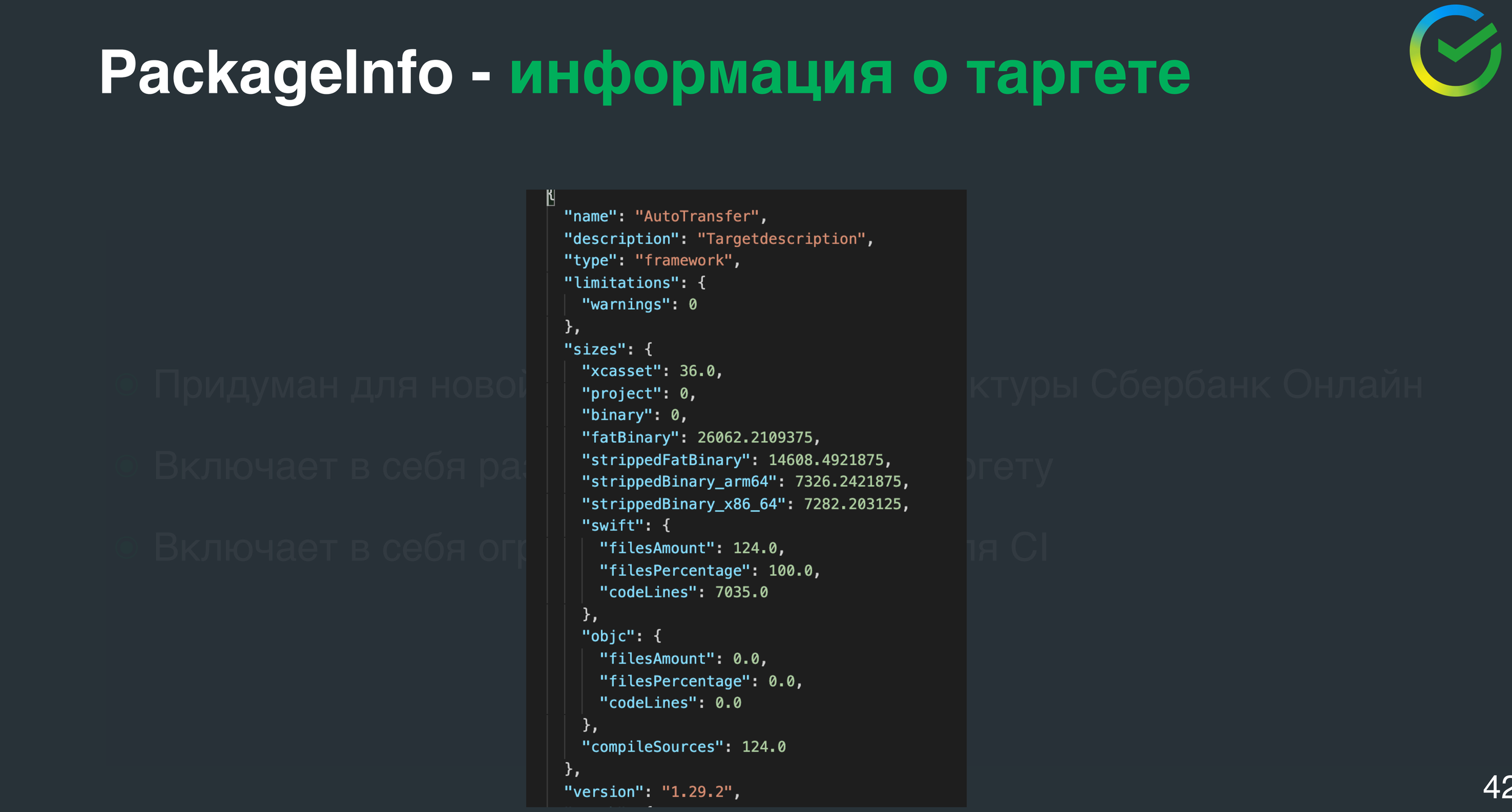
Удаление legacy иконок: CI

- Проверка на дубликаты по названию
- ~~Проверка на дубликаты по содержанию (VIPS)~~
- Пересчет размеров .xcassets для таргета
- Проверка на ограничение для текущего таргета
- Code review от core команды

PackageInfo - информация о таргете

- Придуман для новой межмодульной архитектуры Сбербанк Онлайн
- Включает в себя различные метрики по таргету
- Включает в себя ограничения по таргету для CI

