



Урок 12

CI

Непрерывная интеграция. Fastlane.

[Continuous Integration \(CI\)](#)

[Fastlane](#)

[Практическое задание](#)

[Дополнительные материалы](#)

Continuous Integration (CI)

Continuous Integration(CI) — это практика разработки программного обеспечения, в которой члены команды часто используют свою работу. Это приводит к слиянию рабочих копий в общую основную ветвь разработки (минимум — ежедневно) и выполнению частых автоматизированных сборок проекта, чтобы скорее выявлять потенциальные дефекты и решать интеграционные проблемы. В обычном проекте, где над разными его частями разработчики трудятся независимо, стадия интеграции является финальной. Она может задержать окончание работ на неопределенное время. Переход к непрерывной интеграции позволяет снизить трудоемкость этого процесса и сделать его более предсказуемым за счет раннего обнаружения и устранения ошибок. Основное преимущество — сокращается стоимость исправления дефекта за счет раннего выявления.

Выясним, что такое CI и как он работает, на простом примере разработки небольшой задачи, которая может быть выполнена за несколько часов.

Сначала берем копию проекта из git-репозитория на свою машину разработки. Пишем программный код, необходимый для выполнения задачи. Этот этап может состоять из изменения основного кода, добавления нового или изменения тестов. CI предполагает высокую степень автоматизированного тестирования (часто модульного).

Как только работа по задаче завершается, выполняется сборка проекта и запускаются тесты. Только если сборка была успешной и тесты прошли без ошибок, выполнение задачи считается хорошим, но не окончательным.

Далее нужно внести изменения в репозиторий. Разумеется, и другие разработчики могут это сделать. Поэтому сначала нужно обновить рабочую копию. После — произвести разрешение возможных конфликтов, собрать проект и запустить тесты. В случае успеха переносим изменения в git-репозиторий.

Но и на этом этапе не завершается работа над задачей! На машине CI из кода в git-репозитории производится сборка проекта и запуск его тестов. Только когда эти шаги завершатся успешно, задачу можно считать полностью выполненной.

Всегда есть риск что-то пропустить на локальной машине разработчика (репозиторий не был должным образом обновлен, сборка проекта прошла некорректно). С использованием CI возможная ошибка интеграции обнаруживается быстро. На этом этапе самая важная задача — исправить это и снова построить сборку. В среде CI никогда не должно быть неудачного завершения сборки интеграции. Время от времени возникают ошибки, но их следует быстро исправлять.

Результат работы CI — стабильная часть проекта, которая работает правильно и не содержит ошибок.

Есть много решений для реализации CI — как платных, так и бесплатных. Самые распространенные: **TeamCity**, **Jenkins** и **Fastlane**.

TeamCity — это многофункциональный сервер непрерывной интеграции от компании JetBrains, готовый к работе сразу после установки. Он поддерживает множество систем контроля версий, аутентификации, сборки и тестирования «из коробки». Присутствует бесплатная и коммерческая версия.

Jenkins — это сервер непрерывной интеграции с открытым исходным кодом. Он написан на Java и имеет большое число плагинов для расширения функционала. Документацию по установке можно найти здесь — <https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins>, ссылка для поиска плагинов — <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>.

В этом уроке будем рассматривать Fastlane.

Fastlane

Fastlane — это консольная утилита, оптимизированная для мобильных устройств CI. Предоставляет простой способ автоматизировать бета-развертывание и выпуск приложений для iOS и Android. Сайт — <https://fastlane.tools>.

Почему Fastlane?

- Автоматическая отправка в App Store;
- Автоматизация запуска тестов;
- Работает на вашем компьютере;
- Поддержка приложений iOS, Mac и Android;
- Большой набор плагинов, есть возможность создавать собственные;
- Обширная документация (<https://docs.fastlane.tools>);
- Лицензия MIT.

Установка Fastlane

Обратимся к документации по установке (<https://docs.fastlane.tools>) и выполним следующие шаги. Установим инструменты командной строки Xcode, выполнив команду в терминале:

```
xcode-select --install
```

Установим сам **Fastlane**, выполнив следующую команду в терминале:

```
sudo gem install fastlane -NV
```

Теперь перейдем в терминале в каталог проекта и выполним команду по настройке **Fastlane**:

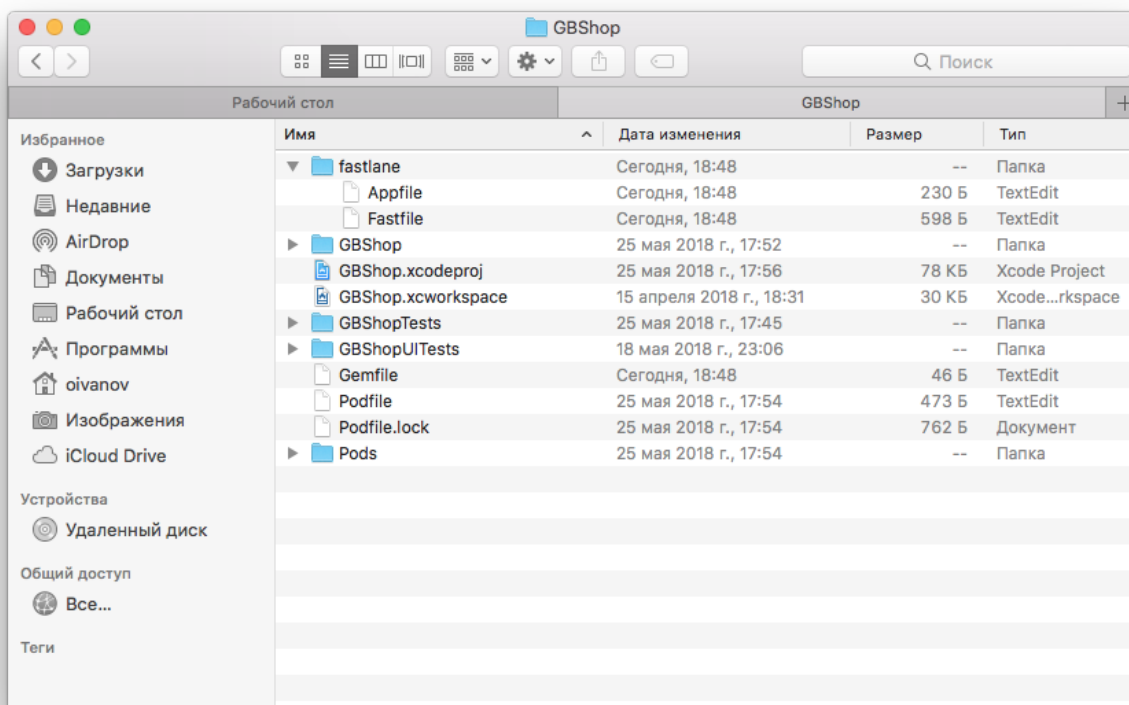
```
fastlane init
```

Нам предложат выбор варианта использования **Fastlane**:

```
mac2:GBShop oivanov$ fastlane init
[✓] Looking for iOS and Android projects in current directory...
[18:43:55]: Created new folder './fastlane'.
[18:43:55]: Detected an iOS/macOS project in the current directory: 'GBShop.xcworkspace'
[18:43:56]: -----
[18:43:56]: --- Welcome to fastlane 🚀 ---
[18:43:56]: -----
[18:43:56]: fastlane can help you with all kinds of automation for your mobile app
[18:43:56]: We recommend automating one task first, and then gradually automating more over time
[18:43:56]: What would you like to use fastlane for?
1. 📸 Automate screenshots
2. 🚀 Automate beta distribution to TestFlight
3. 📱 Automate App Store distribution
4. 🛠️ Manual setup - manually setup your project to automate your tasks
? ☐
```

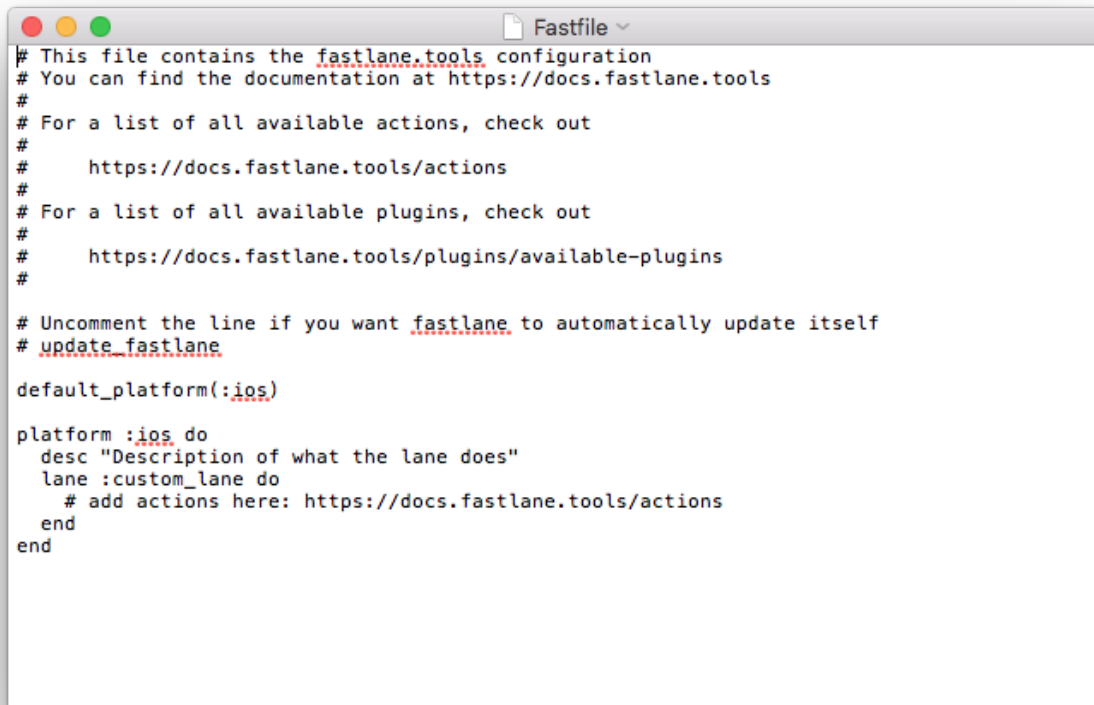
Выбираем вариант 4 — **Manual Setup**. Ниже произведем собственную настройку **Fastlane** для проекта **GBShop**.

После успешной инициализации **Fastlane** можно наблюдать изменение файловой организации проекта — в его корне появилась папка **fastlane** с файлами **Appfile** и **Fastfile**.



Appfile содержит настройки проекта для публикации. В нем можно указывать **bundle identifier** проекта, **Developer Portal Team ID**.

Fastfile содержит всю информацию, необходимую для распространения приложения. При создании он имеет следующий вид:



```
# This file contains the fastlane.tools configuration
# You can find the documentation at https://docs.fastlane.tools
#
# For a list of all available actions, check out
#
#   https://docs.fastlane.tools/actions
#
# For a list of all available plugins, check out
#
#   https://docs.fastlane.tools/plugins/available-plugins
#

# Uncomment the line if you want fastlane to automatically update itself
# update_fastlane

default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do
    # add actions here: https://docs.fastlane.tools/actions
  end
end
```

В нем по умолчанию представлена платформа ios и пустая полоса (lane) **custom_lane**. Выполним ее с помощью следующей команды в терминале:

```
mac2:GBShop oivanov$ fastlane ios custom_lane
[✓] 🚀
[19:53:17]: fastlane detected a Gemfile in the current directory
[19:53:17]: however it seems like you don't use `bundle exec`
[19:53:17]: to launch fastlane faster, please use
[19:53:17]:
[19:53:17]: $ bundle exec fastlane ios custom_lane
[19:53:17]:
[19:53:17]: Get started using a Gemfile for fastlane https://docs.fastlane.tools/getting-started/ios/se
tup/#use-a-gemfile
[19:53:18]: -----
[19:53:18]: --- Step: default_platform ---
[19:53:18]: -----
[19:53:18]: Driving the lane 'ios custom_lane' 🚀

+-----+-----+-----+
|               fastlane summary               |
+-----+-----+-----+
| Step | Action                | Time (in s) |
+-----+-----+-----+
| 1    | default_platform      | 0            |
+-----+-----+-----+

[19:53:18]: fastlane.tools finished successfully 🚀
mac2:GBShop oivanov$
```

Как видим, **custom_lane** отработала, но ничего не сделала — это и понятно, ведь она пока пустая. Заполним ее необходимыми действиями.

