USETECH

iOS Background Modes. Применяем и укрощаем на практике

Анна Жаркова
Lead Mobile developer

О себе

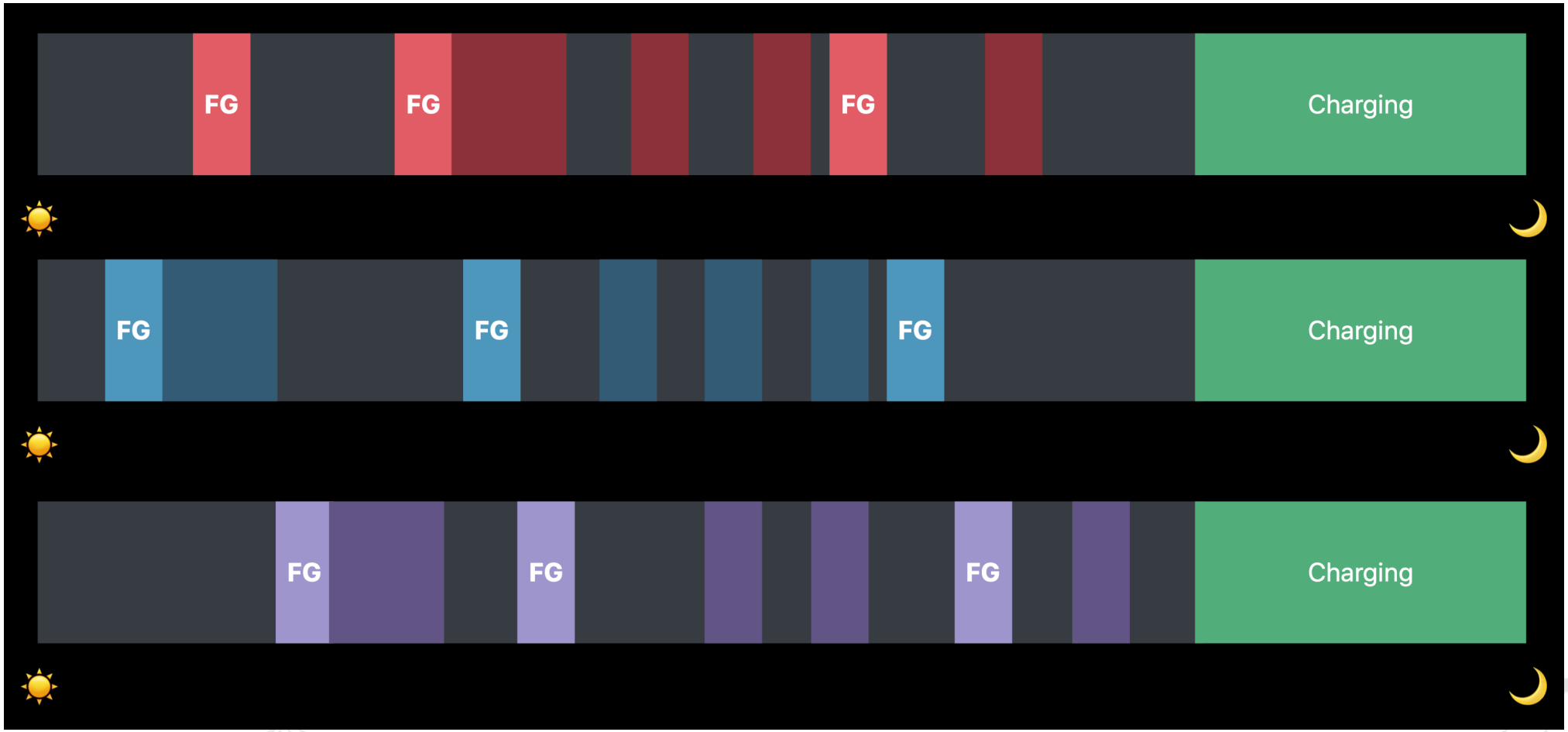
- Опыт в коммерческой разработке ПО более 6 лет
- Ведущий мобильный разработчик компании «Usetech».
- Разработка iOS и Android приложений нативных (Swift/Objective-C, Kotlin/Java) и кроссплатформенных (Xamarin, Kotlin multiplatform)
- Разработка архитектуры мобильных приложений для обеих платформ
- Управление командой направления.
- Веду младших разработчиков (менторство).
- Пишу статьи, выступаю на митапах

Goals. Сегодня рассмотрим:

- iOS Background Mode
- Концепция энергосбережения Apple. Ограничения
- Рекомендуемые решения
- Случаи, когда стандартные решения не подходят. Способы обхода по правилам и нет
- Новые решения в iOS 13/iOS 14. Помогут ли в нестандартных случаях

