

SwiftUI в Production

Смелый выбор?

Testing the waters

Обо мне

Bekzat, iOS разработчик

- SDU, 2012 - 16
- Kolesa, 2016 - 17
- Booking, 2017 - now

(Да, это компания, где работают ~20 казахов в одном офисе :)

Flights iOS Team

- 3 iOS devs
- Starting from **April 2020**
- 85 files, 7.5k LOC* in UI module
- Native Screens + WebView (iOS 13 & iOS 12)
- Using MVVM
- Live Previews + Screenshot tests
- Support for deep links



Global trends

Declarative UI

vs Imperative

ReactNative

ComponentKit

Flutter

LayoutKit

UIKit

Why SwiftUI ?

- Внутренний native declarative framework на подобии SwiftUI, который начался разрабатываться с 2018 года (еще до анонса на WWDC)
- Достаточно высокий adoption iOS 13 к началу апреля > 90%
- Традиционная поддержка iOS n - 2, n - 1 и возможный drop iOS 12 к концу года
- Fallback на webview для iOS 12

*Только наша команда экспериментриует с SwiftUI

Declarative: Saying what you want

Imperative: Saying how to achieve it

Imperative (HOW): “I see that table located under that big picture is empty. Me and my wife are going to walk over there and sit down.”

Declarative (WHAT): “Table for two, please.”

Imperative: Go out of the north exit of the parking lot and take a left. Get on A-10 North until you get to the 12th street exit. Take a right off the exit like you’re going to Ikea. Go straight and take a right at the first light. Continue through the next light then take your next left. My house is 90.

Declarative: My address is Cairostraat 90, Purmerend, Netherlands, 1448PC

Достаточно близок к SwiftUI + полный контроль над кодом

```
func render() -> Widget {  
    return HStackWidget {  
  
        SizeWidget(  
            width: 32,  
            height: 32,  
            child: OverlayWidget(  
                background: ImageWidget(image: item.icon, contentMode: .scaleAspectFill),  
                alignment: .center,  
                child: child  
            )  
        )  
  
        VStackWidget {  
            TextWidget(item.name, font: .presetStrong)  
            TextWidget(item.details, color: .themeGrayscale)  
        }  
    }  
}
```


Но нам не хватало, целого ряда фич, которые были доступны в SwiftUI

UIKit better interoperability*

Animations*

ViewModifiers

Geometry Reader*

Environments

Preferences

Problems with UIKit

- Heterogeneous components: views vs cells vs view controllers.
- Eg: some component is implemented as UITableViewCell, but you need to use it in the UIStackView or vice versa
- you cannot simply reuse it - you need to make a wrapper cell or make some refactoring to extract UIView out of the cell.

vs

- Everything is a View. Reuse everywhere.
- Move code out easily, before view becomes too massive.
- Encapsulate again and again

Problems with UIKit


UIKit - shared mutable non-observable objects. Can be mutated in several places. Hard to track.

vs

Pure functions operating on immutable data structures. Easier to reason

Everything is function

UI = f(state)



```
func a(_ x: Bool) -> UIView {
  if x {
    return b()
  } else {
    let d = { UILabel() }
    return c(d)
  }
}

func b() -> UIView {
  return a(false)
}

func c(_ f: () -> UIView) -> UIView {
  return UIStackView(arrangedSubviews: [f(), f(), f()])
}
```

Input: a(true)

```
- a(true)
  - b()
    - a(false)
      - c(d)
        - d()
        - d()
        - d()
```

Output:

```
- UIStackView
  - UILabel
  - UILabel
  - UILabel
```

Problems we knew in SwiftUI

- Missing CollectionView, UITextView
- 3rd Party Libraries like ASCollectionView / fallback to UIKit / own implementation
- Broken Navigation ?
- Missing customization in List
- equivalent to viewDidLoad

Быстрый начальный порог входа*

Почти всё можно реализовать с базовыми знаниями

Как side effect:
Везде начинаешь видеть V, H, Z stacks

VStack + HStack + ZStack &

Text + Image + Color + Shape &

.overlay() .background() .frame()