Подготовка проекта

Так как мы работаем с git-репозиторием, добавим действие **ensure_git_status_clean**. Оно вызывает исключение, если есть незафиксированные изменения git, и завершает работу Fastlane.

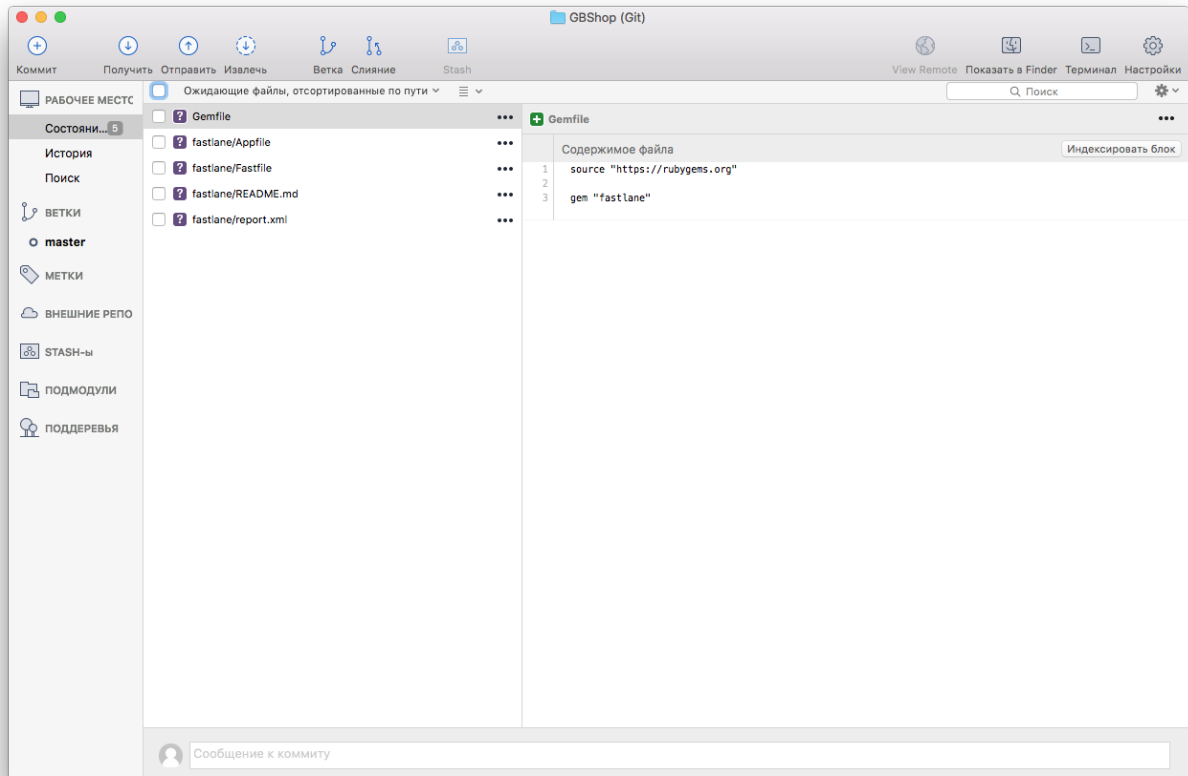
```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    ensure_git_status_clean

  end
end
```

С помощью приложения **Sourcetree** видим несколько отсутствующих файлов в git.



Выполняем еще раз команду запуска **custom_lane**:

```
fastlane ios custom_lane
```

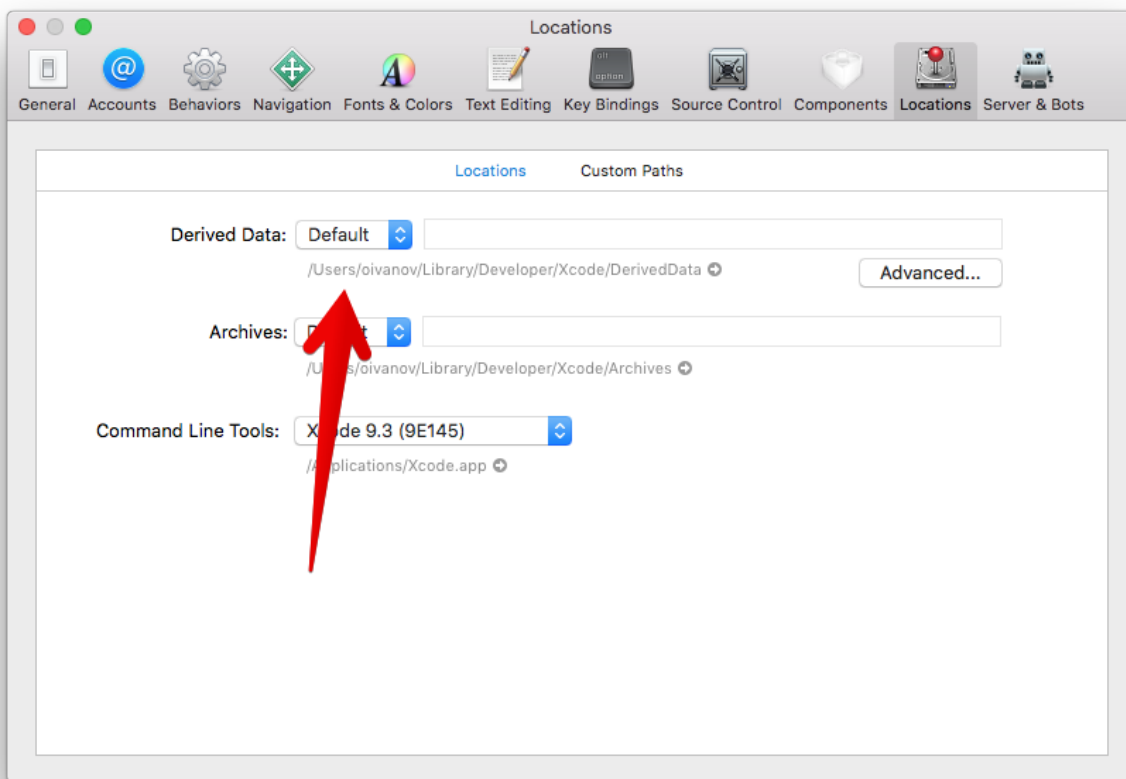
Увидим ошибку незафиксированного изменения git:

```
tup/#use-a-gemfile
[20:11:09]: -----
[20:11:09]: --- Step: default_platform ---
[20:11:09]: -----
[20:11:09]: Driving the lane 'ios custom_lane' 🚀
[20:11:09]: -----
[20:11:09]: --- Step: ensure_git_status_clean ---
[20:11:09]: -----
[20:11:09]: $ git status --porcelain
[20:11:09]: ▶ ?? Gemfile
[20:11:09]: ▶ ?? fastlane/
+-----+
|           Lane Context           |
+-----+
| DEFAULT_PLATFORM | ios |
| PLATFORM_NAME    | ios |
| LANE_NAME        | ios custom_lane |
+-----+
[20:11:09]: Git repository is dirty! Please ensure the repo is in a clean state by committing/stashing/
discarding all changes first.
+-----+
|           fastlane summary           |
+-----+
| Step | Action | Time (in s) |
+-----+
| 1    | default_platform | 0 |
| ✨    | ensure_git_status_clean | 0 |
+-----+
[20:11:09]: fastlane finished with errors
[!] Git repository is dirty! Please ensure the repo is in a clean state by committing/stashing/discardi
ng all changes first.
mac2:GBShop oivanov$
```

Пока закомментируем это действие, так как будем постоянно модифицировать файл **Fastfile**.

Произведем очистку проекта с использованием действий **clear_derived_data** (удаляет производные данные Xcode) и **xclean** (очищает проект, используя **xcodebuild**).

clear_derived_data на самом деле очищает каталог **DerivedData** с производными данными Xcode (кэши проекта).



Для **xcclean** используем дополнительные параметры **workspace("GBShop.xcworkspace")** и **scheme("GBShop")**.

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

  end
end
```

Выполняем команду запуска **custom_lane** и видим успешную очистку проекта.

```
GBShop — -bash — 109x45
[21:13:04]: however it seems like you don't use `bundle exec`
[21:13:04]: to launch fastlane faster, please use
[21:13:04]: $ bundle exec fastlane ios custom_lane
[21:13:04]: Get started using a Gemfile for fastlane https://docs.fastlane.tools/getting-started/ios/setup/#use-a-gemfile
[21:13:04]: -----
[21:13:04]: --- Step: default_platform ---
[21:13:04]: -----
[21:13:04]: Driving the lane 'ios custom_lane' 🚀
[21:13:04]: -----
[21:13:04]: --- Step: clear_derived_data ---
[21:13:04]: -----
[21:13:05]: Derived Data path located at: /Users/oivanov/Library/Developer/Xcode/DerivedData
[21:13:05]: Successfully cleared Derived Data 🗑️
[21:13:05]: -----
[21:13:05]: --- Step: xcclean ---
[21:13:05]: -----
[21:13:05]: For a more detailed xcodebuild log open /Users/oivanov/Library/Logs/fastlane/xcbuid/2018-05-30/55091/xcodebuild.log
[21:13:05]: $ set -o pipefail && xcodebuild -scheme "GBShop" -workspace "GBShop.xcworkspace" clean | tee '/Users/oivanov/Library/Logs/fastlane/xcbuid/2018-05-30/55091/xcodebuild.log' | xcpretty --color --simple
[21:13:06]: ▶ Cleaning Pods/OHHTTPStubs [Debug]
[21:13:06]: ▶ Check Dependencies
[21:13:06]: ▶ Cleaning Pods/Alamofire [Debug]
[21:13:06]: ▶ Check Dependencies
[21:13:06]: ▶ Cleaning Pods/Pods-GBShop [Debug]
[21:13:06]: ▶ Check Dependencies
[21:13:06]: ▶ Cleaning GBShop/GBShop [Debug]
[21:13:06]: ▶ Check Dependencies
[21:13:06]: ▶ Clean Succeeded

+-----+-----+-----+
|               fastlane summary               |
+-----+-----+-----+
| Step | Action                | Time (in s) |
+-----+-----+-----+
| 1    | default_platform      | 0            |
| 2    | clear_derived_data    | 1            |
| 3    | xcclean                | 1            |
+-----+-----+-----+

[21:13:06]: fastlane.tools finished successfully 🎉
mac2:GBShop oivanov$
```

Далее добавим действие проверки наличия изменений **Pods** — **cocoapods**.

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
    cocoapods(
      use_bundle_exec: false
    )
  end
end
```

```
end
end
```