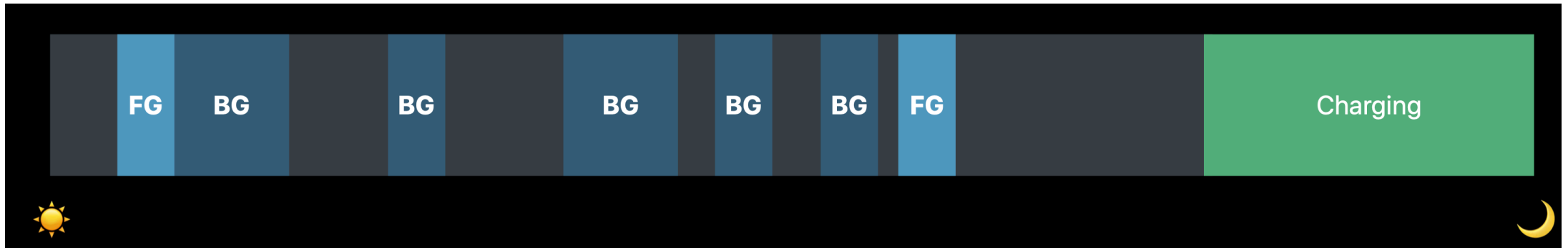
Концепция энергосбережения Apple

Концепция энергосбережения Apple



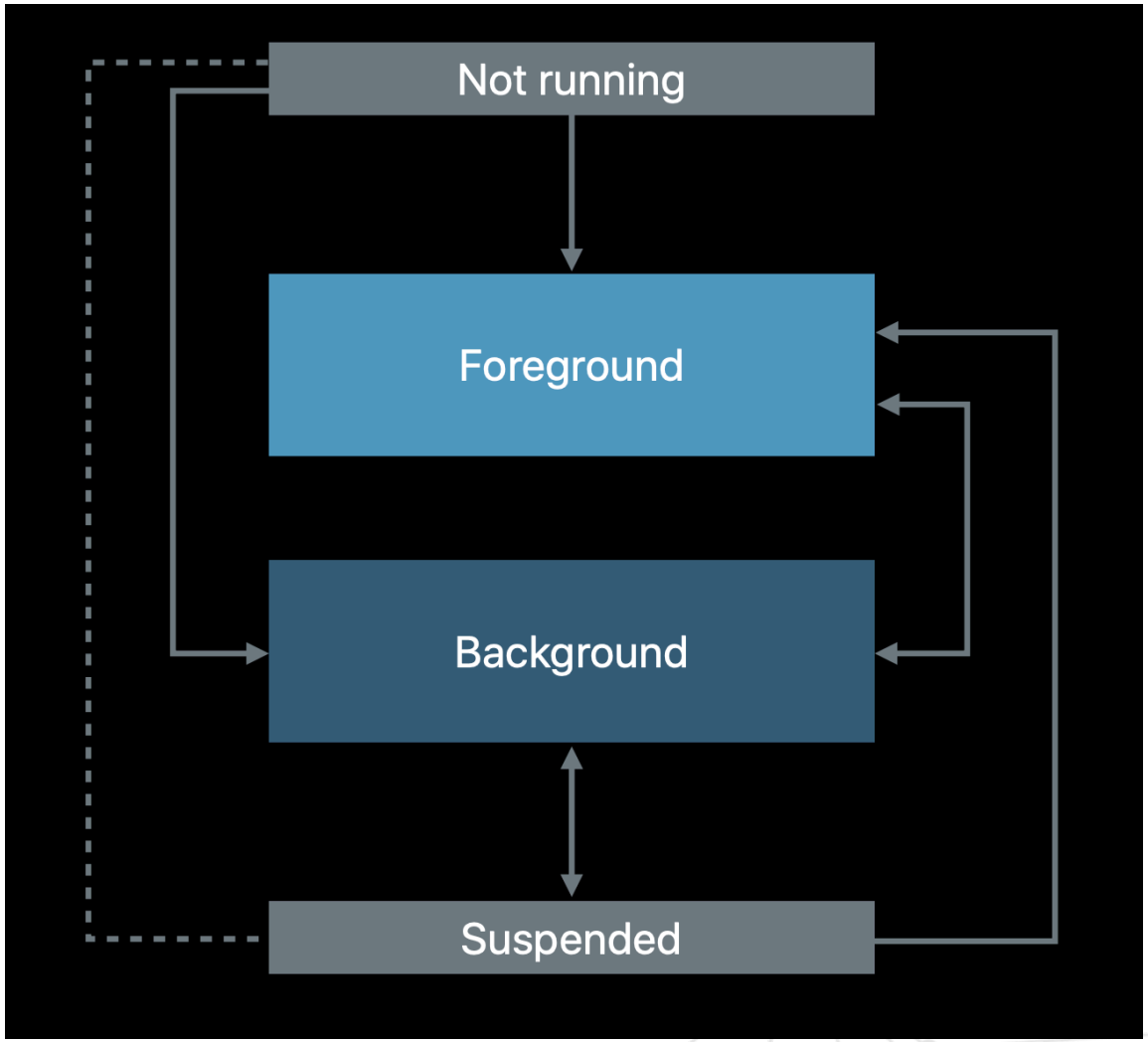
Концепция энергосбережения Apple

Концепция энергосбережения Apple

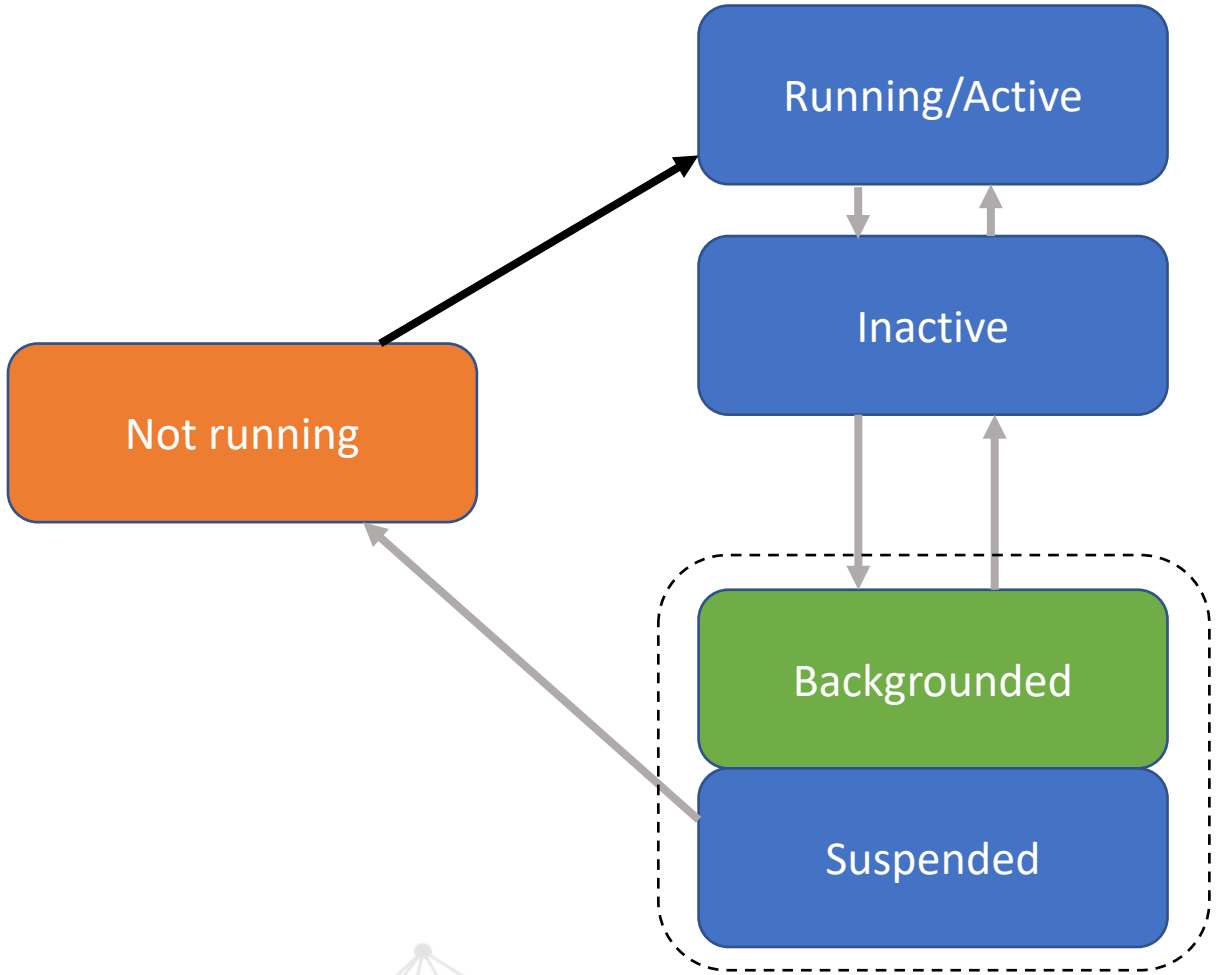


Background mode

Фоновый режим – инструмент поддержки работы приложения в неактивном состоянии

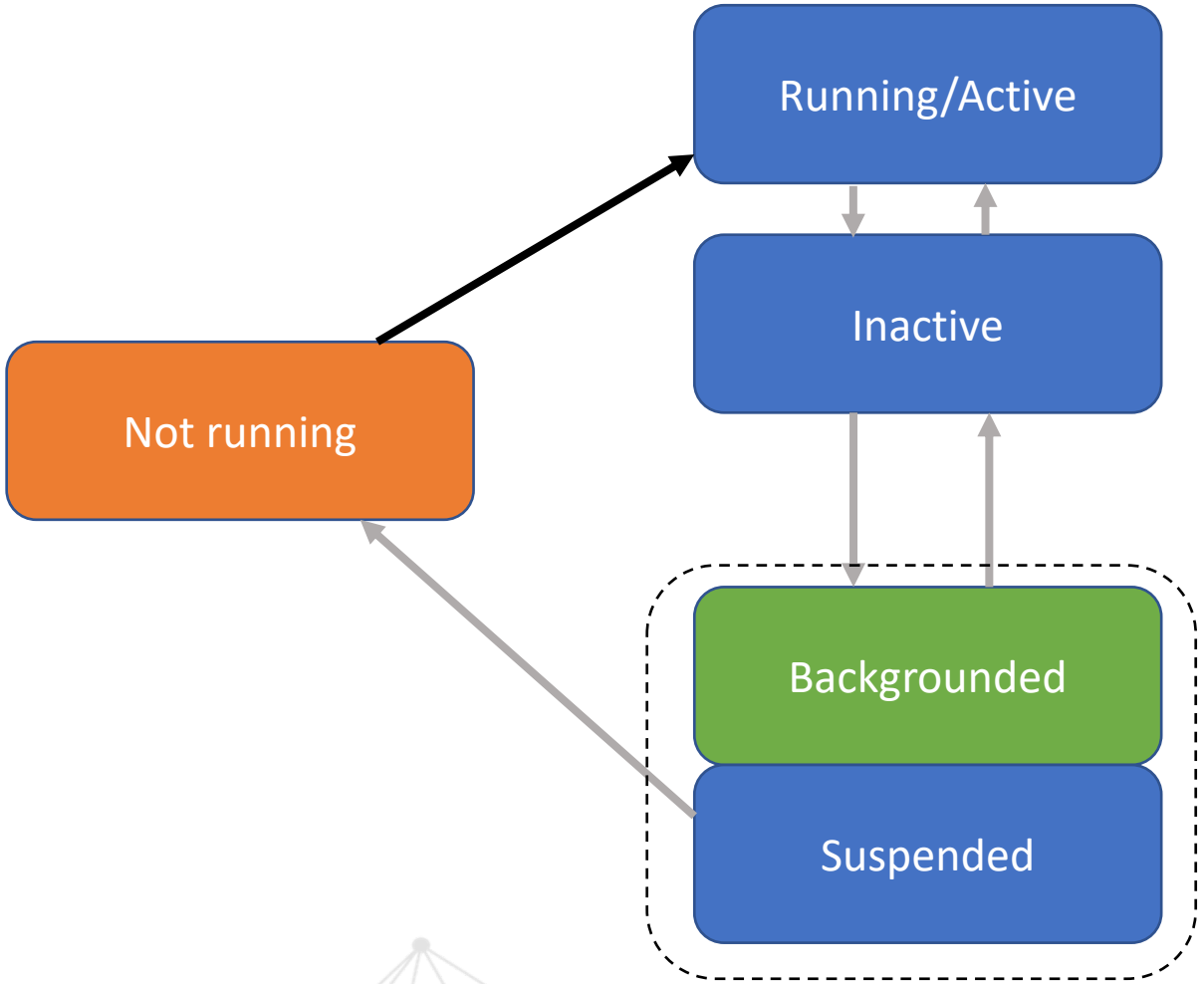


Состояния и цикл жизни приложения



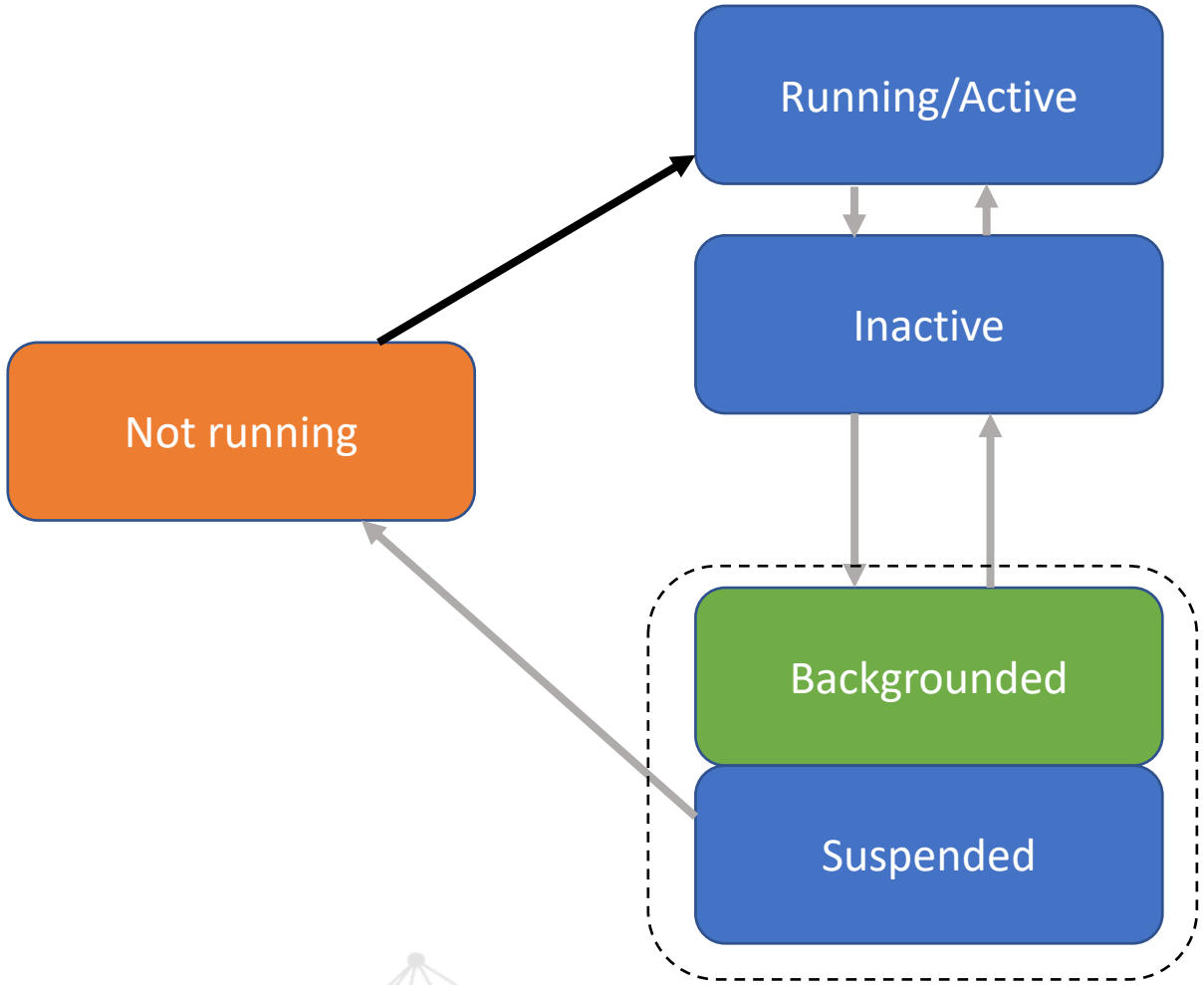

```swift
func applicationWillResignActive(_ application: UIApplication) {
    //Подготовка к уходу в фон
}

func applicationDidEnterBackground(_ application: UIApplication) {
    //Уход в фон
}

func applicationWillEnterForeground(_ application: UIApplication) {
    //Старт приложения, возвращение из Background
}

func applicationDidBecomeActive(_ application: UIApplication) {
    //Старт приложения, возвращение из Background
}
```

Состояния и цикл жизни приложения



Backgrounded приложение перемещается в фоновый режим, продолжается на выполнение фонового кода

Состояния и цикл жизни приложения



Если кода для выполнения в фоне нет, или его выполнить нельзя, приложение приостанавливается (suspended) системой

Задача: найти триггер

Background mode

Android  Service, Intent service, Foreground service

iOS  ???

