Marcin Rabieko
iOS Developer

23.01.2020

XCode Templates tutorial – How to create custom template step by step

Improving work processes is always on a programmer's mind. We're on the lookout for tools and solutions to speed up the coding, testing or organising our work. I find using templates for repetitive tasks a very good way to make my work more efficient. Today I'd like to show you how I make **XCode Templates** work for me. I hope this tutorial will be helpful in improving your daily tasks as well.

Table of contents

What are XCode Templates?

XCode Templates is a tool for creating code snippets to give you a better starting point to achieve you goal. In this tutorial I will walk you through preparing a custom template for MVVM project architecture

Often we need to create from scratch the structure and files for a new module, and this process is pretty much the same each time. For example – in the MVVM pattern, to create a Login module we ne to create folders and at least 4 classes:

1. LoginModule Folder
2. LoginView
3. LoginViewModel
4. LoginViewController
5. LoginModel

Adding each class using required code is time consuming. With XCode templates we can speed up th process of adding them to our project. I will show you how to configure a template for use with a new MVVM module.

XCode Templates Installation

To install templates in XCode we need to add a new folder which will contain our custom templates.

XCode iOS Templates location

All the Xcode custom template files are located in **~/Library/Developer/Xcode/Templates/** and grouped into sections by folder name. You can add it manually or using the terminal by running the following command:

mkdir ~/Library/Developer/Xcode/Templates/Custom Templates

File template structure

Template main folder

Each XCode file template is a separate folder with the extension .xctemplate. If you want to create a template named “View, Model & ViewModel”, you have to create a folder named **“View, Model & ViewModel.xctemplate”** in **~/Library/Developer/Xcode/Templates/File Templates/Custom Templat**

Template internal folders and files

The TemplateInfo.plist file contains basic template description. Implementation below gives us the ability to type **Module name** while creating XCode app templates with the wizard.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0">
<dict>
        <key>Kind</key>
        <string>Xcode.IDEKit.TextSubstitutionFileTemplateKind</string>
        <key>Platforms</key>
        <array>
                <string>com.apple.platform.iphoneos</string>
        </array>
        <key>Options</key>
        <array>
                <dict>
                        <key>Identifier</key>
                        <string>productName</string>
                        <key>Required</key>
                        <true/>
                        <key>Name</key>
<string>Module Name</string>
                        <key>Description</key>
                        <string>The name of the Model, View and ViewModel to create</string>
                        <key>Type</key>
                        <string>text</string>
                        <key>Default</key>
                        <string>Module1</string>
                </dict>
        </array>
</dict>
</plist>
```

You can put pretty much anything into the actual template file. You can use text macros like ___FILEBASENAME___ to reference the filename. The name is derived from the productName option fr our TemplateInfo.info file which is set in new file wizard.

Implementation and structure

My template's folder structure, and Swift classes look like this:

Step 1. Let's start from the top. Here's the ViewController implementation:

```
import UIKit

class ___FILEBASENAMEASIDENTIFIER___: UIViewController {

    let viewModel: ___VARIABLE_productName___ViewModel
    let mainView: ___VARIABLE_productName___View

    init() {
        viewModel = ___VARIABLE_productName___ViewModel(withModel: ___VARIABLE_productName___())
        mainView = ___VARIABLE_productName___View()
        super.init(nibName: nil, bundle: nil)

        mainView.configure(withViewModel: viewModel)
    }

    required init?(coder _: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        setupView()
    }

    private func setupView() {
        view.addSubview(mainView)
        mainView.snp.makeConstraints { make in
            make.top.leading.trailing.bottom.equalToSuperview()
        }
    }
}
```

As you can see we implemented the following:

- declaration of viewModel and mainView variables. It will contain our productName from TemplateInfo.plist set in file wizard
- initialization of above variables
- default initializer
- required initializer (we use ViewControllers from code, not Storyboards)
- viewDidLoad implementation
- setupView function adding mainView and set SnapKit constraints

Step 2. Model implementation:

```
import Foundation

class ___FILEBASENAMEASIDENTIFIER___ {

}
```

The model class name will be generated automatically by XCode so the above is just an example.

Step 3. View implementation:

```
import UIKit

class ___FILEBASENAMEASIDENTIFIER___: UIView {

    init() {
        super.init(frame: CGRect.zero)
    }

    required init?(coder _: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    func configure(with viewModel: ___VARIABLE_productName:identifier___ViewModel) {
        // configure the view with a ___VARIABLE_productName:identifier___ViewModel
    }
}
```

Class view

Class view contains a default initializer, and a required initializer – both required to initialize the View from the code. We also need to configure function to bind viewModel and the view. Note that ViewMod name is the same as the ViewModel name in the ViewController

Step 4. ViewModel implementation:

```
import Foundation

class ___FILEBASENAMEASIDENTIFIER___ {

    private let model: ___VARIABLE_productName:identifier___
```

```
    init(withModel model: ___VARIABLE_productName:identifier___) {
        self.model = model
    }
}
```

ViewModel is initialized with our Model created in p.2.

Using file templates in XCode

To get started You just need to click File -> New -> File...

Find your template in the list.

Then type the name of the module.

And add it to your project.

Advanced options in Xcode templates

Some example file templates demonstrating some of the options available can be found on GitHub . You can also reverse engineer Xcode default templates and see how to improve custom templates fo better use. They are located in Xcode.app/Contents/Developer/Library/Xcode/Templates or Xcode.app/Contents/Developer/Platforms/<platform_name>/Developer/Library/Xcode/Templates.

Share XCode project templates

In order to share your template with others you need to send all of the template files. For someone els to be able to use it they need to add the template into the following folder:.
~/Library/Developer/Xcode/Templates/File Templates/Custom Templates

and name it accordingly. Now the template is available in XCode.

Summary

This is a full working XCode template to speed up your coding. You can change any part of the code t your needs. Here is a help tip for text macro references:

TemplateInfo.plist can be more complex, you can eg. use the wizard to add a selection list to pick files type You can also create your own default implementation and structure for new projects and targets

type.You can also create your own default implementation and structure for new projects and targets

5 (100%) 12 vote[s]

Trending

21.05.2020