@State, @Bindings, @ObservableObject, @EnvironmentObject

SwiftUI создан чтобы упростить базовые вещи

Basic Features

Custom Features

SwiftUI

A navigation controller

A table view

Cell configuration

Label font setup

```
struct ArticleListView: View {  
    let articles: [Article]  
  
    var body: some View {  
        NavigationView {  
            List(articles.identified(by: \.id)) { article in  
                VStack(alignment: .leading) {  
                    Text(article.title).font(.headline)  
                    Text(article.preview).font(.subheadline)  
                }  
            }.navigationBarTitle(Text("Articles"))  
        }  
    }  
}
```

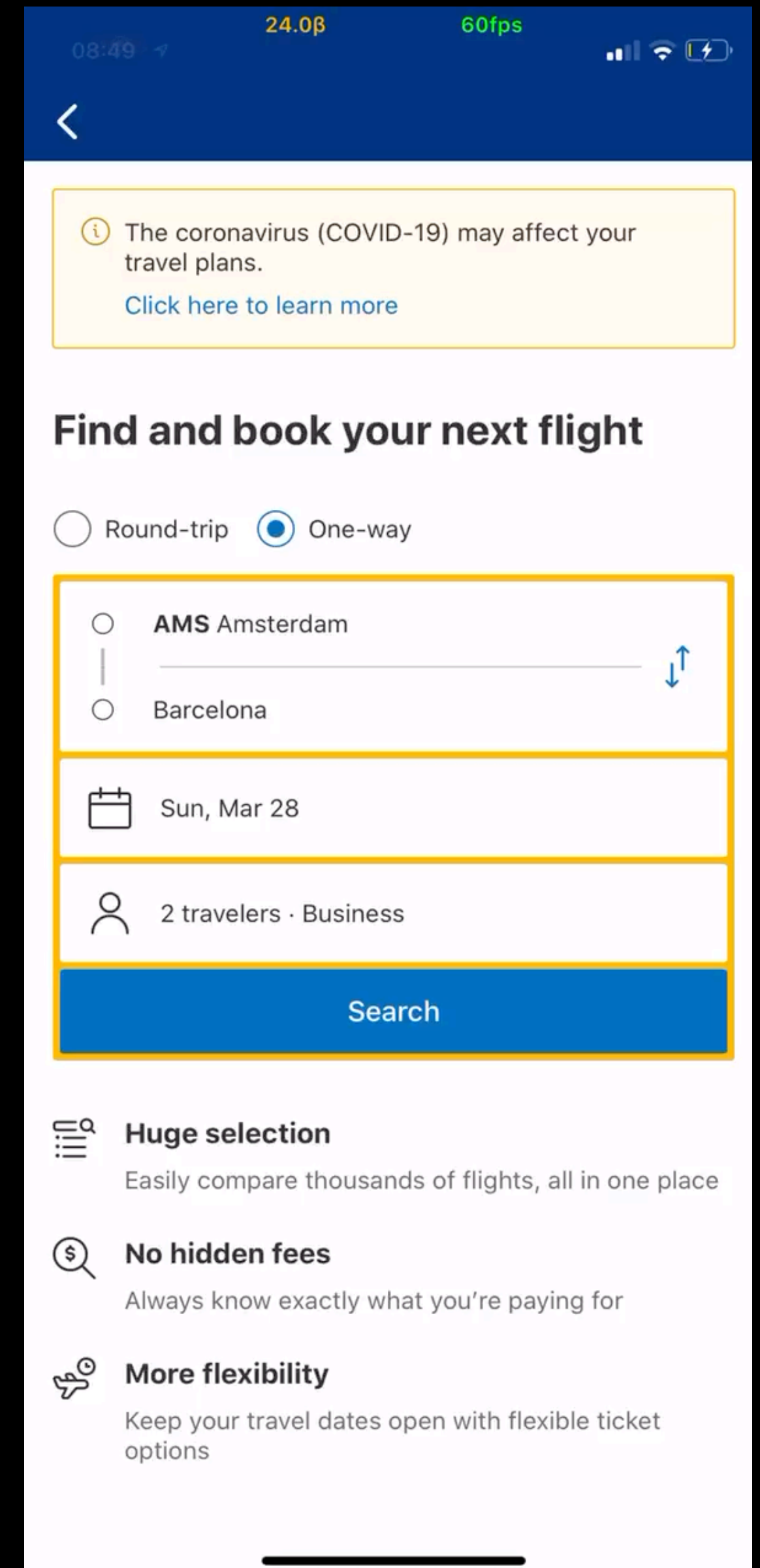

Screen (Обычный View в роли UIViewController)

```
struct SUISearchResultsScreen: View {
    @ObservedObject