Назначение Background modes

- Сохранить состояние и остановить процессы (работу с UI/сенсорами, запросы) за < 30 секунд
- Запустить фоновую задачу с разрешенным функционалом (время: 30 секунд или длительность исполнения задачи)

Расширение: UIBackgroundTask

```
// Запрос задачи с ID
self.backgroundTaskID = UIApplication.shared.
            beginBackgroundTask (withName: "Some Tasks") {
   //Обработка окончания задачи
   UIApplication.shared.endBackgroundTask(self.backgroundTaskID!)
   self.backgroundTaskID = UIBackgroundTaskInvalid
}

//Долгий запрос
self.requestLong()

// Завершение задачи
UIApplication.shared.endBackgroundTask(self.backgroundTaskID!)
self.backgroundTaskID = UIBackgroundTaskInvalid
```

Background modes

Calls	Finish Foreground Work	Music	Maintenance
Downloading and Uploading Files		Siri Integration	MFi Accessory
Bluetooth	Periodic Content Updates		Connecting to Hotspots
Step Counting	Watch App	New Server Data	Navigation

Виды UIBackground Modes

- Аудио и AirPlay
- Обновления информации о местоположении (Location updates)
- Речь по IP (Voice over IP)
- Внешняя вспомогательная коммуникация (External accessory communication)
- BLE
- Фоновая выборка (Background fetch) и обработка (Background processing)
- Удаленные уведомления (Remote Notifications)

Background Modes и цикл жизни приложения

Если приложение реализует выполнение в фоне, то не вызывается метод **applicationWillTerminate(_:)**

Вызов произойдет только при необходимости освободить ресурсы памяти

```
func applicationDidReceiveMemoryWarning(_ application: UIApplication) {
    print("warning \(Date().formatted)")
}

func applicationWillTerminate(_ application: UIApplication) {
    print("terminated \(Date().formatted)")
}
```

Цель ограничений Background mode (Apple)

- Сохранение заряда батареи для работы устройства и других приложений
- Перформанс работы устройства и других приложений
- Безопасность пользователя и его данных от скрытых действий приложений

Идея Background mode (Apple)

Поиск баланса между предоставлением приложениям ресурсов для выполнения фоновых задач и сохранения скорости реагирования приложения и устройства

Мощность
Производительность
Конфиденциальность

Актуальность
- состояния
- контента

Надежность

Рекомендуемые решения с Background mode

- Максимально перенести логику в форграунд
- Сократить работу в фоне
- Использовать отсроченное выполнение задач

Посмотрим теперь стандартный кейс

Приложение с чатом на XMPP

Ранее XMPPFramework позволял работать с jabber в фоне для получения сообщений.

Теперь:

Рекомендация использовать PushKit и VoIP

Требует также доработки со стороны сервера. Есть ограничения от заказчика

Приложение с чатом на XMPP

Решение:

Сервис обращается к Jabber только в форграунде. При уходе в фон сервис отсоединяется от Jabber

```
func applicationDidEnterBackground(_ application: UIApplication) {
        XmppService.shared.stop()
}

func applicationWillEnterForeground(_ application: UIApplication) {
        XmppService.shared.loadAndConnect()
}
```

Приложение с чатом на XMPP

Решение:

Уведомление о новых сообщениях приходит в виде push-notifications

```
{
//...
content-available: 1

apns-priority: 5

apns-push-type: background

//...
}
```

Разрешение **Remote notifications** UIBackground mode

- apns-priority = 5 (c iOS 13)

- apns-push-type = background (c iOS 13)

Приложение с чатом на XMPP

Решение:

1. Сервис обращается к Jabber только в форграунде. При уходе в фон сервис отсоединяется от Jabber

2. Запрос сообщений для комнаты идет также из форграунда

3. Сообщения логируются на бекенде (требования заказчика) . История запрашивается у бекенда

4. Уведомление о новых сообщениях приходит в виде push-notifications

Summary. Рекомендуемые решения с Background modes

- Максимально перенести логику в форграунд

- Сократить работу в фоне:
  - фоновые уведомления

- Использовать отсроченное выполнение задач (сетевые запросы, шедуллинг)

Теперь самое интересное и сложное

Сложные кейсы с Background Mode на практике:

- Приложение с трекингом перемещения
- Отслеживание (считывание) параметров аппарата
- Работа с внешним устройством через BLE

Сложные кейсы с Background Mode на практике:

- Длительная или периодическая работа в фоне
- Старт работы в фоне по сигналу и/или в заданное время
- Доступ к сенсорам из фона

Приложение с трекингом перемещения

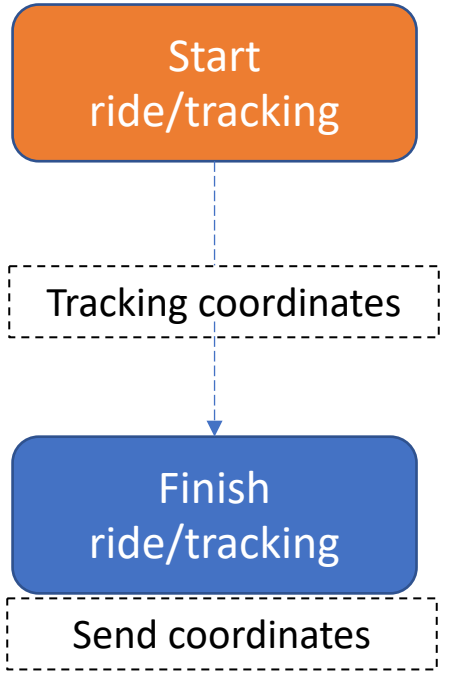
Приложение с трекингом перемещения

Приложение велопроката

Автоматический трекинг перемещения пользователя по маршруту от начала и до конца поездки.

Background/Foreground events:

- старт/конец поездки
- сбор координат (Location Updates)
- отправка координат (networking)

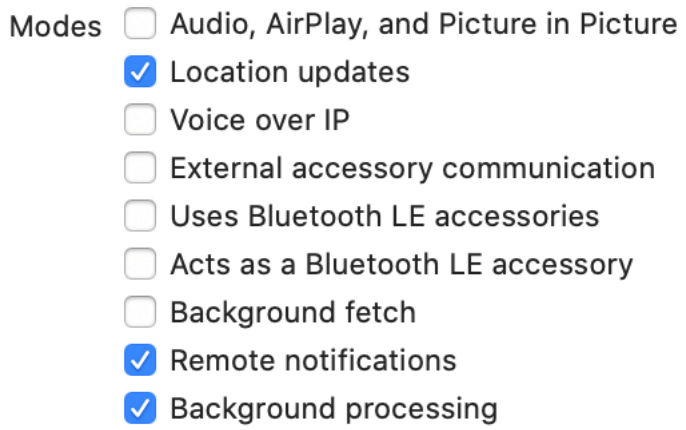


Приложение с трекингом перемещения

Приложение велопроката

Исходные данные:

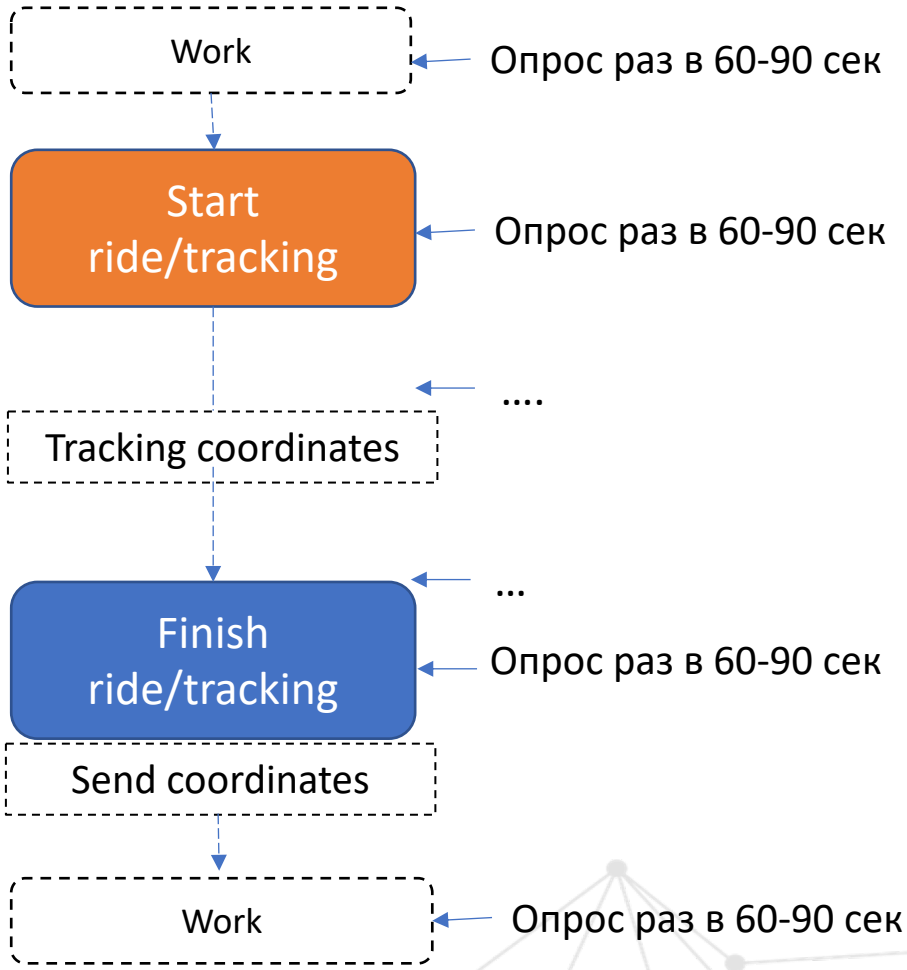
- Location updates
- Background processing
- Remote notifications (с 2018 года)



Приложение с трекингом перемещения

<u>Первая реализация (2017 год, iOS 9 – iOS 10).</u>
В iOS фоновый режим сокращается с ~10 минут до 180 секунд

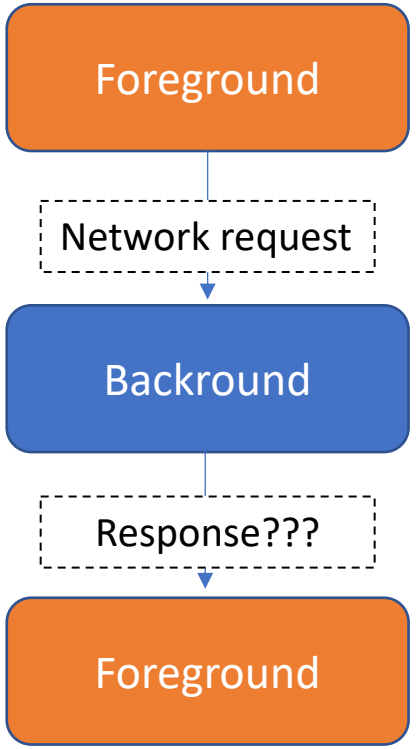
- нет автоматического оповещения о начале/конце события
- опрос сервера раз в 1-1,5 минуты (флаги начала/конца)
- сбор координат раз в n секунд
- собранные данные в конце отправляются



Потерянные ответы сетевых запросов

Приложение внезапно уходит в фон.
Выполнялась работа по отправке или загрузке данных

```
var urlConfiguration = URLSessionConfiguration.default

 lazy var urlSession: URLSession? = {
     return URLSession(configuration: urlConfiguration, delegate: self,
     delegateQueue: OperationQueue.init())
 }()
```



Потерянные ответы сетевых запросов

Решение:

Реализуем поддержку фонового режима в сетевом клиенте с помощью backgroundHandler и конфигурации сессии.

```
var urlBackgroundConf = URLSessionConfiguration.background(withIdentifier: "background")
urlBackgroundConf.isDiscretionary = true
urlBackgroundConf.shouldUseExtendedBackgroundIdleMode = true

lazy var urlBackgroundSession: URLSession? = {
    return URLSession(configuration: urlBackgroundConf, delegate: self,
    delegateQueue: OperationQueue.init())
}()
```

Потерянные ответы сетевых запросов

Решение:

Реализуем поддержку фонового режима в сетевом клиенте с помощью backgroundHandler и сессии.

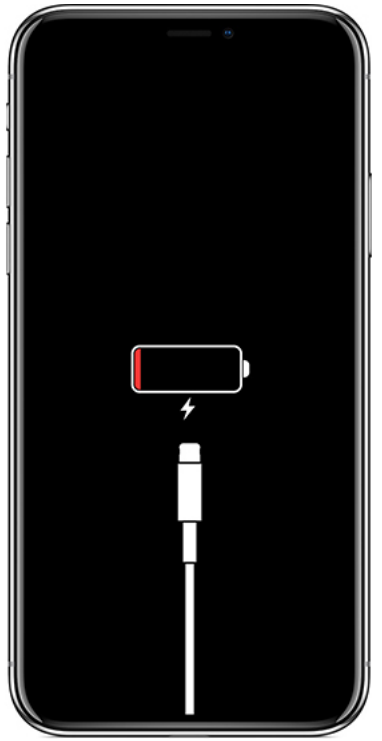
```
func application(_ application: UIApplication, handleEventsForBackgroundURLSession identifier: String,
completionHandler: @escaping () -> Void) {
    networkManager.backgroundHandler = completionHandler
}
```

Приложение с трекингом перемещения

Первая реализация (2017 год, iOS 9 – iOS 10).

Запрос раз в 60-90 создает искусственный триггер

У приложения есть возможность собрать и обработать данные



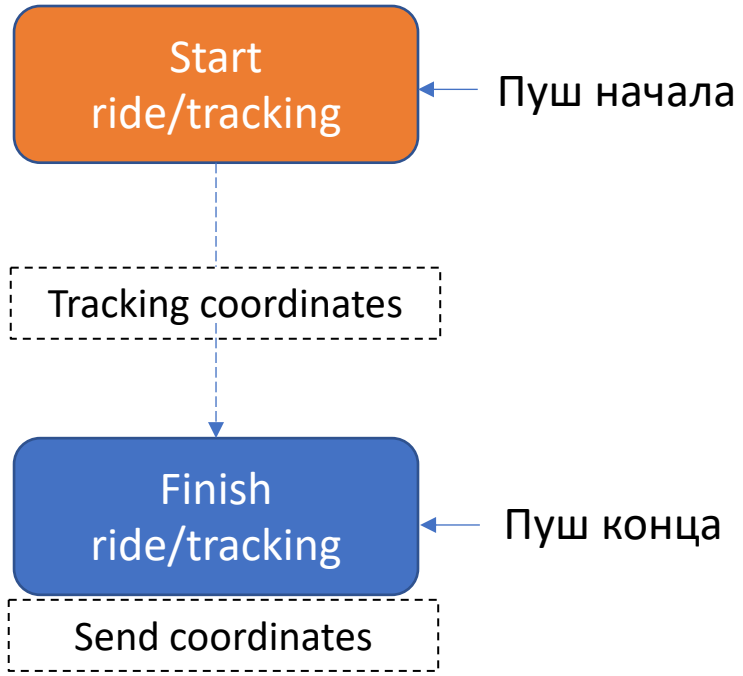
Минусы:

- затраты батареи
- затраты трафика на запрос раз в 60-90 секунд

Приложение с трекингом перемещения

Вторая реализация (2018 год, iOS 11).