Выполняем команду запуска **custom_lane** и видим успешное обновление **Pods**-проекта.

```
GBShop — -bash — 110x59
[21:00:07]: Driving the lane 'ios custom_lane' 🚀
[21:00:07]: -----
[21:00:07]: --- Step: clear_derived_data ---
[21:00:07]: -----
[21:00:08]: Derived Data path located at: /Users/oivanov/Library/Developer/Xcode/DerivedData
[21:00:17]: Successfully cleared Derived Data 🗑️
[21:00:17]: -----
[21:00:17]: --- Step: xcclean ---
[21:00:17]: -----
[21:00:18]: For a more detailed xcodebuild log open /Users/oivanov/Library/Logs/fastlane/xcbuild/2018-05-30/53718/xcodebuild.log
[21:00:18]: $ set -o pipefail && xcodebuild -scheme "GBShop" -workspace "GBShop.xcworkspace" clean | tee '/Users/oivanov/Library/Logs/fastlane/xcbuild/2018-05-30/53718/xcodebuild.log' | xcpretty --color --simple
[21:00:26]: ▶ 2018-05-30 21:00:26.961 xcodebuild[54975:8081585] DTDeviceKit: deviceType from b8dc31aad168c2f2f3c2c51f291421fabd88cfa0 was NULL
[21:00:27]: ▶ 2018-05-30 21:00:27.016 xcodebuild[54975:8081358] DTDeviceKit: deviceType from 3e3b225af8ecc3eab3db899603bd077ce53f350e1 was NULL
[21:00:27]: ▶ 2018-05-30 21:00:27.035 xcodebuild[54975:8081651] DTDeviceKit: deviceType from b8dc31aad168c2f2f3c2c51f291421fabd88cfa0 was NULL
[21:00:27]: ▶ 2018-05-30 21:00:27.060 xcodebuild[54975:8081548] DTDeviceKit: deviceType from 3e3b225af8ecc3eab3db899603bd077ce53f350e1 was NULL
[21:00:28]: ▶ Cleaning Pods/Alamofire [Debug]
[21:00:28]: ▶ Check Dependencies
[21:00:28]: ▶ Cleaning Pods/OHHTTPStubs [Debug]
[21:00:28]: ▶ Check Dependencies
[21:00:28]: ▶ Cleaning Pods/Pods-GBShop [Debug]
[21:00:28]: ▶ Check Dependencies
[21:00:28]: ▶ Cleaning GBShop/GBShop [Debug]
[21:00:28]: ▶ Check Dependencies
[21:00:28]: ▶ Clean Succeeded
[21:00:28]: -----
[21:00:28]: --- Step: cocoapods ---
[21:00:28]: -----
[21:00:28]: $ pod install
[21:00:31]: ▶ Analyzing dependencies
[21:09:08]: ▶ Downloading dependencies
[21:09:08]: ▶ Using Alamofire (4.7.2)
[21:09:08]: ▶ Using OHHTTPStubs (6.1.0)
[21:09:08]: ▶ Generating Pods project
[21:09:08]: ▶ Integrating client project
[21:09:08]: ▶ Sending stats
[21:09:08]: ▶ Pod installation complete! There are 2 dependencies from the Podfile and 2 total pods installed.
[21:09:08]: ▶ [!] Automatically assigning platform 'ios' with version '11.2' on target 'GBShop' because no platform was specified. Please specify a platform for this target in your Podfile. See 'https://guides.cocoapods.org/syntax/podfile.html#platform'.

+-----+
|               fastlane summary               |
+-----+
| Step | Action                  | Time (in s) |
+-----+
| 1    | default_platform        | 0            |
| 2    | clear_derived_data      | 490          |
| 3    | xcclean                 | 11           |
| 4    | cocoapods               | 39           |
+-----+

[21:09:09]: fastlane.tools just saved you 9 minutes! 🚀
mac2:GBShop oivanov$
```

Запуск тестов

Теперь одно из самых интересных действий автоматического запуска тестов — **scan** и **run_tests**. **Scan** использует простую **xcodebuild**-команду. Действие **scan** можно запускать отдельно через терминал:

```
fastlane scan
```

Run_tests имеет множество параметров запуска: тип(ы) симулятора, схема проекта, покрытие тестами (**code_coverage**). Пропишем это действие в **Fastfile**.

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
    cocoapods(
      use_bundle_exec: false
    )

    # run tests
    run_tests

  end
end
```

Выполняем команду запуска **custom_lane** и видим результат работы действия **ru_tests**. В проекте прошли все тесты, кроме одного.

```
GBShop — -bash — 109x45

** TEST FAILED **
[21:52:02]: Exit status: 65

+-----+
| Test Results |
+-----+
| Number of tests | 12 |
| Number of failures | 1 |
+-----+

+-----+
| Lane Context |
+-----+
| DEFAULT_PLATFORM | ios |
| PLATFORM_NAME | ios |
| LANE_NAME | ios custom_lane |
| XCODEBUILD_DERIVED_DATA_PATH | /Users/oivanov/Library/Developer/Xcode/DerivedData/GBShop-aywvflckbcxkxhdxotfikhogoaprm |
| SCAN_DERIVED_DATA_PATH | /Users/oivanov/Library/Developer/Xcode/DerivedData/GBShop-aywvflckbcxkxhdxotfikhogoaprm/Logs/Test/C7C7A13C-7A47-477C-9984-92437E9AD025_TestSummaries.plist |
| SCAN_GENERATED_PLIST_FILES | /Users/oivanov/Library/Developer/Xcode/DerivedData/GBShop-aywvflckbcxkxhdxotfikhogoaprm/Logs/Test/C7C7A13C-7A47-477C-9984-92437E9AD025_TestSummaries.plist |
+-----+

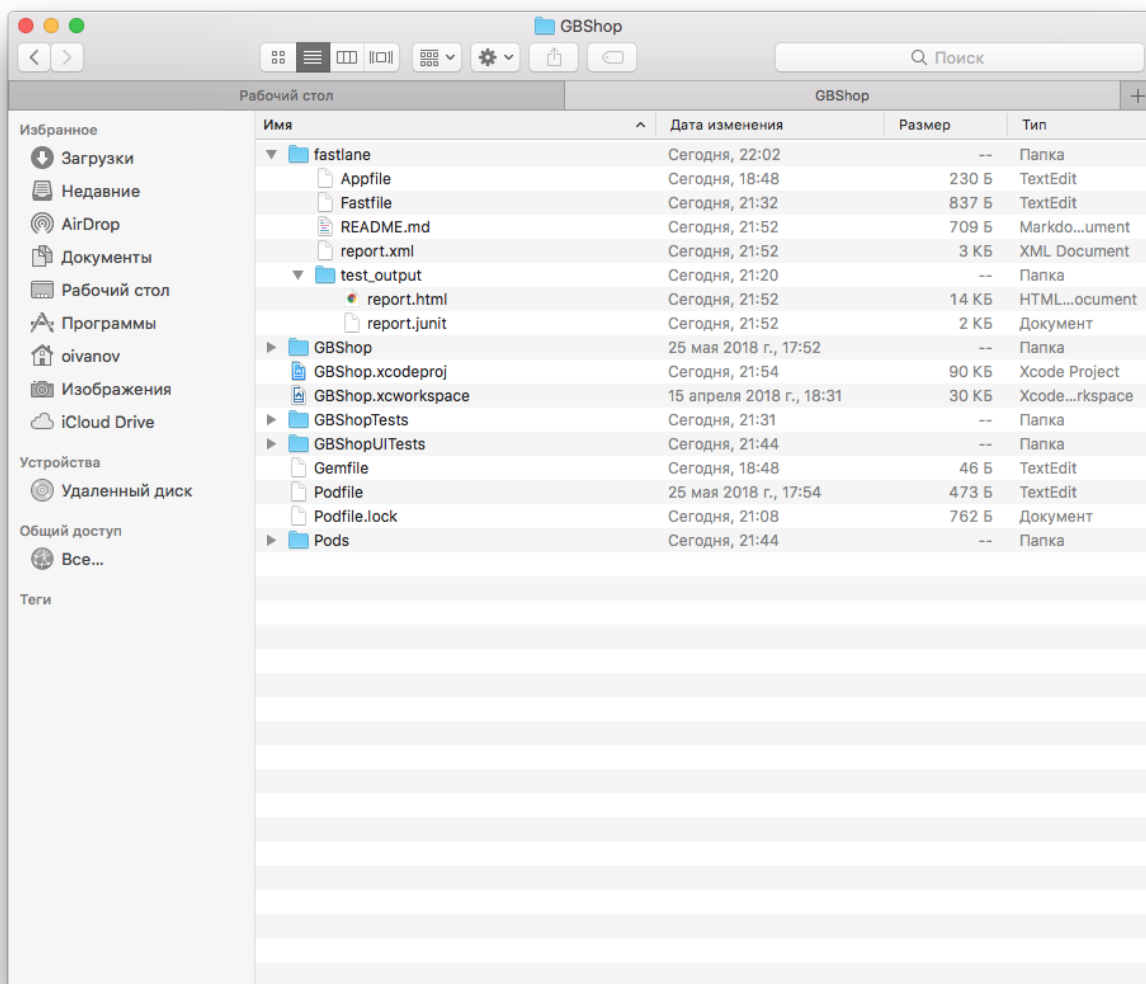
[21:52:02]: Tests have failed

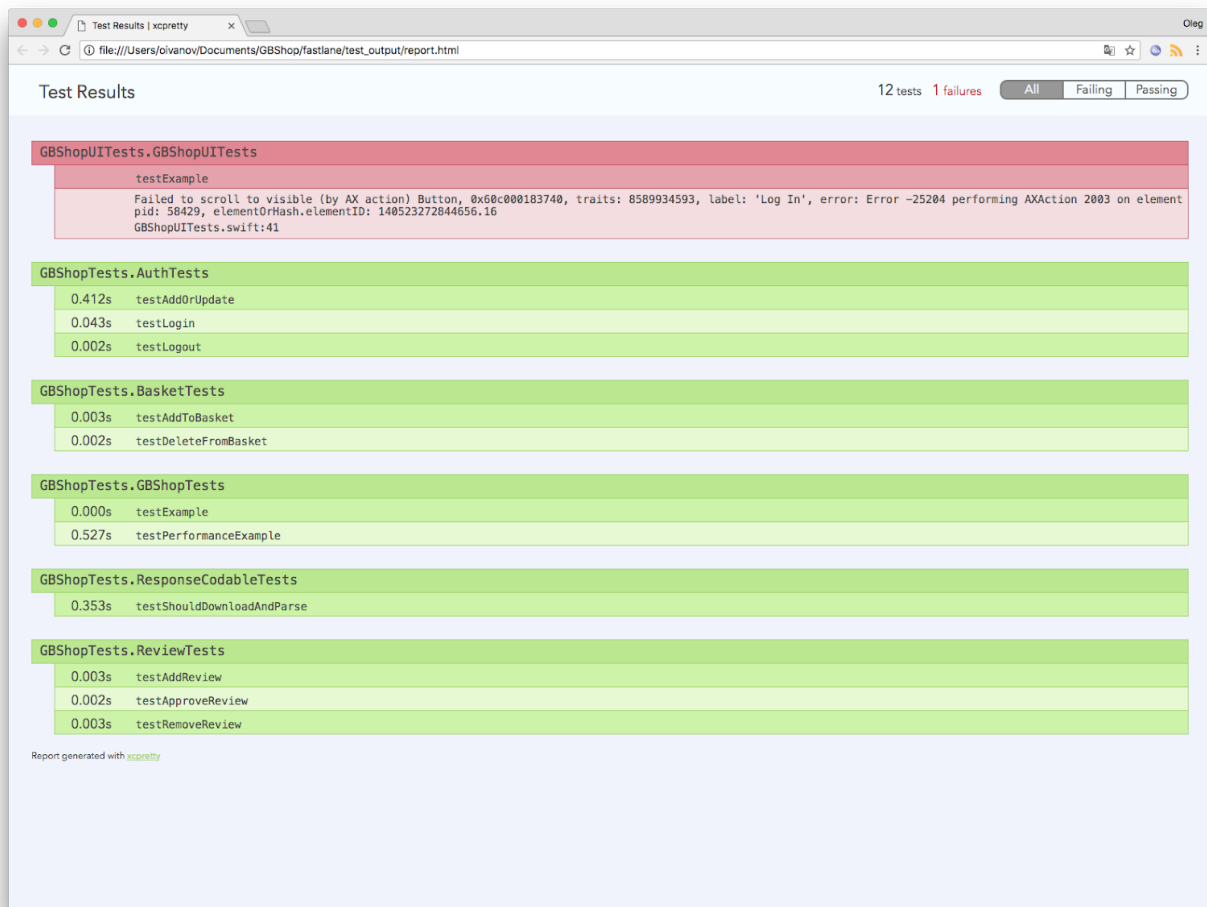
+-----+
| fastlane summary |
+-----+
| Step | Action | Time (in s) |
+-----+
| 1 | default_platform | 0 |
| 2 | clear_derived_data | 2 |
| 3 | xcclean | 4 |
| 4 | cocoapods | 38 |
| ✖ | run_tests | 401 |
+-----+