Introduction to cross-platform app development

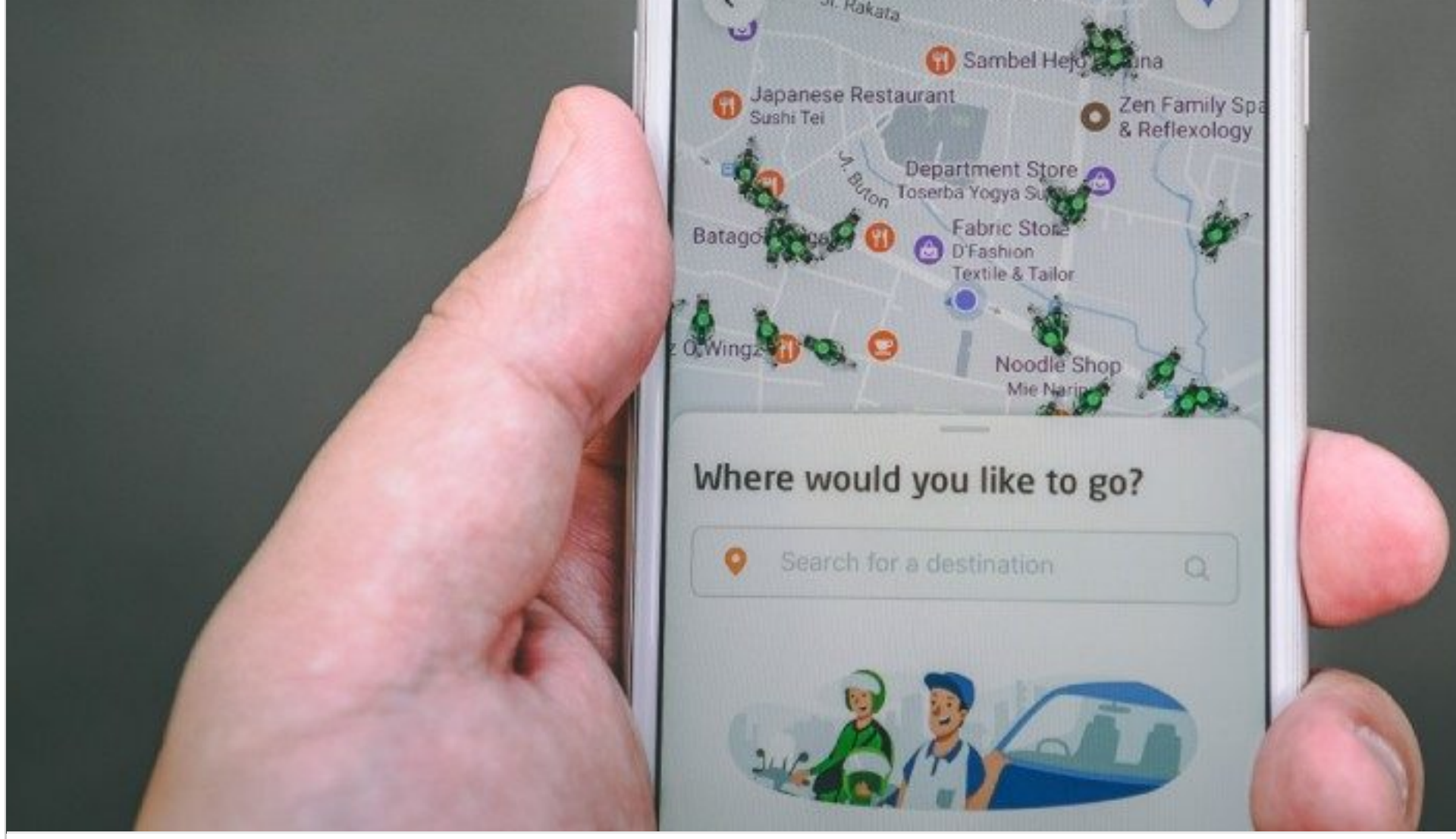
16.05.2020

MVP app development – Lean and agile way to develop anything

11.09.2019

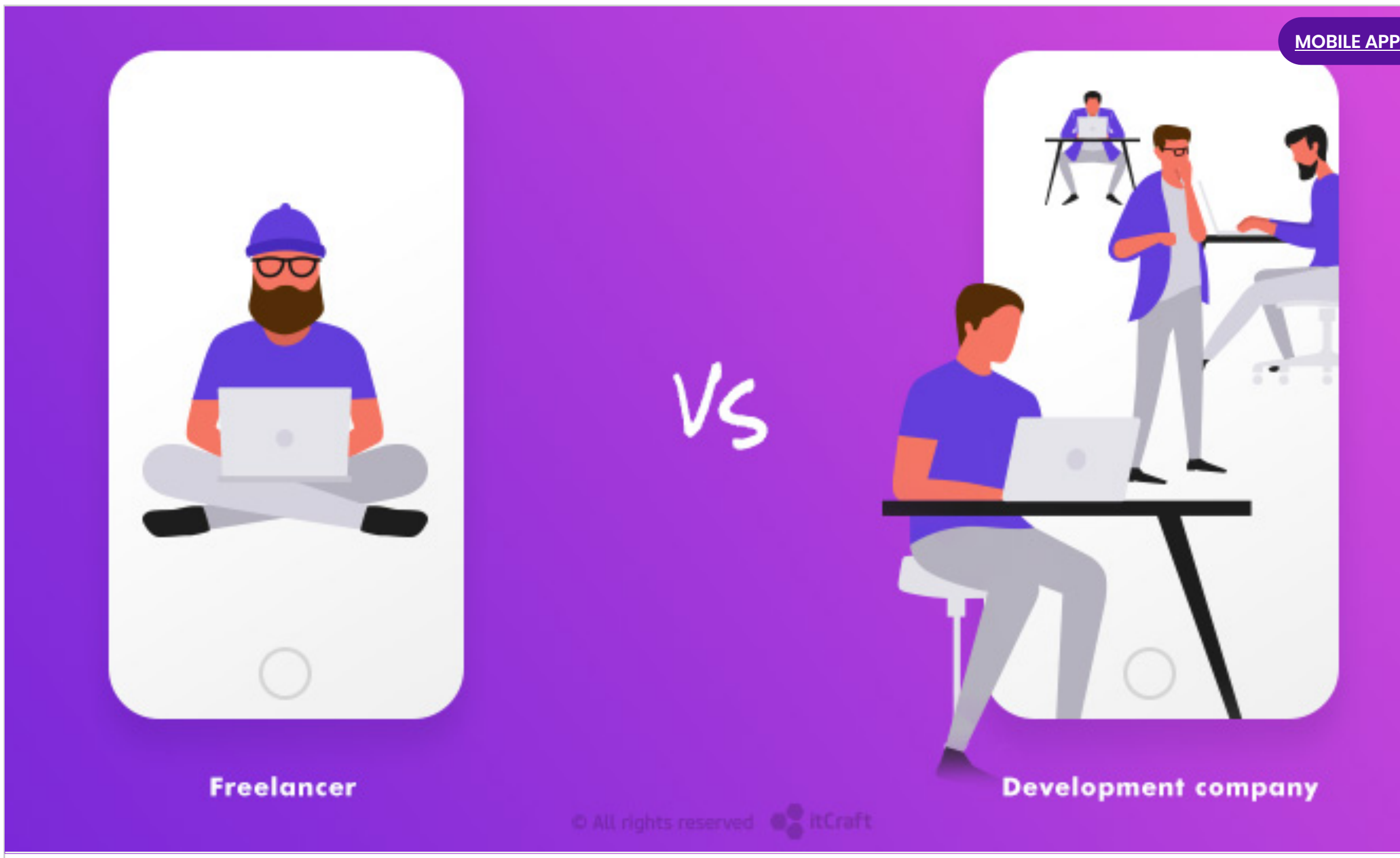
Product Owner's guide to Agile Scrum Pt.2 – Single point of contact

You may also like



9.04.2020

How much does it cost to make a booking type app?



7.12.2018

Freelancer vs Software Development Company

5.07.2019

7 React Native myths vs reality – what we (think we) know

Start a Project with itCraft

We operate global. Our main headquarters are based in **Warsaw, Bydgoszcz, Toruń and Gdańsk** in Poland but we can arrange meeting in **New York or in London**. We are also present in **Norway**. Are you from another country – USA, United Kingdom, German Emirates? Don't worry. Give us your idea for the application and we will make it reality. Below you will find all the contact details yo need.

ESTIMATE PROJECT

NEW YORK

347 Fifth Avenue
Suite 1402-118
New York, NY 10016

BYDGOSZCZ

LONDON

London (WC2) Office
7 Bell Yard
London WC2A 2JR

STOCKHOLM

ul. Towarowa 20, first floor
85-746 Bydgoszcz,
Poland

GDAŃSK

GPNT
ul. Trzy Lipy 3,
80-172 Gdańsk

Gustavslundsvägen 12
167 51 Bromma
Stockholm, Sweden

WARSAW

Brain Embassy
Aleje Jerozolimskie 181B
02-222 Warszawa

itCraft

+1 646 630 9053 hello@itcraftapps.com