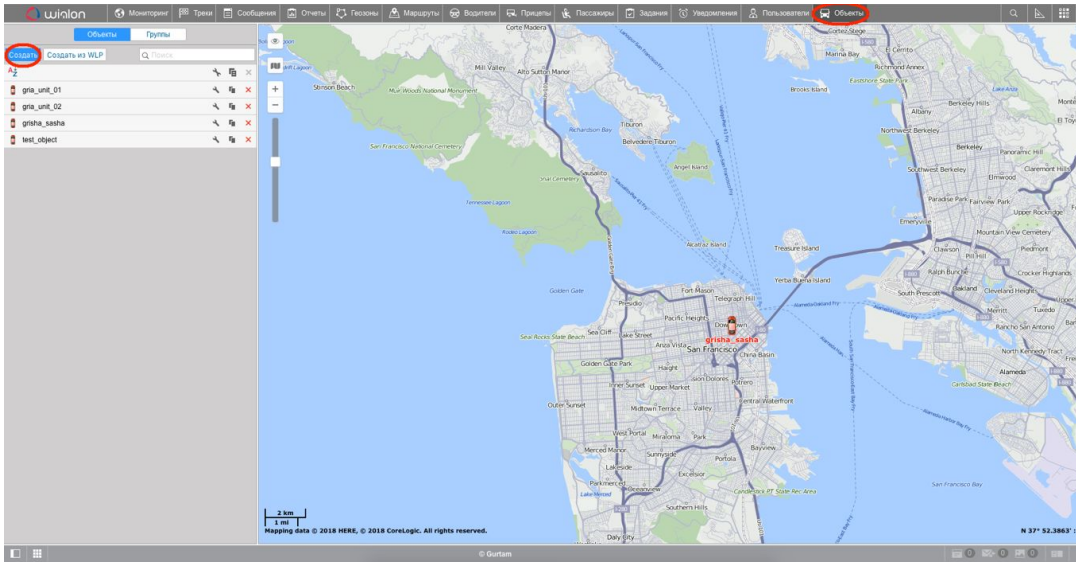
Gurtam выпустили библиотеку WiaTagKit, которая позволяет мобильным устройствам «общаться» с платформой Wialon. Это значит, что теперь любое ваше приложение можно легко превратить в трекер или вообще написать новое.

Почему следует воспользоваться новой библиотекой WiaTagKit?

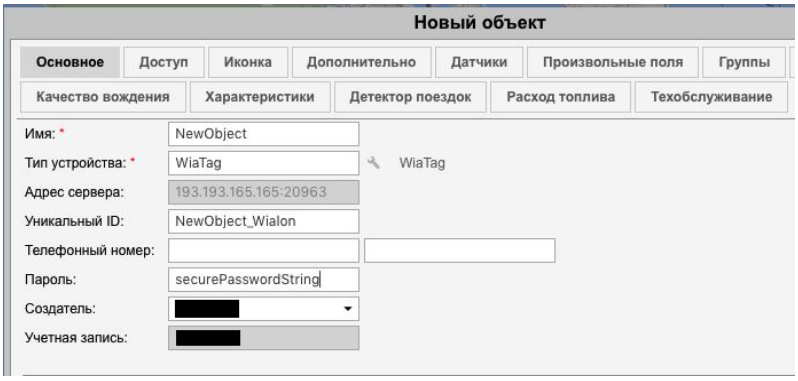
- **Новый закрытый протокол**. Более защищенный, современный и компактный (по сравнению с GPSTag), а значит он потребляет меньше трафика и его сложнее взломать.
- **Вы получаете готовый компонент** и экономите время и деньги на разработке. Таким образом ваши решения появляются на рынке быстрее, и вам не нужно тратить силы на реализацию протокола связи с Wialon.

Руководство для iOS

1. Создаем объект, который будет отправлять сообщения в Wialon: заходим в учетную запись, открываем вкладку **«Объекты»**, нажимаем кнопку **«Создать»**.