PackageInfo - информация о таргете



Размеры для PackageInfo

- Сборка fat binary таргета
- Strip bitcode
- Разделение по архитектурам
- Подсчёт строк кода с помощью cloc
- Подсчёт размеров xcassets

Размеры для PackageInfo

- xcrun bitcode_strip
- lipo -thin <arch>

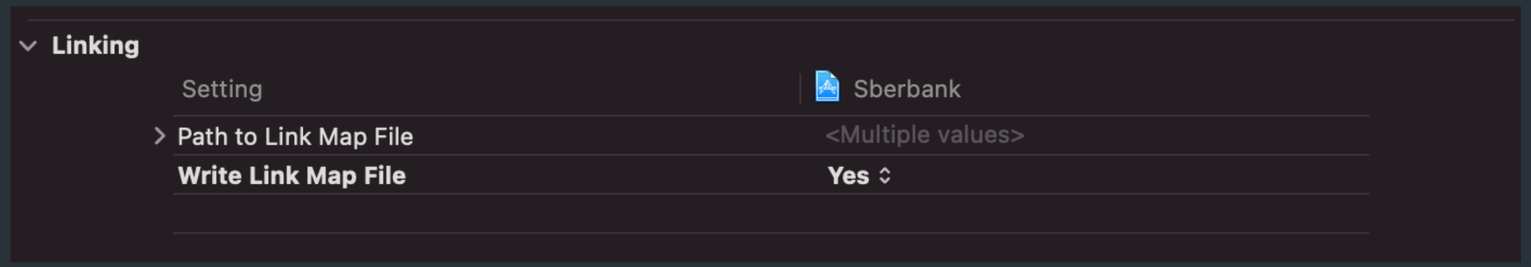
```ruby
def json_report
  Dir["**/#{@target_name}.framework"].each do |path|
    next unless path.include?('Sberthage/Build')

    @target_path = File.expand_path(path)
  end
  raise "Can't find such module #{target_name}" if @target_path.empty?

  hash = {
    'targetName' => @target_name,
    'targetVersion' => target_version,
    'size' => { 'fatBinary' => fat_binary_with_bc_size,
                'strippedFatBinary' => stripped_bc_fat_binary_size,
                'strippedBinary_arm64' => stripped_bc_for_arch('arm64'),
                'strippedBinary_x86_64' => stripped_bc_for_arch('x86_64') }
  }
  clean_artifacts
  hash
end
```

Для static framework размеры включают в себя символы от других framework, залинкованных в анализируемый

Определение абсолютных размеров таргетов



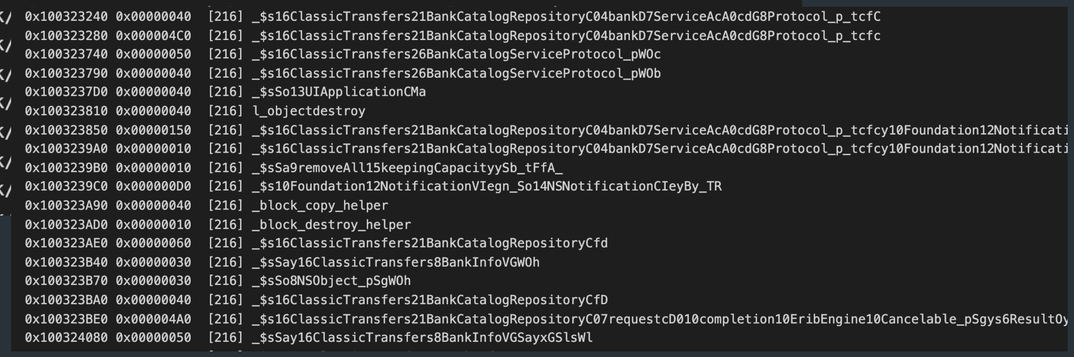
LD_GENERATE_MAP_FILE = YES

$(TARGET_TEMP_DIR)/$(PRODUCT_NAME)-LinkMap-$(CURRENT_VARIANT)-$(CURRENT_ARCH).txt

SberbankApp-LinkMap-normal-x86_64.txt

Link Map file

Object files



Sections


```
"name": "ClassicTransfers",
"size": 5884.333984375,
"files": [
  {
    "name": "InitialAccountTransferPresenter.o",
    "size": 124.1806640625
  },
  {
    "name": "RequisitesAccountTransferViewController.ViewInteractor",
    "size": 110.2265625
  },
  {
    "name": "SaveService.o",
    "size": 93.8466796875
  },
  {
    "name": "MainForkModuleProtocol.o",
    "size": 92.64453125
  },
```