- замена периодического запроса на пуш-уведомления (фоновые пуши)
- периодический запрос GPS раз в n секунд (по таймеру)
- опционально включение GPS при старте (пожелание заказчика)



Приложение с трекингом перемещения

Проблемы:

- GPS из фона не включается

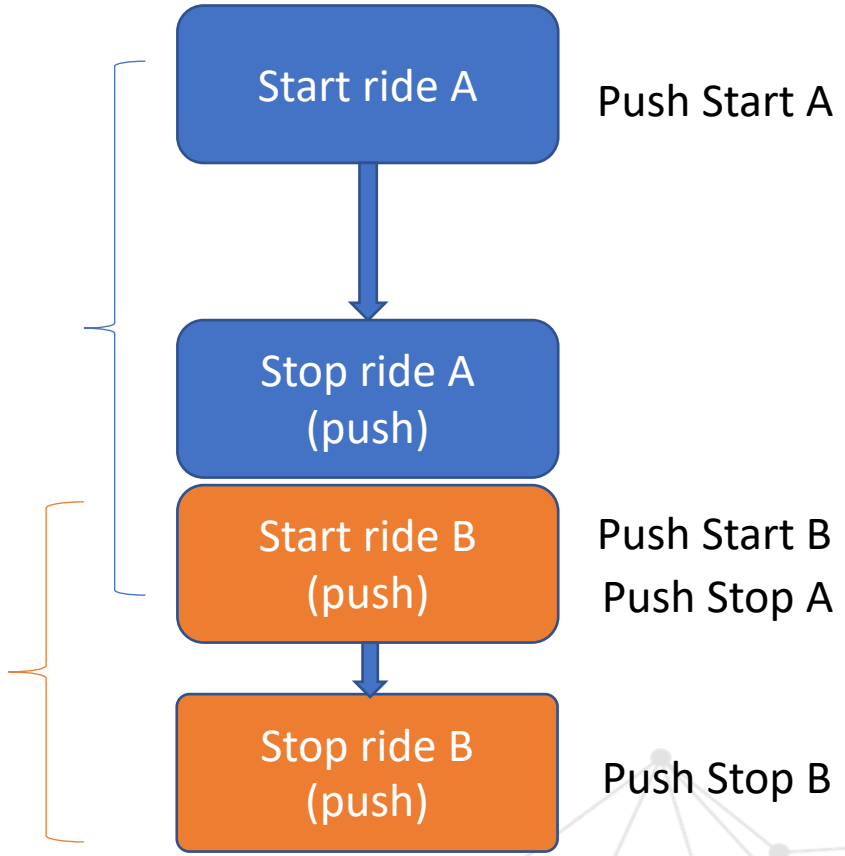
Apple запрещает включать сенсоры из фона

Приложение с трекингом перемещения

Проблемы:

- Сбой старта/окончания сбора данных из-за пуша (на сервере все ок)

- Наложение поездок (пуш)



Приложение с трекингом перемещения

Проблемы:

- GPS из фона не включается
- Сбой старта/окончания сбора данных из-за пуша
- Наложение поездок (пуш)
- Без триггеров приложение «засыпает»

Приложение с трекингом перемещения

Решения:

~~0. Запуск трекинга вручную из форграунда – отклонен заказчиком~~

1. GPS включен всегда.

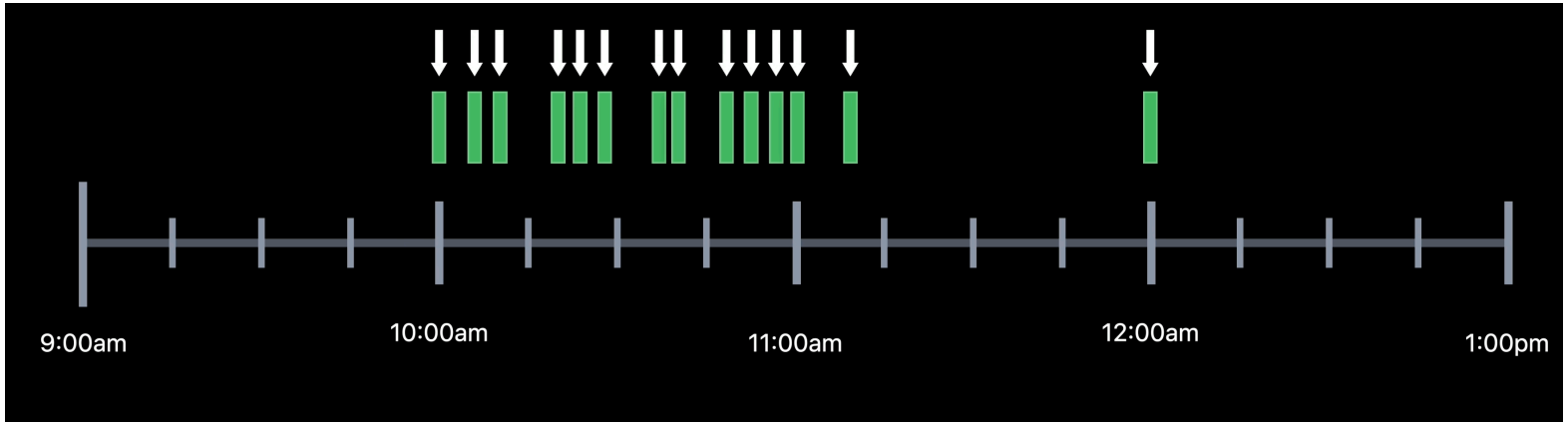
2. Координаты дополнительно логируются для последующей отправки из форграунда

3. Добавлена поддержка Silent-пушей.

Приложение с трекингом перемещения

Минусы:

- Silent-пуши не регулярны и не будят приложение так, как надо
- Координаты теряются



Приложение с трекингом перемещения

Реализация (2019 год, iOS 12).