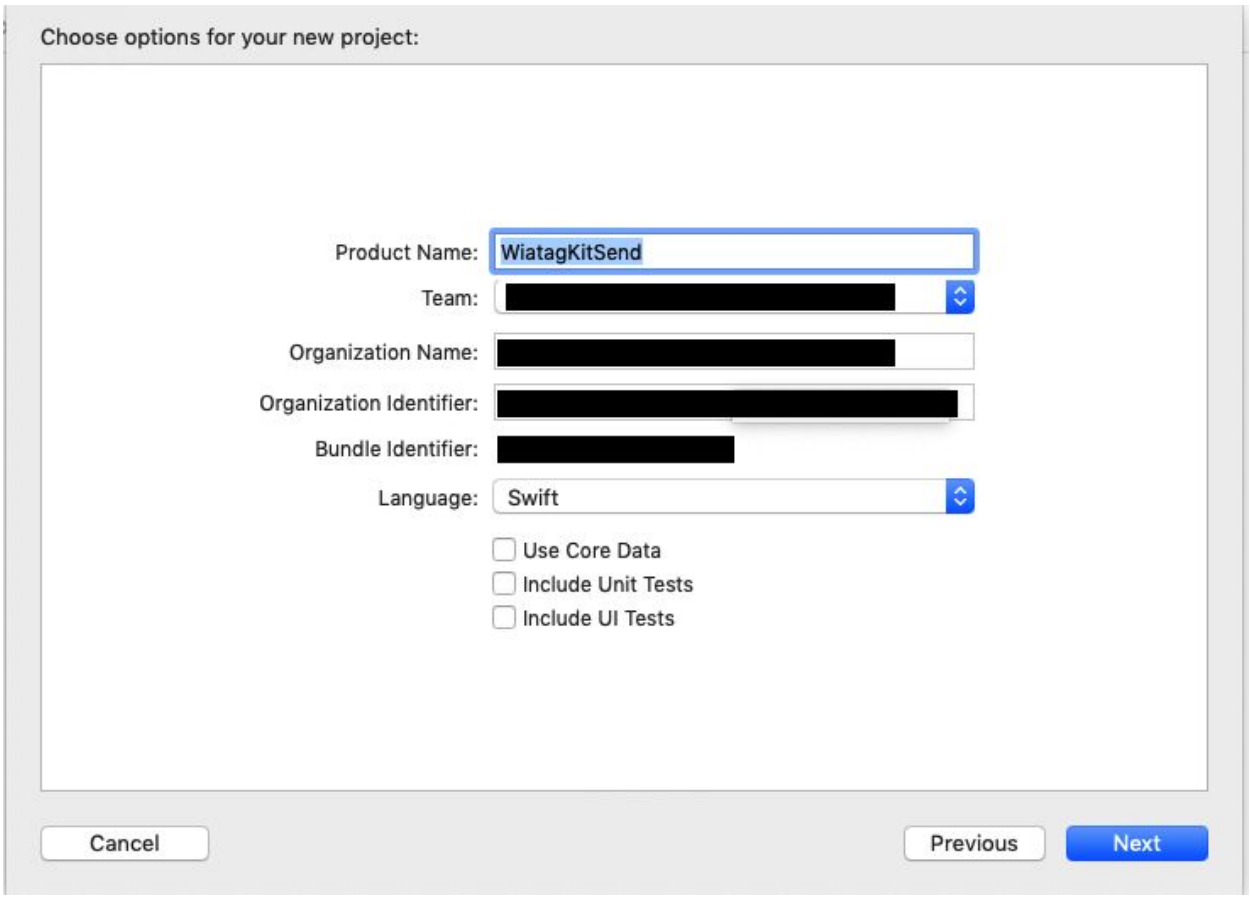
2. Настраиваем объект в появившемся диалоге: задаем **«Имя»**, выбираем **«Тип устройства»** WiaTag, заполняем поля **«Уникальный ID»** и **«Пароль».**
Важно! Любой, кто знает ID и пароль объекта, может отправлять в Wialon сообщения от его имени. Поэтому убедитесь, что пароль знают только доверенные лица.



Для отправки сообщения нам понадобится содержимое полей **«Адрес сервера»**, **«Уникальный ID»** и **«Пароль»**.

Приступаем к написанию кода

Работаем в среде **Xcode 10**, выбираем язык **Swift 4.2**. Создаем **Single-view App** с именем WiatagKitSend.



Подключаем **wiatag-kit-ios** при помощи **cocoapods**. Для этого в терминале выполняем:

```
cd /path/to/your/project/
pod init
```

Затем открываем **Podfile** и вставляем туда строки:

```
pod 'WiaTagKit'
```

Возвращаемся к терминалу и выполняем:

```
pod install
```

Закрываем проект, открываем **workspace**. Переходим во **ViewController.swift** и импортируем модуль **WiaTagKit**.

Отправляем сообщение в Wialon

1. Инициализируем **WTMessageSender,** используя **«Адрес сервера»**, **«Уникальный ID»** и **«Пароль»** объекта в Wialon.
2. Инициализируем объект **WTMessage**.
3. Отправляем **WTMessage**, используя соответствующий метод **WTSender**.