[21:52:02]: fastlane finished with errors

[!] Tests have failed
mac2:GBShop oivanov$
```

После прохождения тестов генерируются отчеты об их результатах. Они располагаются в новых файлах **report.***:





Создание скриншотов для App Store

Для хорошего оформления приложения в App Store к нему необходимо приложить скриншоты экранов. Без автоматизации этот процесс занимает много времени. На сайте <https://docs.fastlane.tools/getting-started/ios/screenshots> приведено подробное описание работы со скриншотами. С помощью **UI tests** подготавливается тест, в котором на симуляторе реализуется прохождение по нужным страницам. В соответствующих местах необходимо добавить команду создания скриншота. Все скриншоты сортируются и складываются в отдельные папки, а также генерируется обзорная html-страница. С помощью другого скрипта все скриншоты можно загрузить в **iTunesConnect** — также без участия пользователя. Остается только настроить параметры создания скриншотов.

Следуя инструкции, выполним команду в терминале относительно папки проекта:

```
fastlane snapshot init
```

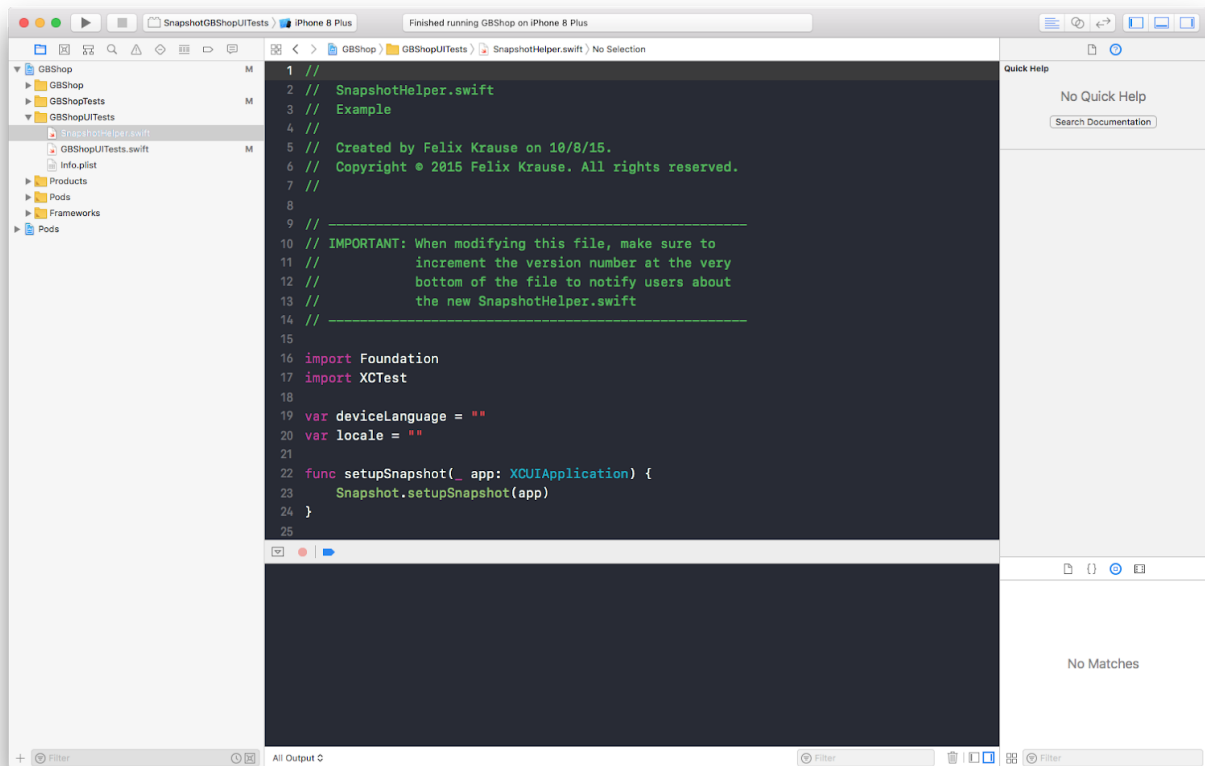
```
mac2:GBShop oivanov$ fastlane snapshot init
[✓] 🚀
✓ Successfully created SnapshotHelper.swift './fastlane/SnapshotHelper.swift'
✓ Successfully created new Snapfile at './fastlane/Snapfile'
-----
Open your Xcode project and make sure to do the following:
1) Add a new UI Test target to your project
2) Add the ./fastlane/SnapshotHelper.swift to your UI Test target
   You can move the file anywhere you want
3) Call `setupSnapshot(app)` when launching your app

let app = XCUIApplication()
setupSnapshot(app)
app.launch()

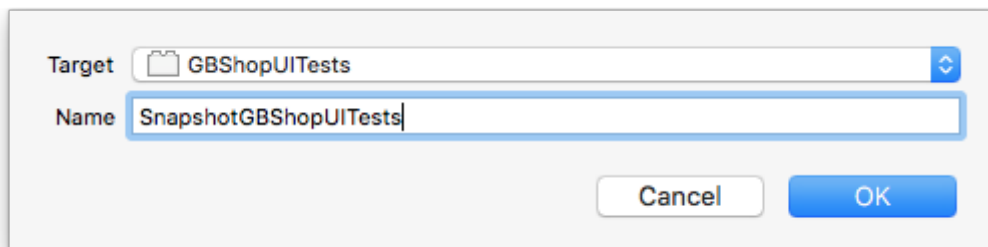
4) Add `snapshot("@Launch")` to wherever you want to trigger screenshots
5) Add a new Xcode scheme for the newly created UI Test target
6) Add a Check to enable the `Shared` box of the newly created scheme

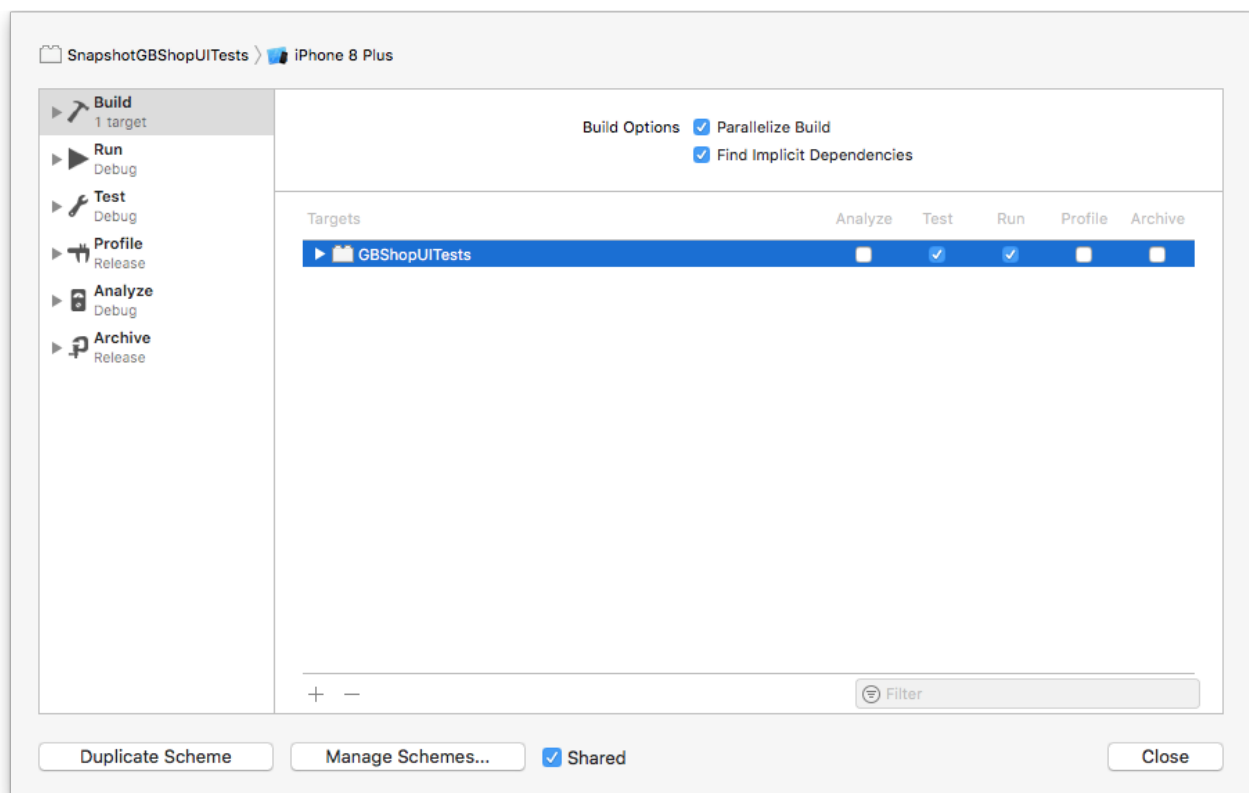
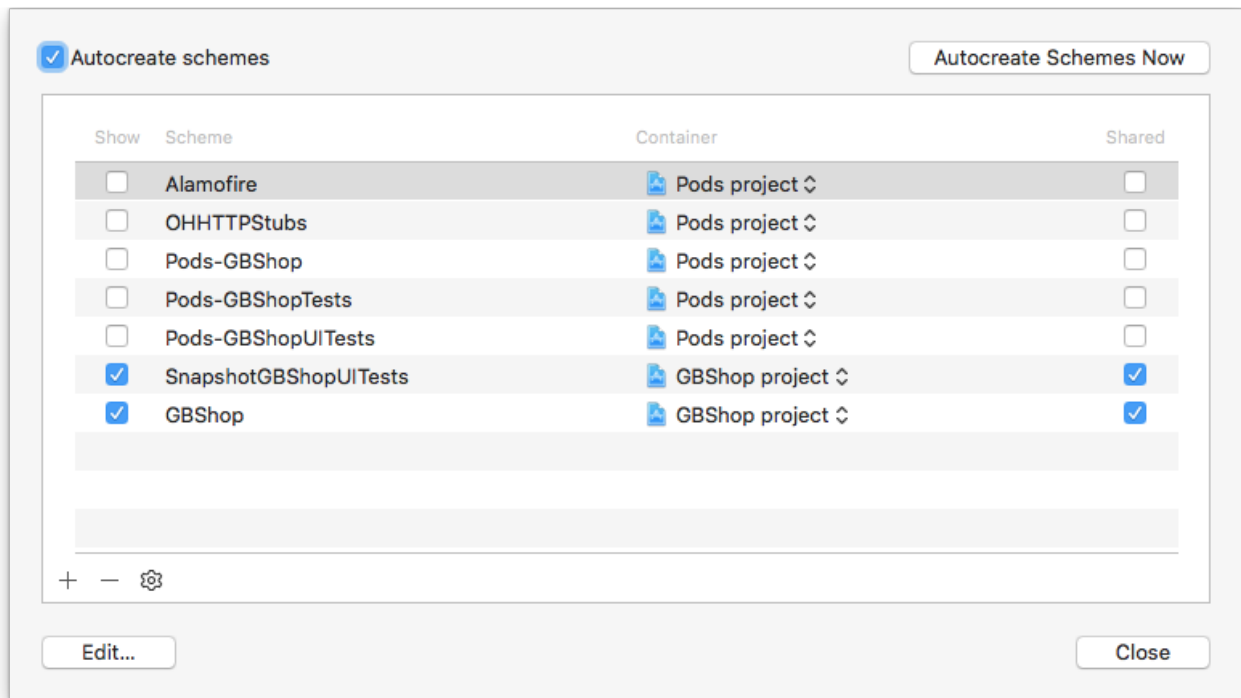
More information: https://docs.fastlane.tools/getting-started/ios/screenshots/
More information: https://docs.fastlane.tools/getting-started/ios/screenshots/
mac2:GBShop oivanov$
```

Полученный в результате работы команды файл `./SnapshotHelper.swift` добавим в проект для цели **GBShopUITests**:



Создадим отдельную схему сборки **SnapshotGBShopUITests** и установим ей флаги **Shared** и **Run**:





Переходим в тест и добавляем код настройки скриншотов **setupSnapshot()** в метод **setUp()**. Расставляем в нужных местах теста код создания скриншотов **snapshot()**:

```
class GBShopUITests: XCTestCase {
    let app = XCUIApplication()

    override func setUp() {
        super.setUp()

        continueAfterFailure = false

        setupSnapshot(app)

        app.launch()
    }

    func testExample() {

        snapshot("LoginScreen")

        app.textFields["username"].tap()
        app.textFields["username"].typeText("test")
        let passwordSecureTextField = app.secureTextFields["password"]
        passwordSecureTextField.tap()
        passwordSecureTextField.typeText("test")
        app.buttons["Log In"].tap()

        app.staticTexts["Welcome!"].tap()
        app.staticTexts["John Doe"].tap()

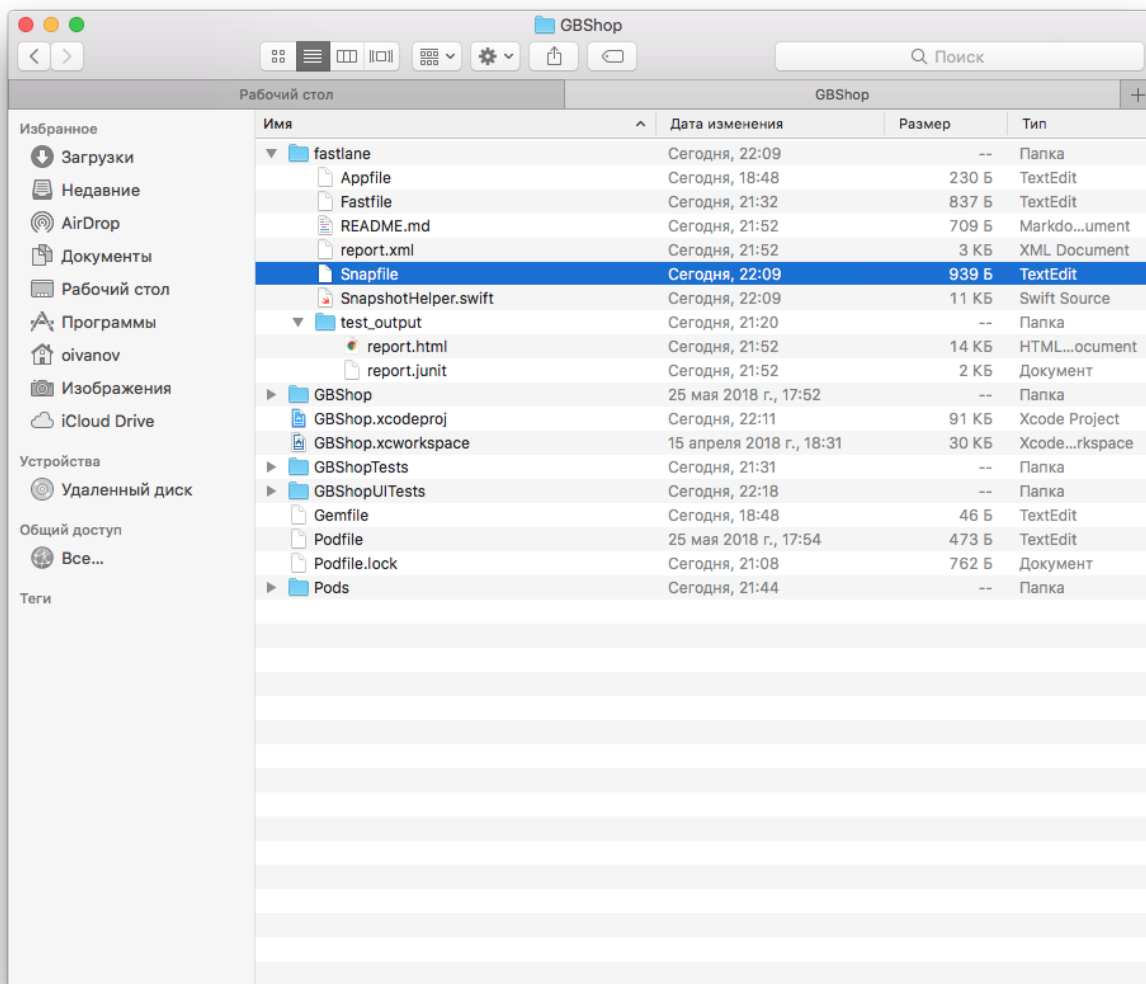
        snapshot("WelcomeScreen")

        let tabBarQuery = app.tabBars
        tabBarQuery.buttons["Product"].tap()

        snapshot("ProductScreen")
    }

    ...
}
```

Обратим внимание, что после работы команды **fastlane snapshot init** в папке проекта появился дополнительный файл **Snapfile** для настройки создания скриншотов.



Донастроим его:

```
# Uncomment the lines below you want to change by removing the # in the
beginning

# A list of devices you want to take the screenshots from
devices([
#   "iPhone 8",

# Будем запускать только на симуляторе "iPhone 8 Plus"
  "iPhone 8 Plus"

#   "iPhone SE",
#   "iPhone X",
#   "iPad Pro (12.9-inch)",
#   "iPad Pro (9.7-inch)",
#   "Apple TV 1080p"
])

languages([

# Выберем русский язык
  "ru-RU"
```

```

# "en-US",
# "de-DE",
# "it-IT",
# ["pt", "pt_BR"] # Portuguese with Brazilian locale
])

# The name of the scheme which contains the UI Tests
# Зададим нашу схему
scheme("SnapshotGBShopUITests")

# Where should the resulting screenshots be stored?
# Зададим название папки с полученными скриншотами
output_directory("./screenshots")

# remove the '#' to clear all previously generated screenshots before creating
new ones
# Настроим очищение старых файлов со скриншотами
clear_previous_screenshots(true)

# Arguments to pass to the app on launch. See
https://docs.fastlane.tools/actions/snapshot/#launch-arguments
# launch_arguments(["-favColor red"])

# For more information about all available options run
# fastlane action snapshot

```

Создание скриншотов можно выполнить следующей командой в терминале:

```
fastlane snapshot
```

Или добавлением действия **capture_ios_screenshots** в Fastfile.

```

default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

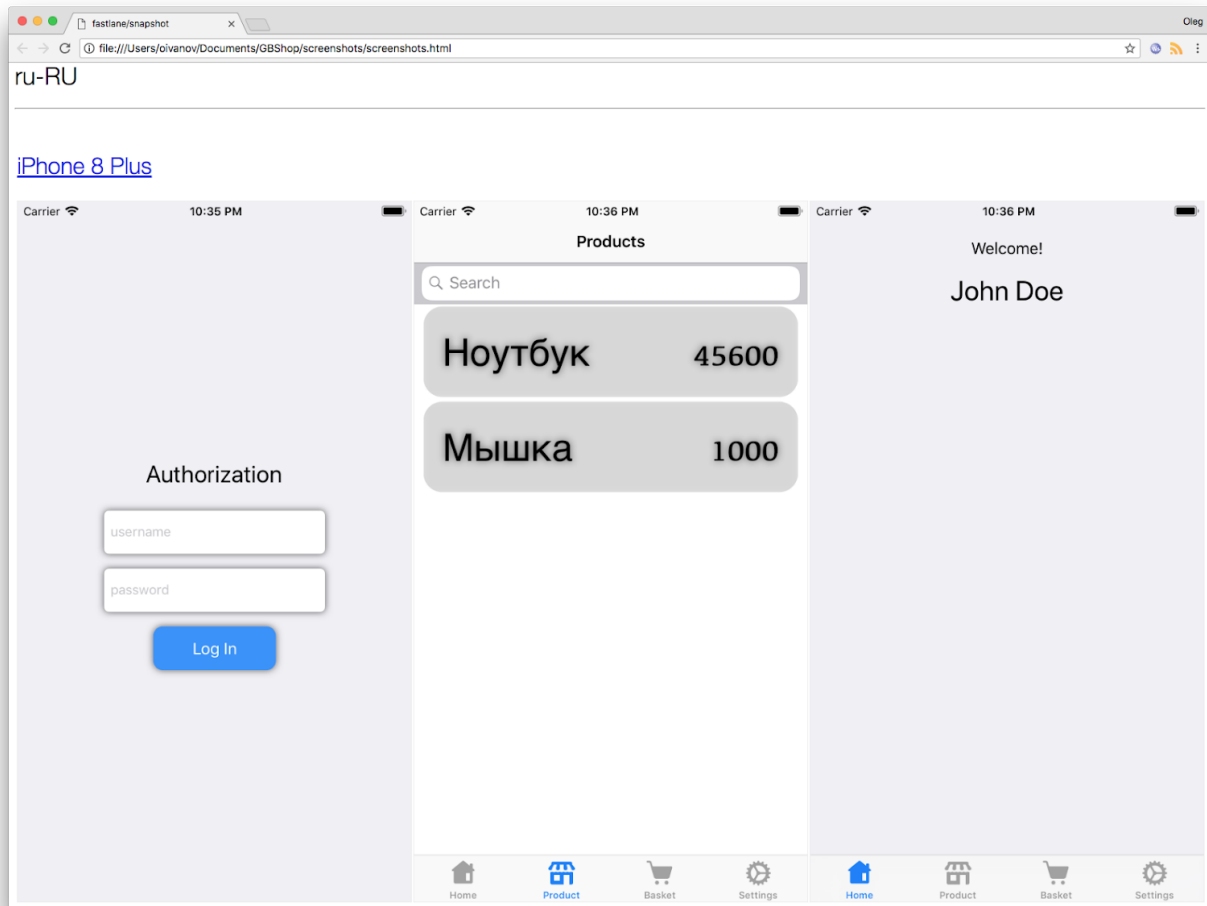
    # update a dependencies
    cocoapods(
      use_bundle_exec: false
    )

    # run tests
    run_tests
  end
end

```

```
# make screenshots
capture_ios_screenshots
end
end
```

Выполняем команду запуска **custom_lane** и видим результат работы по созданию скриншотов: автоматическое открытие в браузере по умолчанию файла **'/Users/oivanov/Documents/GBShop/screenshots/screenshots.html'**.