    var searchResultsViewModel: SearchResultsViewModel

    let serviceProvider: ServiceProvider

    @State
    private var showSortOptionsBottomSheet: Bool = false

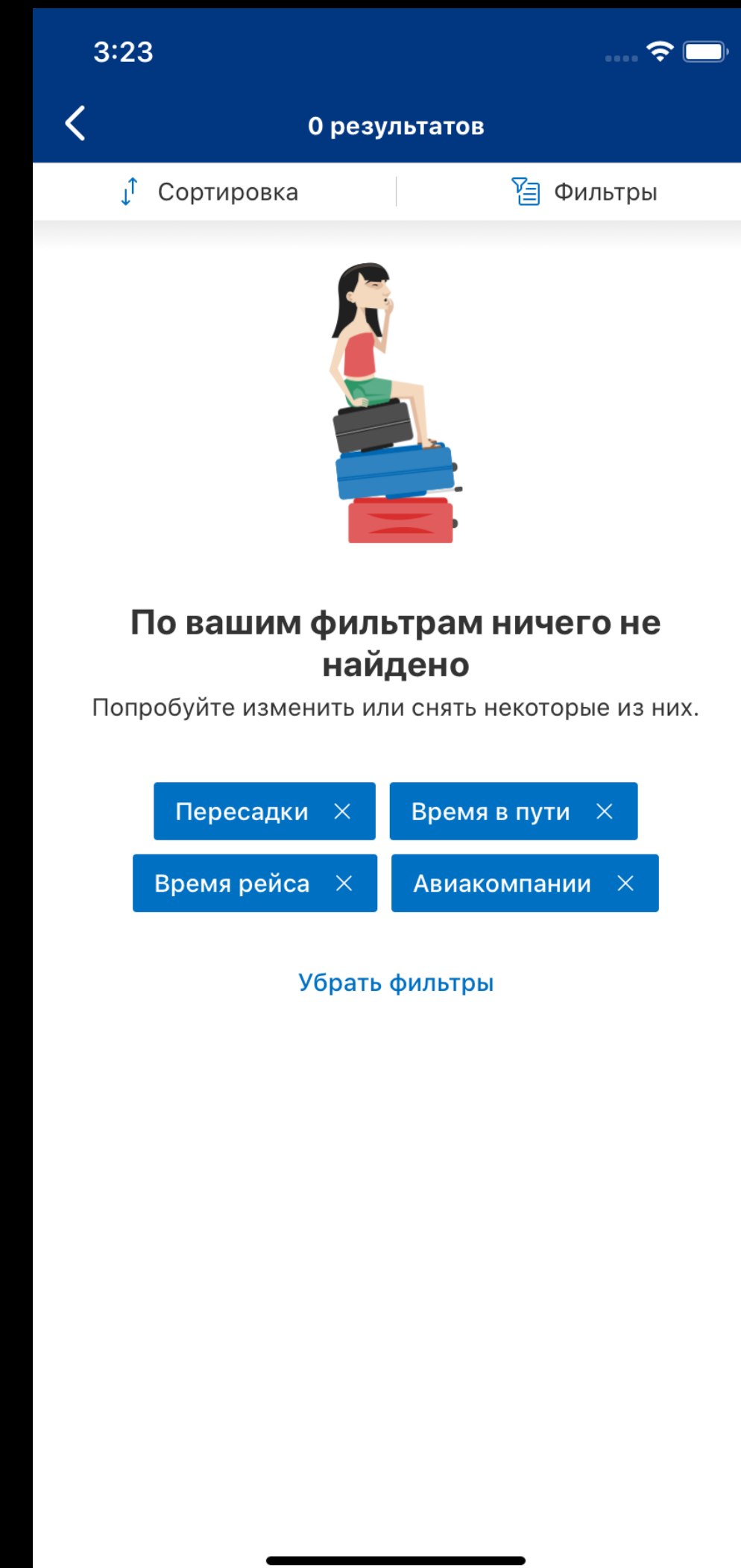
    var body: some View {
        List {
            Section { ... }
            Section {
                ForEach(self.searchResultsViewModel.flightOffers, id: \.token) { flight in
                    SUISearchResultsCard(flight: flight, ... )
                }
            }
            SpinnerView()
                .onAppear {
                    if self.searchResultsViewModel.viewState != .loadingFirst {
                        self.searchResultsViewModel.fetchMore()
                    }
                }
        }.onAppear {
            self.searchResultsViewModel.viewAppear()
        }
    }
}
```