- iOS управляет регулярностью пушей.
- Время работы в фоне сокращается до 30 секунд

Приложение с трекингом перемещения

Реализация (2019 - 2020 год, iOS 12 – iOS 13).
Решение:

- priority сообщений до 5 (рекомендация Apple)

```
{
//...
content-available: 1

apns-priority: 5

apns-push-type: background

//...
}
```

Приложение с трекингом перемещения

Реализация (2019 - 2020 год, iOS 12 – iOS 13).
Решение:

- перенос ответственности за триггер начала/конца на стороннее решение, Firebase
  («Firebase – не только пуши», Mobius Spb 2020
  доклад А.Немцева)

```
private var someRef: DatabaseReference?
private var someObserver: UInt?

private func startListening() {
    if someObserver == nil {
        someObserver = someRef.observe(.value, with: handleChanged)
        }
}

 private func handleChanged(snapshot: DataSnapshot) {
    //EVENT!!!!
 }
```

Приложение с трекингом перемещения

Реализация (2019 - 2020 год, iOS 12 – iOS 13).
Решение:

- отслеживание изменения региона местоположения

```
func locationManager(_ manager: CLLocationManager, didEnterRegion region: CLRegion) {
    location = locationManager.location
    startMonitoringForNewLocation()
}

func locationManager(_ manager: CLLocationManager, didExitRegion region: CLRegion) {
    location = locationManager.location
    startMonitoringForNewLocation()
}

private func startMonitoringForNewLocation() {
    //Создание нового региона
    //Перезапуск
}
```

Приложение с трекингом перемещения

Реализация (2019 - 2020 год, iOS 12 – iOS 13). Решение:

- priority сообщений до 5 (рекомендация Apple)

- перенос ответственности за триггер начала/конца на стороннее решение («Firebase – не только пуши», доклад А.Немцева)

- отслеживание изменения региона местоположения

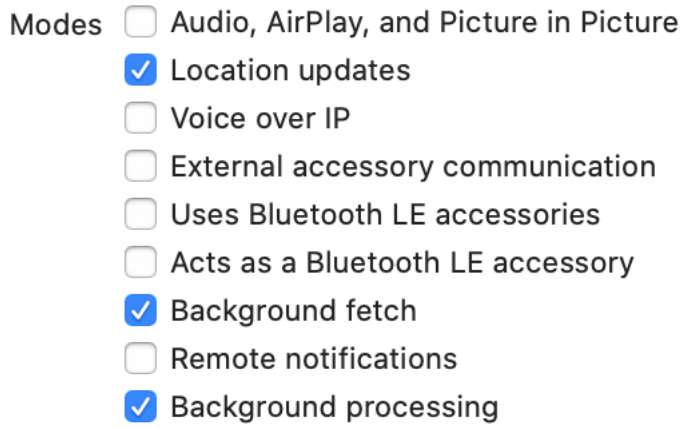
Приложение с длительным отслеживанием сенсоров

Приложение с длительным отслеживанием сенсоров

Приложение **[Данные удалены]** нон-стоп или по ручному пуску отслеживает параметры устройства через заданные промежутки времени.

Отправка событий осуществляется также с определенной периодичностью

- Location updates
- Background fetch
- Background processing



Приложение с длительным отслеживанием сенсоров

- опрос всех доступных сенсоров в фоне (в т.ч и местоположения)
- отправка данных в фоне
- шедуллинг событий по таймеру
- периоды опроса и отправки могут не совпадать с периодом в 30 секунд

Приложение с длительным отслеживанием сенсоров

Триггеры:

- Сбор координат (опционально)
- Отправка по сети (background handler + session)

Периоды опроса и отправки могут не совпадать с периодом в 30 секунд

Приложение успеет «уснуть»

VoIP – универсальное решение (из stackoverflow)

- iOS в фоне не закрывает сокет (триггер)

- Приложение перезапускается (в основном) после перезагрузки устройства или выгрузки из памяти (как sticky service в Android)

Запрет на VoIP с iOS 13

В iOS 13 Apple ограничит этот фоновый процесс: можно использовать только для вызовов без доступа к другим формам сбора данных.

Приложение не пройдет AppReview

VoIP – не универсальное решение

- iOS в фоне не закрывает сокет (триггер)
- Приложение перезапускается (в основном) после перезагрузки устройства или выгрузки из памяти (как sticky service в Android)

Если UIBackground modes в приложении указан, но не используется, приложение не пройдет AppReview

Запрет на неиспользуемые UIBackground Modes

- Использование разрешений, которые гарантированно продляют работу в фоне, без реализации функционала:

VoIP/ Audio/ BLE / Location updates ….

Приложение не пройдет AppReview

Приложение с длительным отслеживанием сенсоров

Другое потенциальное решение:

Background fetch

setMinimumBackgroundFetchInterval(_:)
application(_:performFetchWithCompletionHandler:)

```
UIApplication.shared.setMinimumBackgroundFetchInterval(1800)


func application(_ application: UIApplication,
performFetchWithCompletionHandler completionHandler:
                                    @escaping (UIBackgroundFetchResult) -> Void) {
  //Что-то происходит
  completionHandler(<...>)
}
```

Приложение с длительным отслеживанием сенсоров

Потенциальное решение:

Background fetch

Проблема:
Задается минимальное время до выполнения задачи.
Задача выполняется, когда удобно системе, а не когда нужно разработке

Приложение с длительным отслеживанием сенсоров

Решение:

- Свой таймер для работы в фоне

```
private var backTimer: Timer?

private func startTaskTimer() {
        stopTaskTimer()
        backTimer = Timer.scheduledTimer(...)
    }

 private func stopTaskTimer() {
        backTimer?.invalidate()
        backTimer = nil
    }
```

Приложение с длительным отслеживанием сенсоров

Решение:

- При переходе в фон запуск UIBackgroundTask по таймеру раз в 30 секунд

```
private func startTaskTimer() {
    stopTaskTimer()
    backTimer = Timer.scheduledTimer(timeInterval: 30,
                                     target: self,
                                     selector: #selector(startTask),
                                     userInfo: nil,
                                     repeats: true)
    RunLoop.current.add(backTimer!, forMode: .common)
}
```

Приложение с длительным отслеживанием сенсоров

Решение:

- Логика по сбору и отправке выполняется при условии, что от предыдущего сбора прошло заданное кол-во времени

```
@objc private func checkCountdown() {
    if (countDown == 0) {
        //begin read
    }
    if (countDown == timeForRead) {
        //stopRead
    }
    if (countDown == timeForSend) {
        //send Data
        countDown = -1 //little hacky

    }
    countDown += 1
}
```

Приложение с длительным отслеживанием сенсоров

Решение:
Сбор координат и отправка по сети (background handler)
выступают доп.тригерами

Summary.Приложение с длительным отслеживанием сенсоров

Решение:

- Свой таймер для работы в фоне
- При переходе в фон запуск UIBackgroundTask по таймеру раз в 30 секунд
- Логика по сбору и отправке выполняется при условии, что от предыдущего сбора прошло заданное кол-во времени
- Сбор координат и отправка по сети (background handler) выступают доп.тригерами

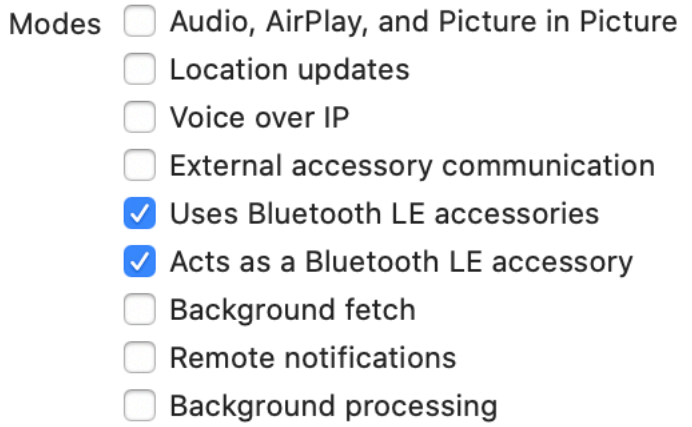
Приложение с длительным получением данных от BLE

Приложение с длительным получением данных от BLE

Приложение снимает показатели с мотоциклов **[Данные удалены]** через специальное устройство.