```
GBShop — -bash — 109x45
[22:33:59]: > Check Dependencies
[22:34:00]: > Building Pods/OHHTTPStubs [Debug]
[22:34:00]: > Check Dependencies
[22:34:00]: > Building Pods/Pods-GBShop [Debug]
[22:34:00]: > Check Dependencies
[22:34:00]: > Building GBShop/GBShop [Debug]
[22:34:00]: > Check Dependencies
[22:34:01]: > Building Pods/Pods-GBShopUITests [Debug]
[22:34:01]: > Check Dependencies
[22:34:01]: > Building GBShop/GBShopUITests [Debug]
[22:34:01]: > Check Dependencies
[22:36:08]: > Test Succeeded
[22:36:12]: > All tests
[22:36:12]: > Test Suite GBShopUITests.xctest started
[22:36:12]: > GBShopUITests
[22:36:12]: > ✓ testExample (30.606 seconds)
[22:36:12]: > Executed 1 test, with 0 failures (0 unexpected) in 30.606 (30.607) seconds
[22:36:12]: > Copying '/Users/oivanov/Documents/GBShop/screenshots/ru-RU/iPhone 8 Plus-LoginScreen.png'...
[22:36:12]: > Copying '/Users/oivanov/Documents/GBShop/screenshots/ru-RU/iPhone 8 Plus-ProductScreen.png'...
[22:36:12]: > Copying '/Users/oivanov/Documents/GBShop/screenshots/ru-RU/iPhone 8 Plus-WelcomeScreen.png'...

+-----+
| snapshot results |
+-----+
| Device | ru-RU |
+-----+
| iPhone 8 Plus | ❤️ |
+-----+

[22:36:13]: Generating HTML Report
[22:36:13]: Successfully created HTML file with an overview of all the screenshots: '/Users/oivanov/Documents/GBShop/screenshots/screenshots.html'

+-----+
| fastlane summary |
+-----+
| Step | Action | Time (in s) |
+-----+
| 1 | default_platform | 0 |
| 2 | capture_ios_screenshots | 190 |
+-----+

[22:36:18]: fastlane.tools finished successfully 🎉
mac2:GBShop oivanov$
```

Подготовка к деплою

Прежде чем выложить приложение в App Store, мы всегда меняем версию сборки. Невозможно выложить сборку с одинаковой версией. С помощью Fastlane это тоже можно делать.

Действие **increment_build_number** увеличивает номер сборки проекта. Чтобы его применить, создадим в **Fastfile** новый **lane project** и в него добавим это действие.

```
platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
```

```

cocoapods (
  use_bundle_exec: false
)

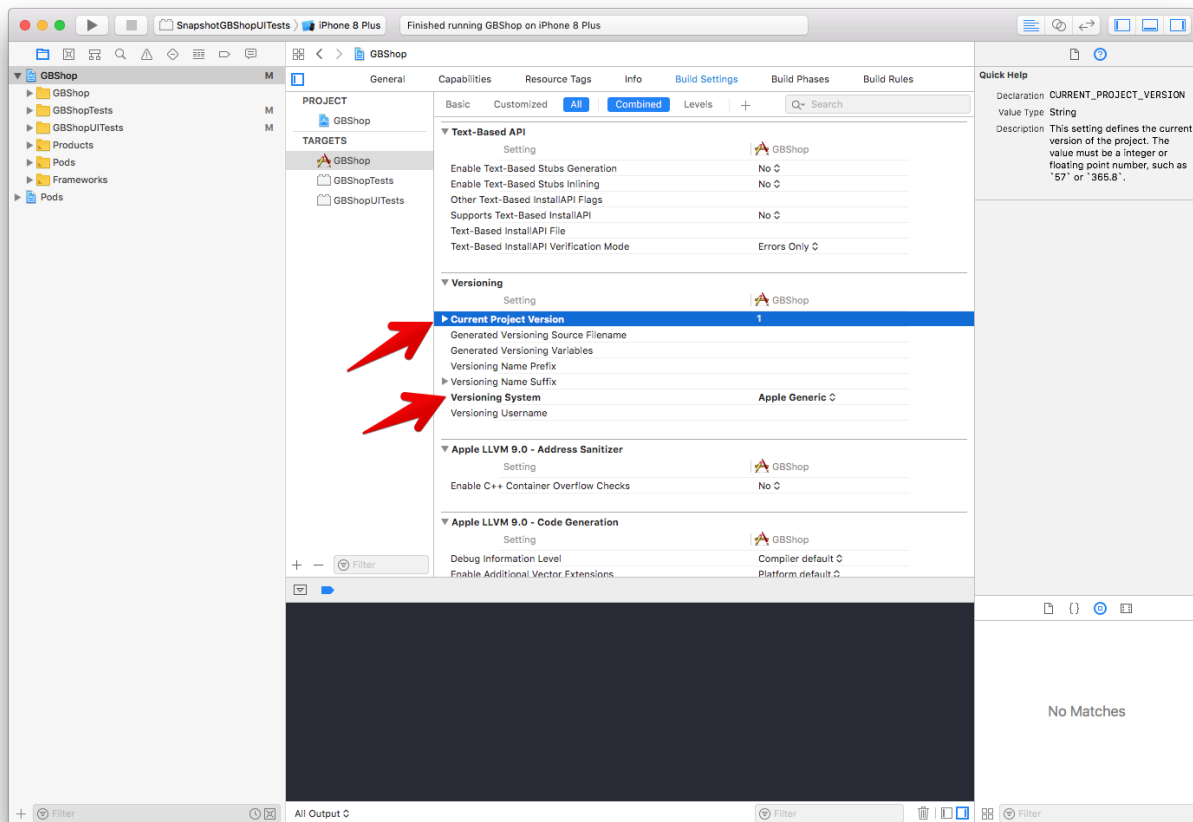
# run tests
run_tests

# make screenshots
capture_ios_screenshots
end

lane :project do
  increment_build_number
end
end

```

Чтобы это действие успешно заработало, донастроим проект в XCode — для цели **GBShop** на вкладке **Build Settings** вставим начальное значение параметра **Current Project Version**, от которого будет идти автоинкремент. Для нашего проекта поставим это значение в **1**. Также проставим параметр **Versioning System** в значение **Apple Generic**.



Выполним команду запуска нового **lane project** в терминале:

```
fastlane ios project
```

И увидим результат ее работы:

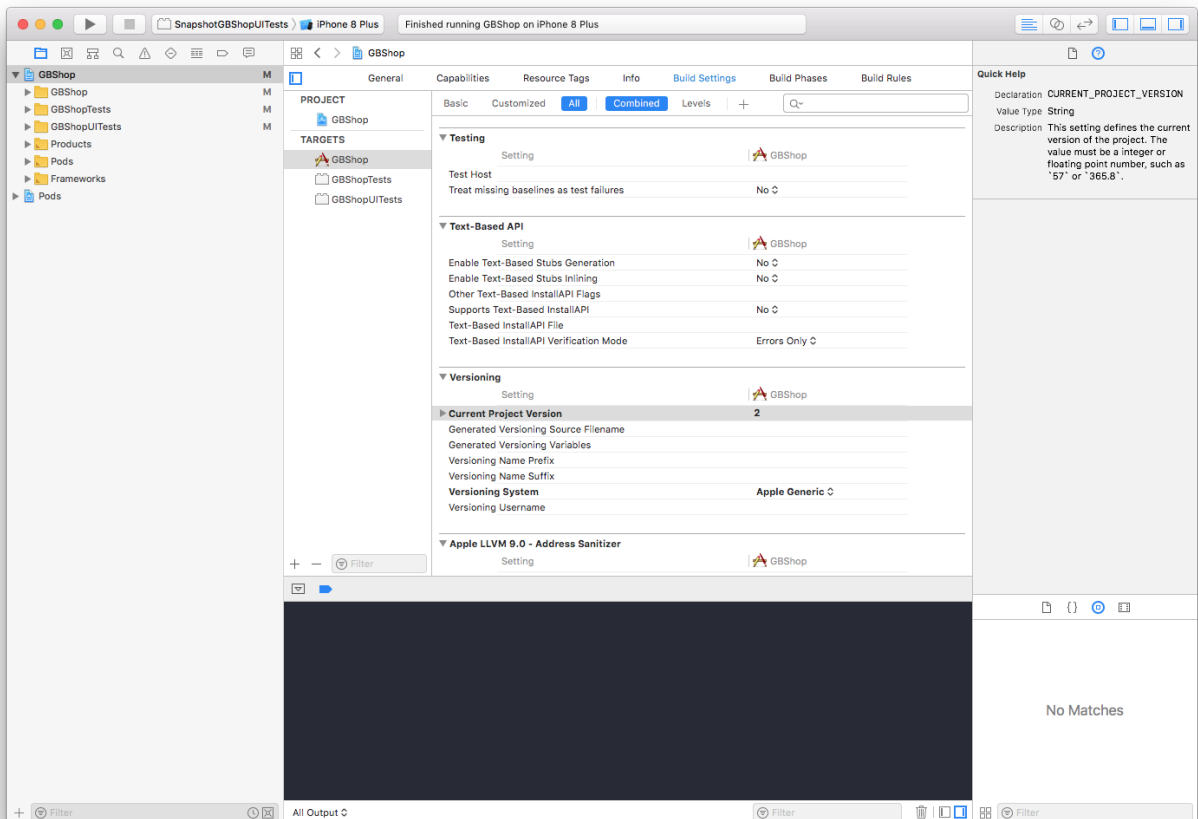
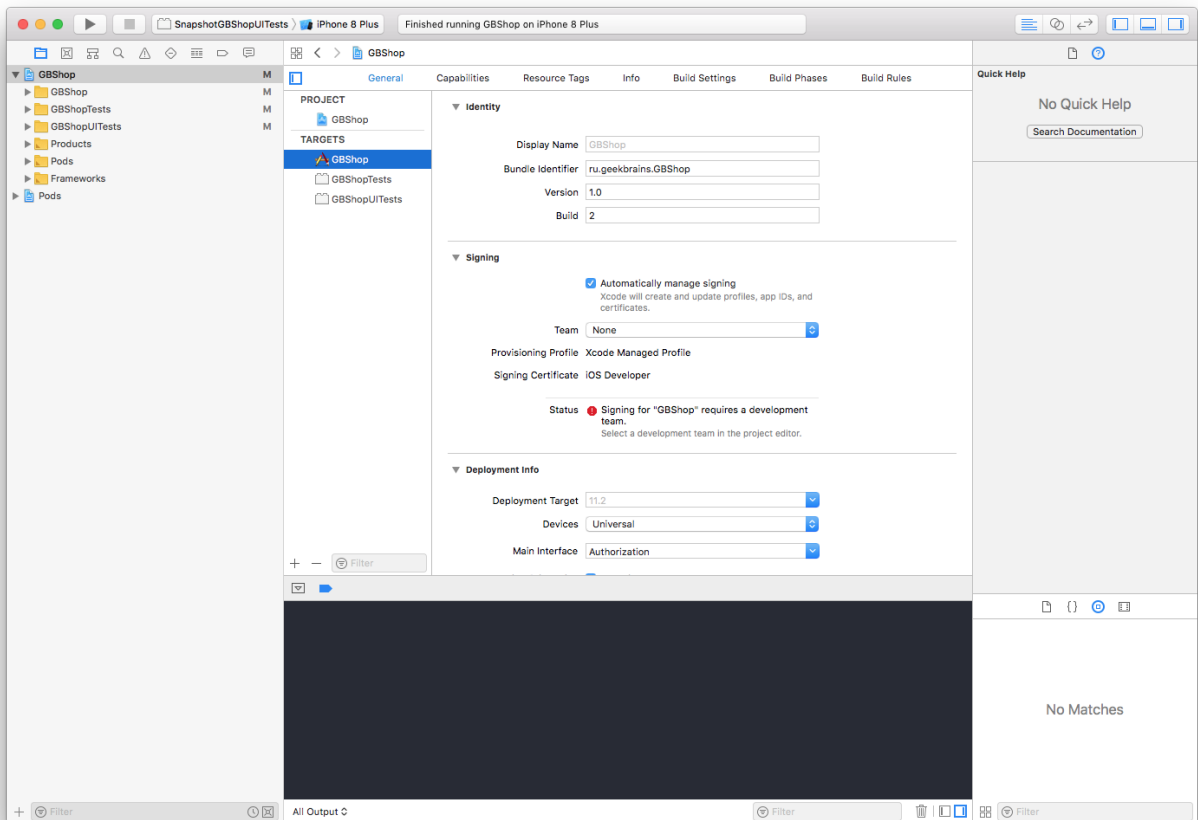
```
mac2:GBShop oivanov$ fastlane ios project
[✓] 🚀
[22:58:35]: fastlane detected a Gemfile in the current directory
[22:58:35]: however it seems like you don't use `bundle exec`
[22:58:35]: to launch fastlane faster, please use
[22:58:35]:
[22:58:35]: $ bundle exec fastlane ios project
[22:58:35]:
[22:58:35]: Get started using a Gemfile for fastlane https://docs.fastlane.tools/getting-started/ios/setup/#use-a-gemfile