ViewModel

```
class SearchResultsViewModel : ObservableObject {  
  
    private(set) var service: FlightsService  
  
    @Published  
    var viewState: SearchResultsViewState = .loadingFirst  
  
    @Published  
    var searchResults: SearchResults?  
  
    // .. initilizers should be is thin because of NavigationLink non-lazy behavior  
  
    func viewAppear() {  
        if viewState == .loadingFirst {  
            fetchSearchResults(searchType: .newSearch)  
        } else {  
            startInvalidationTimer()  
        }  
    }  
  
    private func fetchSearchResults()// fetch using combine  
}
```

Пока что единственное место где используется
Swift UI - more advanced technique: **PreferenceKey**
для передачи информации от child к parent



Live Previews + Testing strategy

1. Mocking data for previews
 2. Using live previews
 3. Reusing them in Snapshot tests. Profit
-
1. Unit testing: currently possible only for ViewModels
 2. SwiftUI view have deterministic states? :) But view tree is black box :(

PreviewProvider

```
struct SUIIndexScreen_Previews: PreviewProvider {
    static var previews: some View {
        let service = FlightsService(apiClient: FlightAPIMock(), deviceIdentifier: "Preview_Device", locale: "en-us")
        let searchBoxViewModel = SearchBoxViewModel(model: SearchOptionsModel.default, deviceIdentifier: service.deviceIdentifier)
        let screen = SUIIndexScreen(serviceProvider: BCServiceProvider.mock, service: service, searchBoxViewModel: searchBoxViewModel)
        return Group {
            //Dark mode
            HStack {
                ZStack {
                    Color.black.edgesIgnoringSafeArea(.all)
                    screen
                    .environment(\.colorScheme, .dark)
                }
            }.previewDisplayName("Dark")
            //Light Mode
            screen
            .previewDisplayName("Light")
        }
    }
}
```