Связь по BLE

- Uses Bluetooth LE accessories
- Acts as a Blutooth LE accessories



Приложение с длительным получением данных от BLE

Реализация до 2018 года. Без фонового режима

Реализация 2018 год.

- Запросы к BLE менеджеру выступают в качестве триггера

Приложение с длительным получением данных от BLE

Проблема:

- Приложение общается с донглом, пока тот не «уснет». Критично на iOS 12
- После ухода в «сон» донгл не просыпается

Лечить прошивку???

Приложение с длительным получением данных от BLE

Нюанс

Донгл не определяется аппаратом (iOS, Android) как самостоятельное BLE устройство

Приложение с длительным получением данных от BLE

Решение

Не дать спать донглу.

- Опросы донгла по таймеру в рамках UIBackgroundTask

- Обработка аварийного отключения (BLE Manager)

```
func reconnect(_ central: CBCentralManager, to peripheral: CBPeripheral) {
    DispatchQueue.main.async {
        self.backgroundTaskId = UIApplication.shared.beginBackgroundTask
                                (withName: "reset_dongle") {
            //stop task
        }

        self.stopBackTimer()
        self.backTimer = Timer.scheduledTimer(...) { _ in
                         central.connect(peripheral, options: [:])
                         //stop task
        }
    }
}
```

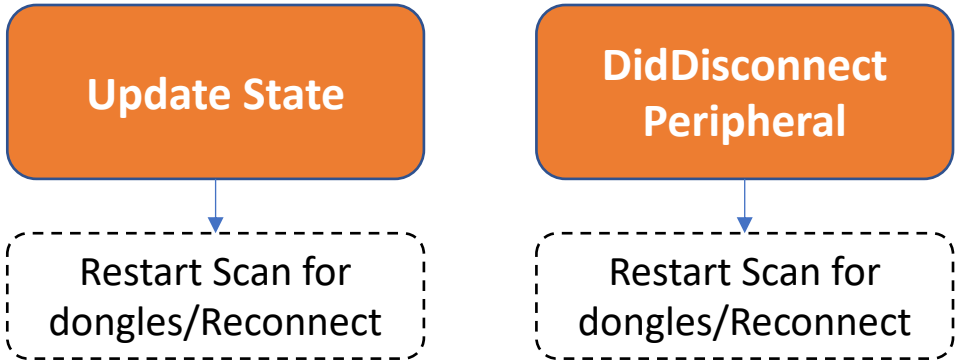
Приложение с длительным получением данных от BLE

Решение

Не дать спать донглу.

- Лечить прошивку
- Опросы донгла по таймеру в рамках UIBackgroundTask
- Обработка аварийного отключения (BLE Manager)

CBCentralManagerDelegate



iOS 13/14. Что изменилось?

iOS 13/14. Что изменилось?

- Background Processing Tasks

- Background App Refresh Tasks

iOS 13/14. Что изменилось?

- Background Processing Tasks

- Background App Refresh Tasks

```
BGTaskScheduler.shared.register(
        forTaskWithIdentifier: "com.task.id",
        using: DispatchQueue.global()) {
            task in
            self.handleTask(task)
        }
    }

private func scheduleTask() {
    do {
        let request = BGProcessingTaskRequest(identifier: "com.task.id")
        request.requiresExternalPower = true
        request.requiresNetworkConnectivity = true
        try BGTaskScheduler.shared.submit(request)
    } catch {
       print(error)
        }
}
```

iOS 13/14. Что изменилось?

- Background Processing Tasks

- Background App Refresh Tasks

```
private func scheduleAppRefresh() {
    do {
        let request = BGAppRefreshTaskRequest(identifier: "com.task.id")
        request.earliestBeginDate = Date(timeIntervalSinceNow: 3600)
        try BGTaskScheduler.shared.submit(request)
        } catch {
            print(error)
        }
    }
```

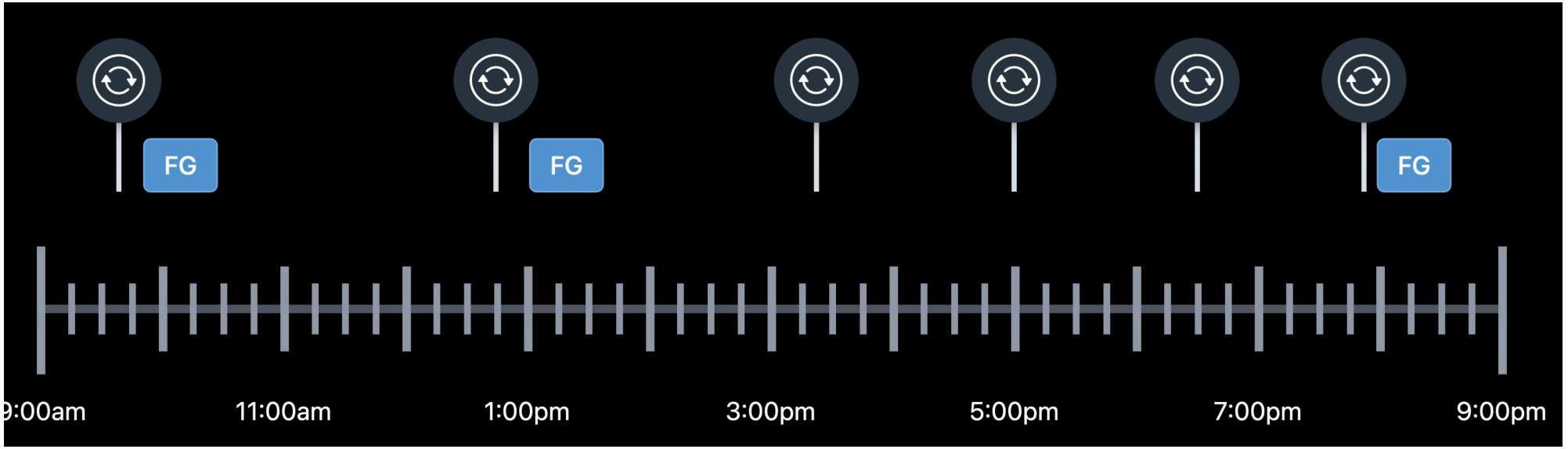
iOS 13/14. Что изменилось?

BGTaskScheduler

- Запуск фоновых задач не раньше заданной даты
- Время запуска регулируется системой, зависит от условий режима
- Право на запуск в фоне имеют только те задачи, которые будут указаны в plist

iOS 13/14. Что изменилось?

Система запоминает частоту и длительность использования приложения в Foreground и составляет график обновлений в Background



iOS 13/14. Что изменилось?

- Background Processing Tasks

- Background App Refresh Tasks

iOS 13/14. Что изменилось?

Подойдет для:

- нерегулярных задач, когда точное время старта задачи не критично

Поможет ли решению других кейсов? Нет

Summary. Тенденция

- Ожидаем еще больше ограничений для перформанса от Apple?
- Новое нативное решение?

Sources

- https://developer.apple.com/videos/play/wwdc2019/707/
- https://developer.apple.com/documentation/backgroundtasks

Спасибо за внимание!

@anioutkajarkova

azharkova
prettygeeknotes