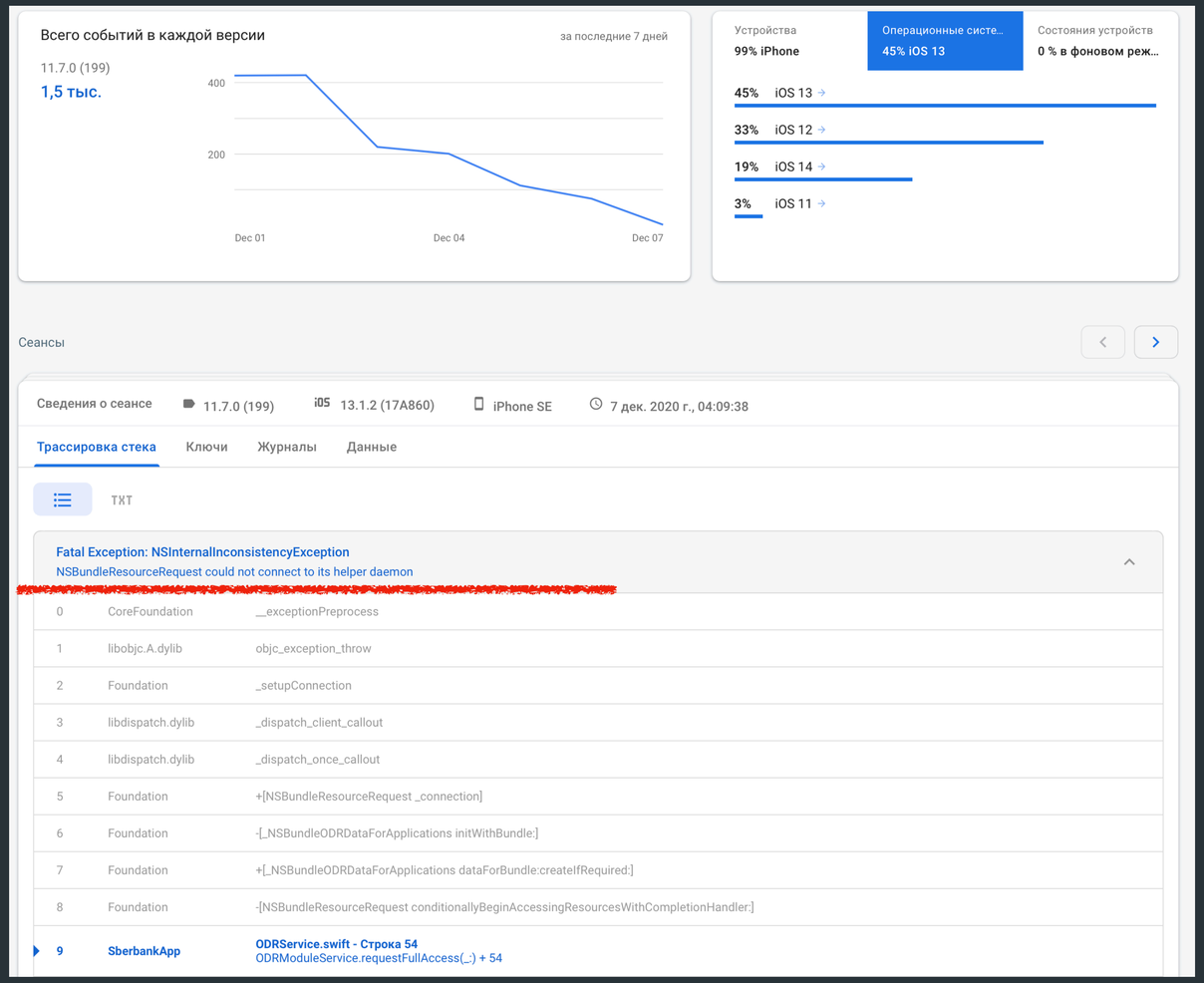
Подсчёт размеров таргетов

- Размеры бинарного файла могут не соответствовать реальным размерам
- Парсинг Linkmap file является универсальным способом для подсчета размеров всех зависимостей

Первая попытка внедрить On-Demand Resources

- 2 xcassets и анимации стикер пака
- Использовали Prefetched tag
- Загрузка данных осуществлялась на старте приложения

