Hello, my name is Ivan. Today I would like to present you my application that is loosely connected with the tourism theme. Travelling is great and I used to travel by car, train, and of course aircrafts.

Aviation is some kind of incredible kind of transportation starting with its extremely heavy carbon footprint and ending with some extraordinary events such as one happened months ago. As far as my college was an airport grunt and my friend was learning piloting aircraft, I came to the idea to create some king of flight tracking application.

It should resemble both FlightRadar24 with its ability to track flights in real time with opportunity to select cheapest and reliable options to move from one city to another such as offered by Yandex.Avia or Aviasales. The application should bring some kind of general information about companies, lead to its pages. It would be great if could get to know more about aircraft, its age and technical features such as it was done in Task 11 with rockets.

Now let’s get closer to technology stack.

The initial idea of the project was making it using SwiftUI and Combine using MVVM architecture in order to get deal with these brand-new frameworks. I also planned to understand how to manage routing in these frameworks as far as in UIKit I prefer using MVVM or MVP together with coordinators.

However, due to the reason SwiftUI was banned I returned to more reliable approach with commonly known UIKit and famous and the only Combine’s real competitor – RxSwift.

So, the technology stack of the application will be UIKit+RxSwift using MVVM-C architecture solution. I use coordinators in order to kill two birds with one stone: I want to get rid of singletons as a harmful for testing approach (I inject services using coordinators), as well as to solve so-called “assembly problem” as far as I do not like to assemble new screens and perform their navigation inside ViewController. I consider that it decreases potential of reusability of the screens.

Here is the time to discuss design.

As far as initial idea was SwiftUI and I planned to check it predictably weak sides such as poor customizability I figured out neuromorphic user interface that was in fact hard to be implemented in UIKit what I noticed implementing Task 11. Here you can see the initial idea.

However, it was some kind of procrastination to use the same design in UIKit as far as we dealt with it. When I was providing design in Figma I accidently added stroke for a piece of text. It looked slightly ludicrous but in general it was an icing on the cake after all these tasks. It was really hard to create something brand new. *{tnx Vlad}*

As benefit of work with this design, I learned more about NSAttributedString and its keys such as kern and stroke. It also provided easy way to enable dark mode. (However, it required traitCollectionDidChange in some places)

The source of data was a real compilation of free and freemium API’s. { Here they are} I should say that their quality leaves much to be desired. For example, Google Flights work really slowly. Aviationstack rarely provides information about aircraft and live date (which was planned to be key part of my project!!).

So, the strongest part, in my opinion, is not in functionality, but in code). RxSwift is great idea to bring more functional programming in routine object-oriented. ViewModels, Coordinators, Services are covered by protocols. All literals such as numbers and strings are put away in Constants enumerations. It resolves “magic numbers” problem and provides easy way to potentially localize you application.

So, I hope you enjoyed the presentation and now let’s get closer to demo.