Вот пример реализации **ViewController**, которая отправляет пустое сообщение, когда контроллер показан:

```
import UIKit
import WiaTagKit

class ViewController: UIViewController {

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        sendMessage()
    }

    func sendMessage() {
        let host = "193.193.165.165"
        let port: UInt = 20963
        let unitId = "NewObject_Wialon"
        let password = "securePasswordString"
        let sender = WTMessageSender(host: host,
                                     port: port,
                                     unitId: unitId,
                                     password: password)
        let message = WTMessage { builder in }
        sender.send(message) { error in
            guard let error = error else {
                print("message sending completed with success")
                return
            }
            print("message sending failed with error \(error)")
        }
    }
}
```

Копируем этот код и вставляем его в ваш **ViewController.swift.**

Важно! Значения **host**, **port**, **unitId** и **password** нужно заменить на **«Адрес сервера» (IP, порт)**, **«Уникальный ID»** и **«Пароль»** вашего объекта типа WiaTag.

Запускаем проект
Если вы все сделали правильно, в консоли Xcode появляется сообщение.

message sending completed with success

Это значит, что мы отправили пустое сообщение.

Добавляем содержимое в сообщение WiaTag

В настоящий момент, сообщение может содержать следующую информацию:

1. Время создания сообщения. Отправляется в любом случае, но может быть переопределено и отличаться от реального времени создания сообщения.
2. Местоположение. Скорее всего, вы захотите его отправлять, если ваше приложение работает с сервисом **CoreLocation**. В этом случае используются два конструктора: `WTLocation(location: CLLocation)` – скорее всего, вам будет достаточно этого конструктора.

   `WTLocation(latitude: Double, longitude: Double, altitude: Double, speed: UInt16, bearing: UInt16, satellites: UInt8).`
3. Флаг SOS-сообщения.
4. Изображение. Для отправки изображения используется класс WTImage.
5. Текстовое сообщение.
6. Уровень заряда батареи.
7. Параметры. Имеют ключи (только текстовые) и значения (текстовые, бинарные и типов **Int**, **Long**, **Float**, **Double**). Нельзя отправить 2 параметра для одного и того же значения ключа.

Для примера создадим сообщение, которое отправляет время, SOS-сигнал, изображение, текстовое сообщение и Int-параметр:

```
let message = WTMessage { builder in
            //setup time
            let date = Date(timeIntervalSinceNow: -20)
            builder.time = date
            //setup location
            let location = CLLocation(latitude: 53, longitude: 27)
            builder.location = WTLocation(location: location)
            //setup image
            let image = UIImage(named: "free_image.jpg")
            let imageData = image?.jpegData(compressionQuality: 1)
            if let imageData = imageData {
                builder.image = WTImage(imageData: imageData,
                                        named: "imageName.jpg")
            }
            //setup SOS signal
```

```
            builder.isSos = true
            //setup text message
            builder.text = "This is my text message!"
            //setup int param
            builder.addParam("int value", withIntValue: 3)
        }
```

Можно отправить массив сообщений одним вызовом. Это более экономично и удобно, если нужно отправить несколько сообщений.

```
func sendMessages() {
        var messages = [WTMessage]()

        let host = "193.193.165.165"
        let port: UInt = 20963
        let unitId = "NewObject_Wialon"
        let password = "securePasswordString"
        let sender = WTMessageSender(host: host,
                                     port: port,
                                     unitId: unitId,
                                     password: password)

        for i in 0...10 {
            let message = WTMessage { builder in
                //setup time
                let date = Date(timeIntervalSinceNow: TimeInterval(-i * 10))
                builder.time = date
                //setup location
                let location = CLLocation(latitude: 53, longitude: 27)
                builder.location = WTLocation(location: location)
                //setup image
                let image = UIImage(named: "free_image.jpg")
                let imageData = image?.jpegData(compressionQuality: 1)
                if let imageData = imageData {
                    builder.image = WTImage(imageData: imageData,
                                            named: "imageName.jpg")
                }
                //setup SOS signal
                builder.isSos = true
                //setup text message
                builder.text = "This is my \(i) text message!"
                //setup int param
```

```
            builder.addParam("int value", withIntValue: 3)
        }
        messages.append(message)
    }

    sender.send(messages) { error in
        guard let error = error else {
            print("message sending completed with success")
            return
        }
        print("message sending failed with error \(error)")
    }
}
```

Это один из примеров работы с **WiaTagKit.** По любым вопросам использования новой библиотеки обращайтесь на development@gurtam.com.