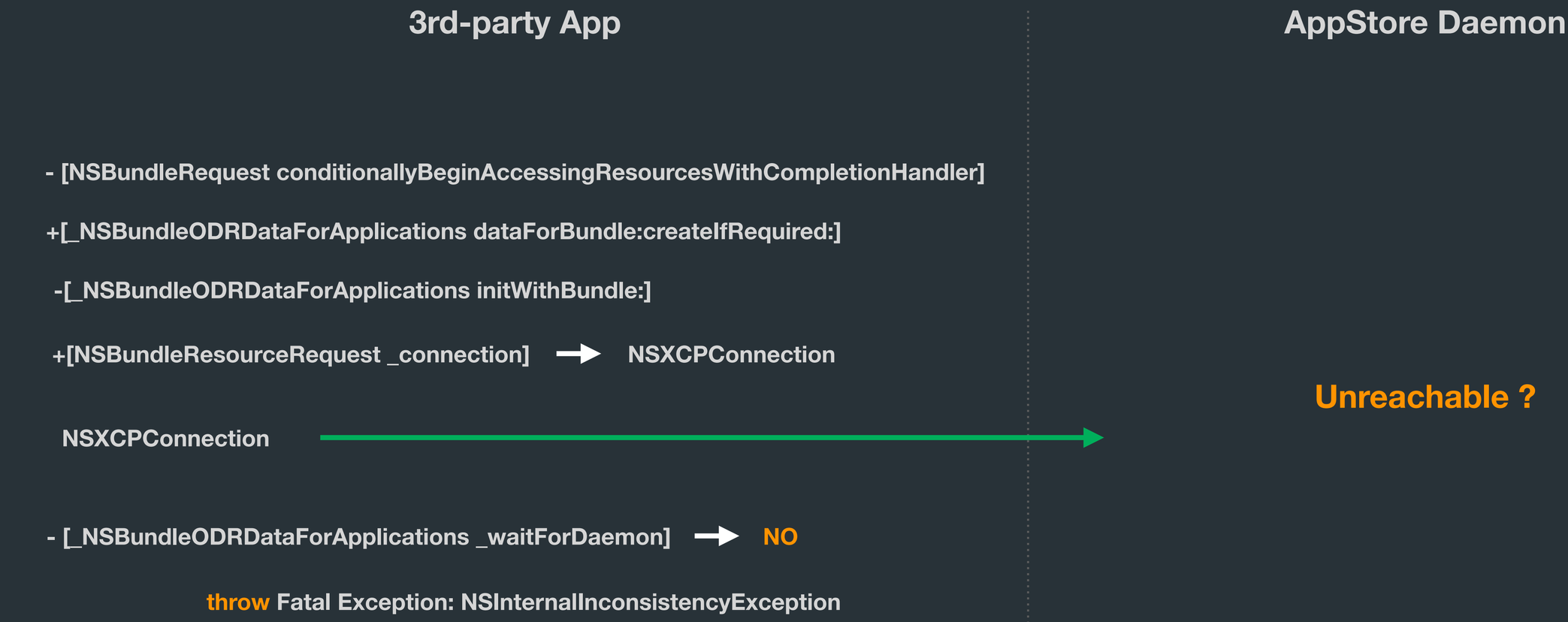
Первая попытка внедрить On-Demand Resources



Первая попытка внедрить On-Demand Resources

- 0 крашей за месяц на Firebase distribution
- 2 тикета на open radar’e
- Никаких легальных workaround’ов
- Открытый тикет с января на feedback asistant
- Потраченный DTS

Как On-Demand Resources крашит клиентское приложение



Первая попытка внедрить On-Demand Resources - Итоги

- Hotfix
- Обратно вернули 6 мб в bundle приложения

Вторая попытка внедрить On-Demand Resources

- Второй раз на те же грабли?
- Деморежим
- On demand only tag
- Загрузка данных при переходе в данный раздел