[22:58:37]: -----
[22:58:37]: --- Step: default_platform ---
[22:58:37]: -----
[22:58:37]: Driving the lane 'ios project' 🚀
[22:58:37]: -----
[22:58:37]: --- Step: increment_build_number ---
[22:58:37]: -----
Current version of project GBShop is:
1

/Users/oivanov/Documents/GBShop
[22:58:41]: $ cd /Users/oivanov/Documents/GBShop && agvtool next-version -all && cd -
[22:58:41]: ▶ Setting version of project GBShop to:
[22:58:41]: ▶ 2.
[22:58:41]: ▶ Also setting CFBundleVersion key (assuming it exists)
[22:58:41]: ▶ Updating CFBundleVersion in Info.plist(s)...
[22:58:42]: ▶ $(SRCROOT)/GBShop/Environment/Info.plist
[22:58:42]: ▶ Cannot find "$(SRCROOT)/GBShop/Environment/Info.plist"
[22:58:42]: ▶ Updated CFBundleVersion in "GBShop.xcodeproj/../GBShopTests/Info.plist" to 2
[22:58:42]: ▶ Updated CFBundleVersion in "GBShop.xcodeproj/../GBShopUITests/Info.plist" to 2
[22:58:42]: ▶ /Users/oivanov/Documents/GBShop

+-----+-----+
|               fastlane summary               |
+-----+-----+
| Step | Action                  | Time (in s) |
+-----+-----+
| 1    | default_platform        | 0            |
| 2    | increment_build_number   | 5            |
+-----+-----+

[22:58:42]: fastlane.tools finished successfully 🚀
mac2:GBShop oivanov$
```



Этот номер версии также фигурирует и в **Info.plist**. Непосредственно из него можно отредактировать любой параметр. Например, поменять short-версию приложения (**CFBundleShortVersionString**). Добавим изменения в **Fastfile**:

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
    cocoapods(
      use_bundle_exec: false
    )

    # run tests
    run_tests

    # make screenshots
    capture_ios_screenshots
  end

  lane :project do
    increment_build_number
    set_info_plist_value(path: "./GBShop/Environment/Info.plist", key:
"CFBundleShortVersionString", value: "2.0")
  end
end
```

Выполним команду запуска **lane project** в терминале и получим новое значение **CFBundleShortVersionString - 2.0**.

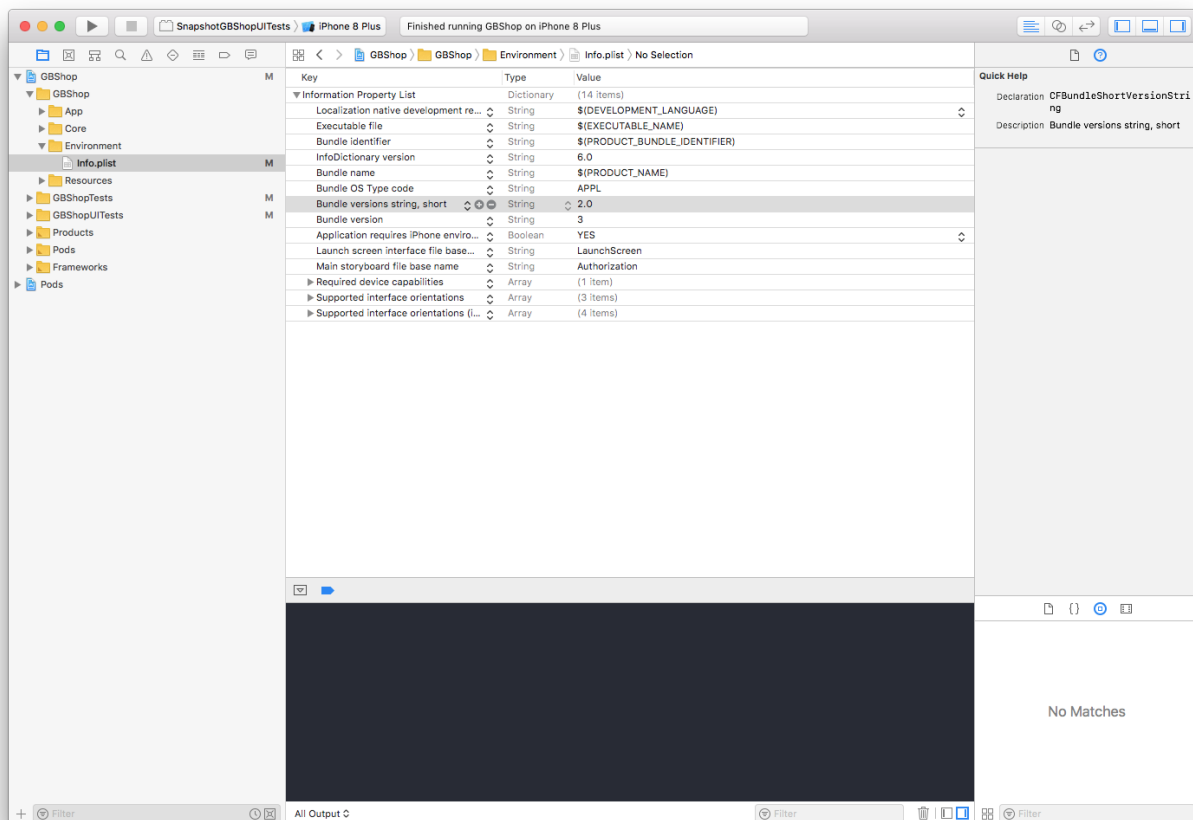
```
GBShop — -bash — 109x45
[23:14:46]: however it seems like you don't use `bundle exec`
[23:14:46]: to launch fastlane faster, please use
[23:14:46]:
[23:14:46]: $ bundle exec fastlane ios project
[23:14:46]:
[23:14:46]: Get started using a Gemfile for fastlane https://docs.fastlane.tools/getting-started/ios/setup/#use-a-gemfile

[23:14:46]: -----
[23:14:46]: --- Step: default_platform ---
[23:14:46]: -----
[23:14:46]: Driving the lane 'ios project' 🚀
[23:14:46]: -----
[23:14:46]: --- Step: increment_build_number ---
[23:14:46]: -----
Current version of project GBShop is:
2

/Users/oivanov/Documents/GBShop
[23:14:47]: $ cd /Users/oivanov/Documents/GBShop && agvtool next-version -all && cd -
[23:14:47]: ▶ Setting version of project GBShop to:
[23:14:47]: ▶ 3.
[23:14:47]: ▶ Also setting CFBundleVersion key (assuming it exists)
[23:14:47]: ▶ Updating CFBundleVersion in Info.plist(s)...
[23:14:47]: ▶ $(SRCROOT)/GBShop/Environment/Info.plist
[23:14:47]: ▶ Cannot find "$(SRCROOT)/GBShop/Environment/Info.plist"
[23:14:47]: ▶ Updated CFBundleVersion in "GBShop.xcodeproj/../GBShopTests/Info.plist" to 3
[23:14:47]: ▶ Updated CFBundleVersion in "GBShop.xcodeproj/../GBShopUITests/Info.plist" to 3
[23:14:47]: ▶ /Users/oivanov/Documents/GBShop
[23:14:47]: -----
[23:14:47]: --- Step: set_info_plist_value ---
[23:14:47]: -----

+-----+-----+
|               fastlane summary               |
+-----+-----+
| Step | Action                  | Time (in s) |
+-----+-----+
| 1    | default_platform        | 0            |
| 2    | increment_build_number   | 1            |
| 3    | set_info_plist_value     | 0            |
+-----+-----+

[23:14:47]: fastlane.tools finished successfully 🎉
mac2:GBShop oivanov$
```



Пришло время собрать проект **GBShop** с помощью действия **gym**. Добавим его со своими параметрами в файл **Fastfile**. Основные параметры:

- уже знакомый нам **scheme** — зададим в «**GBShop**»;
- **configuration** — конфигурация, используемая при создании приложения. Зададим в «**Debug**»;
- **export_method** — метод, используемый для экспорта архива. Зададим в «**development**».

Все параметры описаны в документации к Fastlane. Например, для gym — <https://docs.fastlane.tools/actions/gym>.

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
    cocoapods (
```

```
        use_bundle_exec: false
    )

    # run tests
    run_tests

    # make screenshots
    capture_ios_screenshots
end

lane :project do
    increment_build_number
    set_info_plist_value(path: "./GBShop/Environment/Info.plist", key:
"CFBundleShortVersionString", value: "2.0")

    gym(
        scheme: "GBShop",
        configuration: "Debug",
        export_method: "development",
        include_symbols: true,
        include_bitcode: true,
        silent: false
    )
end
end
```

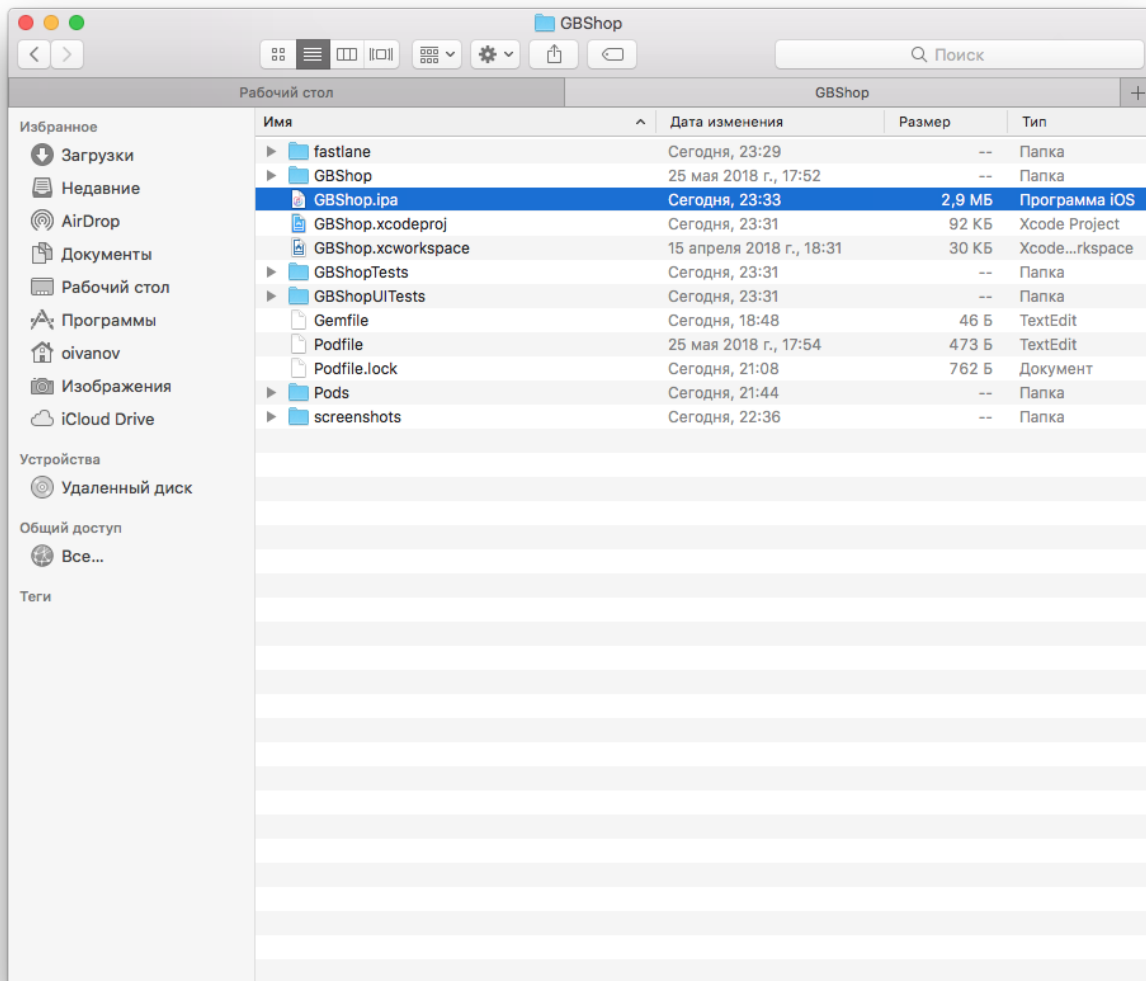


```
GBShop — -bash — 109x45
[23:31:32]: > Compiling MessageResponse.swift
[23:31:32]: > Compiling SettingsViewController.swift
[23:31:33]: > Compiling GBShop_vers.c
[23:31:33]: > Linking GBShop
[23:31:33]: > Compiling Authorization.storyboard
[23:31:37]: > Compiling ProductList.storyboard
[23:31:37]: > Compiling Main.storyboard
[23:31:37]: > Compiling Basket.storyboard
[23:31:37]: > Compiling Settings.storyboard
[23:31:38]: > Compiling ProductDescription.storyboard
[23:31:39]: > Compiling LaunchScreen.storyboard
[23:31:39]: > Processing Info.plist
[23:31:39]: > Running script '[CP] Embed Pods Frameworks'
[23:31:40]: > Touching GBShop.app
[23:31:51]: > Signing /Users/oivanov/Library/Developer/Xcode/DerivedData/GBShop-aywvflckbcxkxhxdotfikhogoaprm/
Build/Intermediates.noindex/ArchiveIntermediates/GBShop/InstallationBuildProductsLocation/Applications/GBShop
.app
[23:32:03]: > Archive Succeeded
[23:32:04]: Generated plist file with the following values:
[23:32:04]: > -----
[23:32:04]: > {
[23:32:04]: >   "method": "development"
[23:32:04]: > }
[23:32:04]: > -----
[23:32:04]: $ /usr/bin/xcrun /Library/Ruby/Gems/2.3.0/gems/fastlane-2.96.1/gym/lib/assets/wrap_xcodebuild/xcbo
uild-safe.sh -exportArchive -exportOptionsPlist '/var/folders/ww/v0726xq17rn7h804ynwh0px40000gp/T/gym_config2
0180530-60425-1ia8zor.plist' -archivePath /Users/oivanov/Library/Developer/Xcode/Archives/2018-05-30/GBShop\
2018-05-30\ 23.31.19.xcarchive -exportPath '/var/folders/ww/v0726xq17rn7h804ynwh0px40000gp/T/gym_output201805
30-60425-18cvqz4'
[23:33:21]: Successfully exported and signed the ipa file:
[23:33:21]: /Users/oivanov/Documents/GBShop/GBShop.ipa

+-----+-----+-----+
|                               |
|               fastlane summary                               |
|                               |
+-----+-----+-----+
| Step | Action                | Time (in s) |
+-----+-----+-----+
| 1    | default_platform      | 0            |
| 2    | increment_build_number | 1            |
| 3    | set_info_plist_value  | 0            |
| 4    | gym                   | 124          |
+-----+-----+-----+

[23:33:21]: fastlane.tools finished successfully 🎉
mac2:GBShop oivanov$
```

В корне папки проекта видим созданный файл **GBShop.ipa**.



После успешной сборки проекта и прогонки всех тестов перед публикацией в App Store можно сделать **tag** в **git**. Для этого выполним действия в **Fastfile**:

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :custom_lane do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
    xcclean(
      scheme: "GBShop",
      workspace: "GBShop.xcworkspace"
    )

    # update a dependencies
    cocoapods(
      use_bundle_exec: false
    )
  end
end
```

```

# run tests
run_tests

# make screenshots
capture_ios_screenshots
end

lane :project do
  increment_build_number
  set_info_plist_value(path: "./GBShop/Environment/Info.plist", key:
"CFBundleShortVersionString", value: "3.0")

  gym(
    scheme: "GBShop",
    configuration: "Debug",
    export_method: "development",
    include_symbols: true,
    include_bitcode: true,
    silent: false
  )

  # commit changes to git
  versionNumber = get_version_number
  buildNumber = get_build_number
  tag = "#{versionNumber}.#{buildNumber}"

  # git_add
  git_add(path: ["../Podfile.lock", "./GBShop.xcodeproj/project.pbxproj",
"./GBShopTests/Info.plist", "./GBShop/Environment/Info.plist",
"./GBShopUITests/Info.plist"])
  git_commit(path: ["../Podfile.lock", "./GBShop.xcodeproj/project.pbxproj",
"./GBShopTests/Info.plist", "./GBShop/Environment/Info.plist",
"./GBShopUITests/Info.plist"], message: "verson up")
  add_git_tag(
    tag: tag
  )

end
end

```

Выполним команду запуска **lane project** в терминале и увидим успешное создание **tag** в **git**.

```
GBShop — -bash — 109x45
[23:56:52]: --- Step: get_build_number ---
[23:56:52]: -----
[23:56:52]: $ cd /Users/oivanov/Documents/GBShop && agvtool what-version -terse
[23:56:52]: ▶ 5
[23:56:52]: -----
[23:56:52]: --- Step: git_add ---
[23:56:52]: -----
[23:56:52]: Successfully added all files 📁.
[23:56:52]: -----
[23:56:52]: --- Step: git_add ---
[23:56:52]: -----
[23:56:52]: Successfully added ".Podfile.lock ./GBShop.xcodeproj/project.pbxproj ./GBShopTests/Info.plist ./
GBShop/Environment/Info.plist ./GBShopUITests/Info.plist" 📁.
[23:56:52]: -----
[23:56:52]: --- Step: git_commit ---
[23:56:52]: -----
[23:56:52]: $ git commit -m version\ up ./Podfile.lock ./GBShop.xcodeproj/project.pbxproj ./GBShopTests/Info.p
list ./GBShop/Environment/Info.plist ./GBShopUITests/Info.plist
[23:56:52]: ▶ warning: unable to access '/Users/oivanov/.config/git/attributes': Permission denied
[23:56:52]: ▶ [master 243840a] version up
[23:56:52]: ▶ 1 file changed, 1 insertion(+), 1 deletion(-)
[23:56:52]: Successfully committed "[\"./Podfile.lock\", \"./GBShop.xcodeproj/project.pbxproj\", \"./GBShopTests/I
nfo.plist\", \"./GBShop/Environment/Info.plist\", \"./GBShopUITests/Info.plist\"]" 📁.
[23:56:52]: -----
[23:56:52]: --- Step: add_git_tag ---
[23:56:52]: -----
[23:56:52]: Adding git tag '3.0.5' 🏷️.
[23:56:52]: $ git tag -am 3.0.5\ \"(fastlane\)' '3.0.5'

+-----+-----+-----+
|               fastlane summary               |
+-----+-----+-----+
| Step | Action                | Time (in s) |
+-----+-----+-----+
| 1    | default_platform      | 0            |
| 2    | get_version_number    | 0            |
| 3    | get_build_number      | 0            |
| 4    | git_add                | 0            |
| 5    | git_add                | 0            |
| 6    | git_commit             | 0            |
| 7    | add_git_tag            | 0            |
+-----+-----+-----+

[23:56:52]: fastlane.tools finished successfully 🎉
mac2:GBShop oivanov$
```

Осталось выгрузить проект в **testflight** iTunes Connect, используя действие **upload_to_testflight**.

```
upload_to_testflight(
  username: "felix@krausefx.com",
  app_identifier: "com.krausefx.app",
  itc_provider: "abcde12345"
)
```

Объединим **lanes** «**project**» и «**custom_lane**» в один **project** и получим результирующий файл работы с **Fastlane**:

```
default_platform(:ios)

platform :ios do
  desc "Description of what the lane does"
  lane :project do

    # checkout release branch
    #ensure_git_status_clean

    # clear data
    clear_derived_data
```

```

xcclean(
  scheme: "GBShop",
  workspace: "GBShop.xcworkspace"
)

# update a dependencies
cocoapods(
  use_bundle_exec: false
)

# run tests
run_tests

# make screenshots
capture_ios_screenshots

increment_build_number
set_info_plist_value(path: "./GBShop/Environment/Info.plist", key:
"CFBundleShortVersionString", value: "2.0")

gym(
  scheme: "GBShop",
  configuration: "Debug",
  export_method: "development",
  include_symbols: true,
  include_bitcode: true,
  silent: false
)

# commit changes to git
versionNumber = get_version_number
buildNumber = get_build_number
tag = "#{versionNumber}.#{buildNumber}"

# git_add
git_add(path: [".Podfile.lock", "./GBShop.xcodeproj/project.pbxproj",
"./GBShopTests/Info.plist", "./GBShop/Environment/Info.plist",
"./GBShopUITests/Info.plist"])
git_commit(path: [".Podfile.lock", "./GBShop.xcodeproj/project.pbxproj",
"./GBShopTests/Info.plist", "./GBShop/Environment/Info.plist",
"./GBShopUITests/Info.plist"], message: "verson up")
add_git_tag(
  tag: tag
)

upload_to_testflight(
  username: "asdasd@asdasd.com",
  app_identifier: "com.32eref.app",
  itc_provider: "shjdghsagfhdsdfhg"
)
end
end

```

Подобных **lanes** можно создавать множество для разных нужд — релиза для разработки, показа заказчику и других.

Практическое задание

1. Автоматизировать сборку и запуск тестов проекта.

Дополнительные материалы

1. [fastlane](#).

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Стив Макконнелл. Совершенный код.