Вторая попытка внедрить On-Demand Resources

```
ublic func conditionallyAccess(_ completion: ((Bool) -> Void)?) {
    privateQueue.async {
        guard !self.accessed else {
            self.returnQueue.async {
                completion?(true)
            }
            return
        }
        self.request?.conditionallyBeginAccessingResources { [weak sel
            guard let self = self else {
                completion?(result)
                return
            }
            self.accessed = result
            self.returnQueue.async {
                completion?(result)
            }
        }
    }
```

```
public func downloadAndAccess(_ completion: ((Error?) -> Void)?) {
    privateQueue.async {
        self.downloadClosure = completion
        guard !self.accessed else {
            self.returnQueue.async {
                completion?(nil)
            }
            return
        }
        self.timer?.resume()
        self.request?.beginAccessingResources { [weak self] error in
            guard let self = self else {
                completion?(error)
                return
            }
            self.timer?.suspend()
            guard let error = error else {
                self.accessed = true
                self.returnQueue.async {
                    completion?(nil)
                }
                return
            }
            self.accessed = false
            self.returnQueue.async {
                completion?(error)
            }
        }
    }
}
```

Тонкости, которые стоит учесть при внедрении On-Demand Resources

- Нет реализации timeout с коробки
- Нужна синхронизация
- Нужно пересоздавать NSBundleResourceRequest для новый запросов
- Нужно предусмотреть обработку ошибок (NSBundleOnDemandResourceOutOfSpaceError)

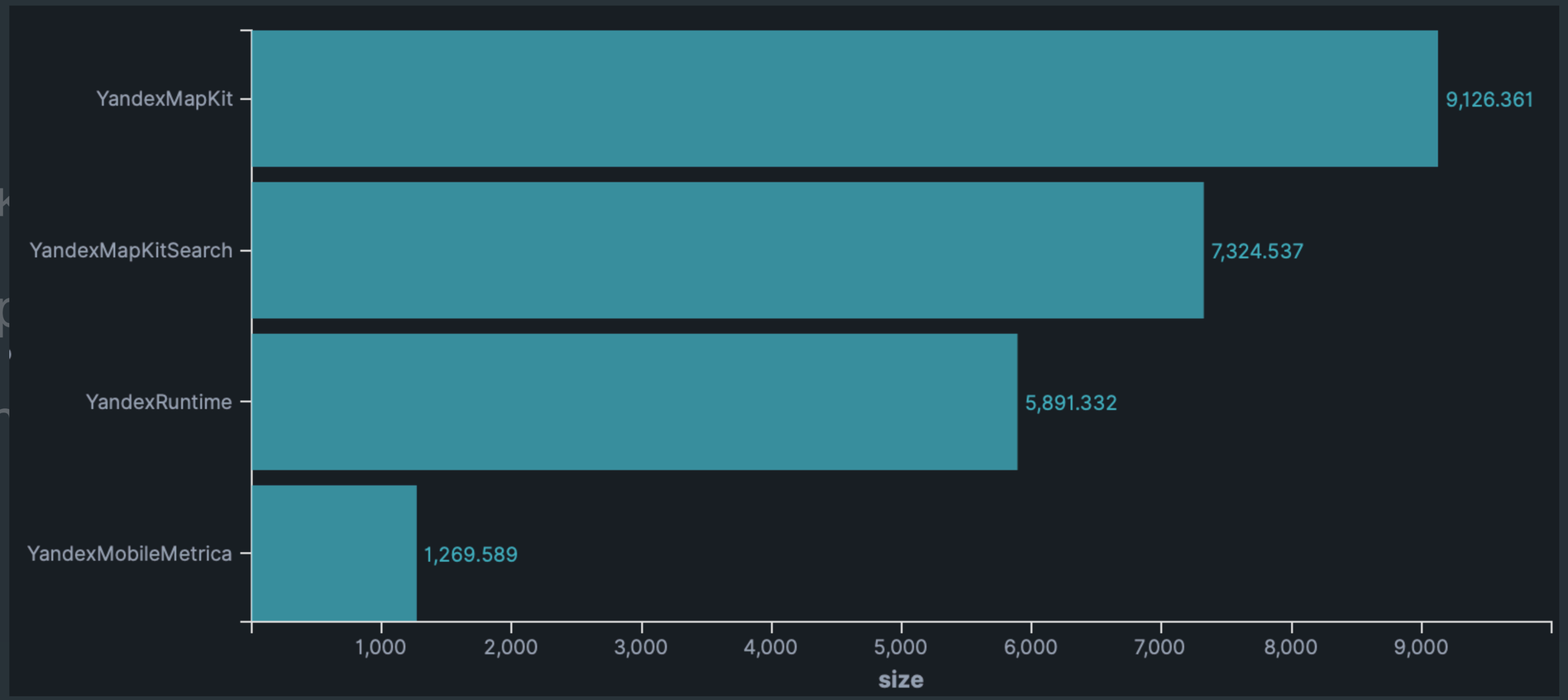
Вторая попытка внедрить On-Demand Resources - Итоги

- ~~Couldn’t connect to helper Daemon~~
- Пару крашей вначале из-за повторных вызовов NSBundleResourceRequest
- Сэкономили 7 Мб

Оптимизация сторонних зависимостей

- Отказались от многих спорных зависимостей
- Перекомпиляция зависимостей под Mach-O static
- Yandex

Оптимизация сторонних зависимостей

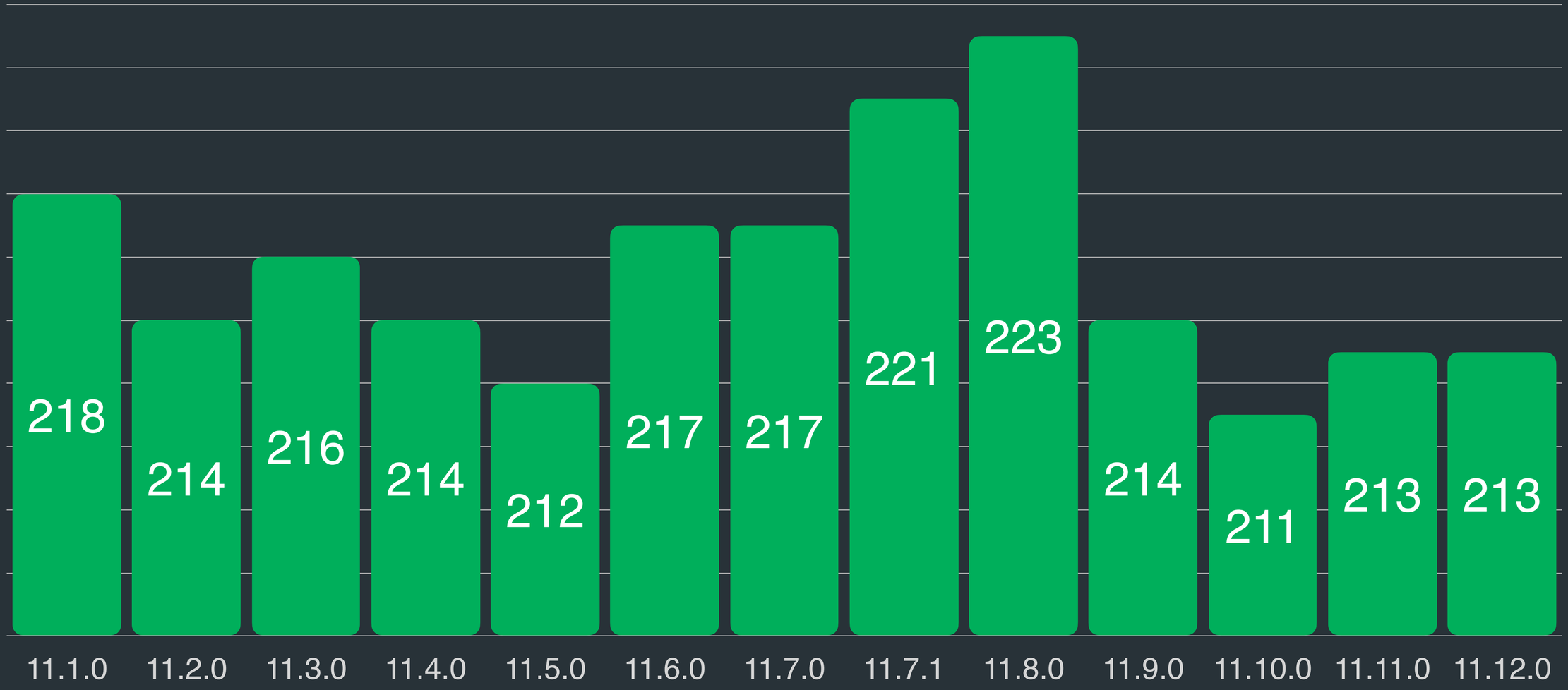


YandexMapKit + YandexMobileMetrica = 23 Мб

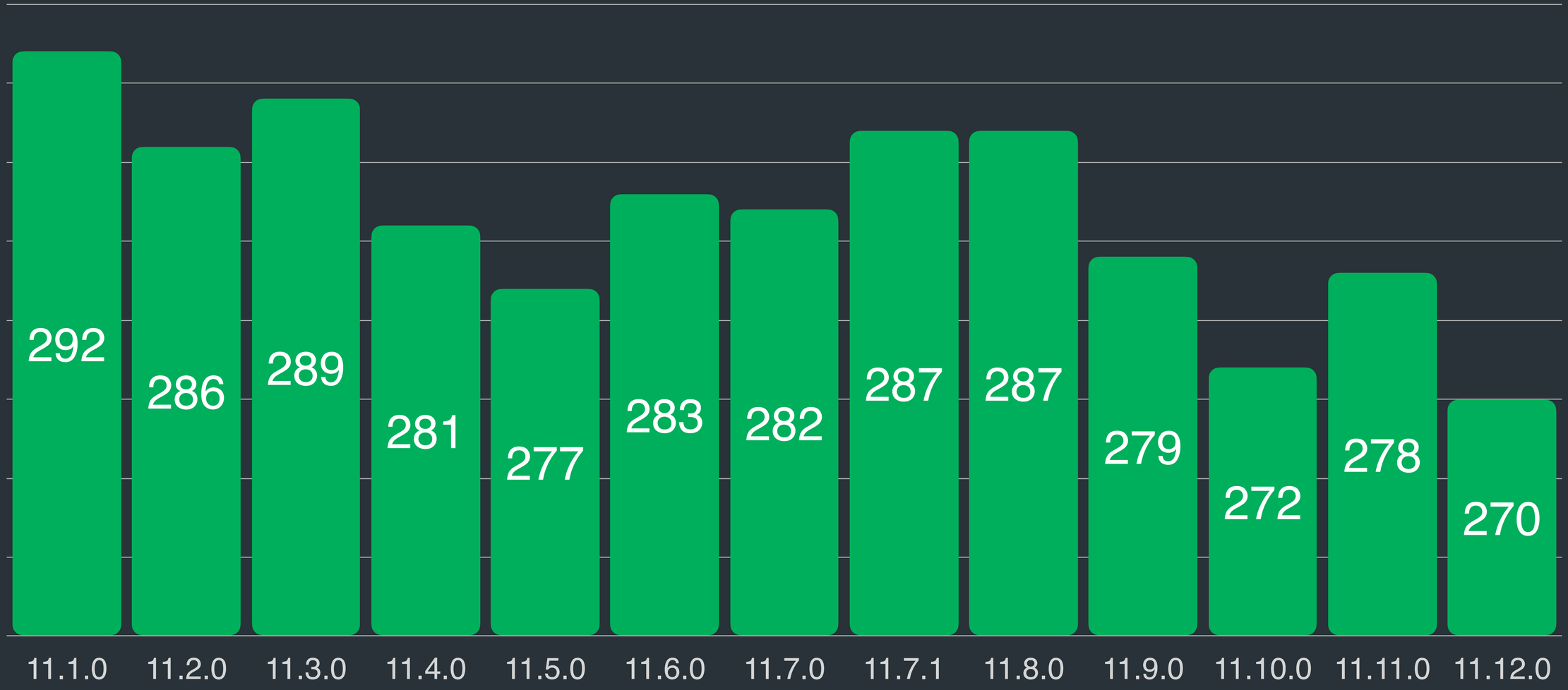
СберБанк 11.12.0

213 Мб - 270 Мб

Размер при скачивании СберБанка
для iPhone X



Размер при установки СберБанка
для iPhone X



Итоги

- Сокращение в чистом виде на 3%
- Сокращения размера приложения на 15% с учетом среднего ежерелизного прироста
- Рост приложения стал контролируемым
- Рост размера - неизбежен, когда увеличивается функциональность приложения
- Оптимизация размеров - это бесконечный процесс, а не разовое мероприятие

Чтобы уменьшить размер

- Удалить мёртвый код
- Удалить дублирующиеся ресурсы
- Удалить неиспользуемые ресурсы
- Провалидировать сторонние зависимости
- Воспользоваться ODR
- Избегать повсеместного использования optional, generic, struct
- Конформить только нужным протоколам

Что делать после?

Из активной стадии в стадию поддержания

- Запланировать работы по удалению мёртвого кода и ресурсов с какой-то периодичностью
- Пользоваться метриками, если вы их начали собирать
- Оптимизация компиляции Swift кода
- Оптимизация размеров container/data вашего приложения

Полезные ссылки

Спасибо за внимание