Snapshot tests

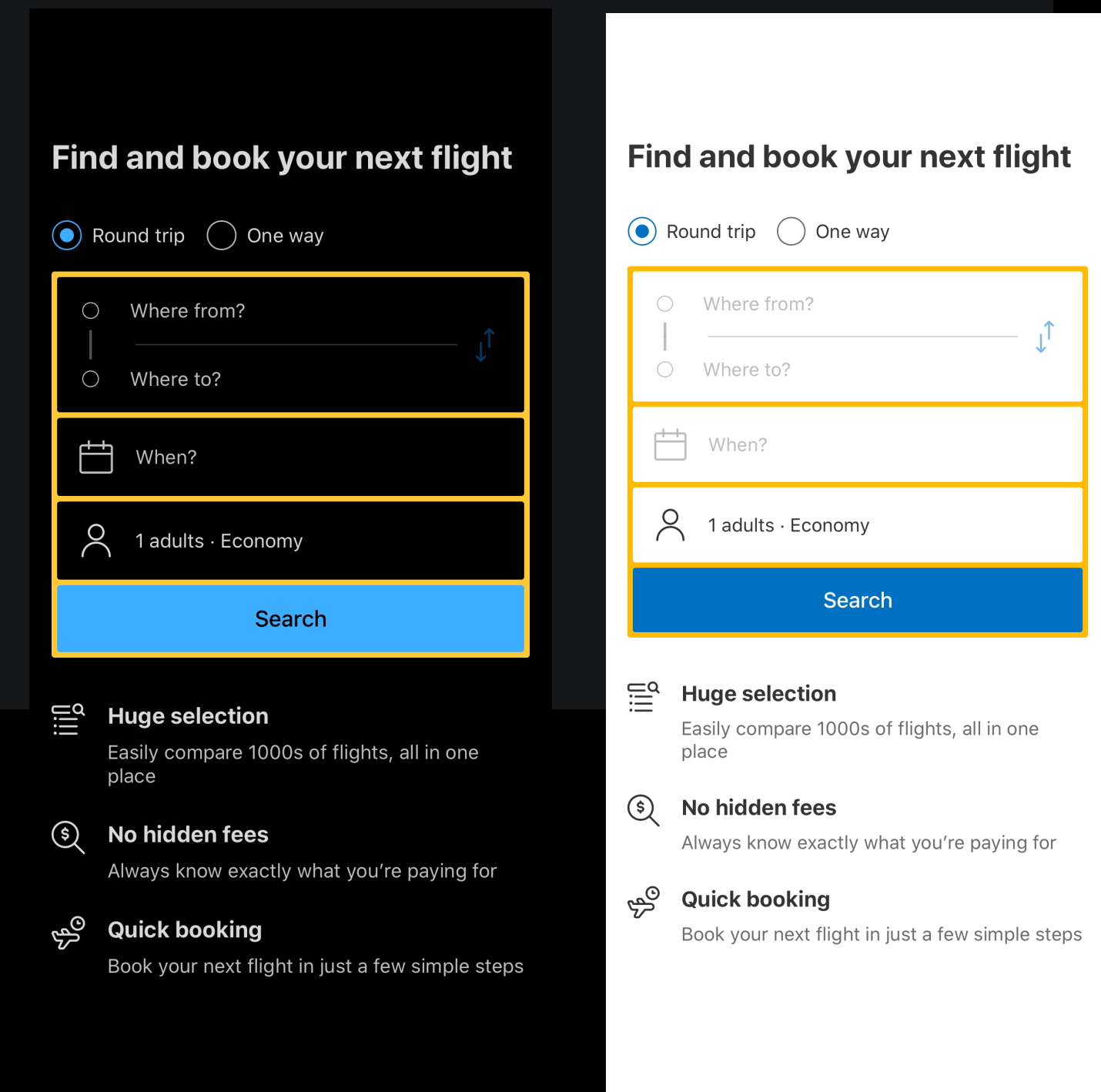
```
func createBaseIndexScreen() -> some View {
    let service = FlightsService(apiClient: FlightAPIMock(), deviceIdentifier: "Preview_Device", locale: "en-us")
    let searchBoxViewModel = SearchBoxViewModel(model: SearchOptionsModel.default, deviceIdentifier:
service.deviceIdentifier)
    return SUIIndexScreen(serviceProvider: BCServiceProvider.mock, service: service, searchBoxViewModel:
searchBoxViewModel)
}

// FBSnapshotTests
func testIndexScreen() {
    FBSnapshotVerifyView(viewFromScreen(createBaseIndexScreen( )))
}

func testIndexScreenDarkMode() {
    FBSnapshotVerifyView(viewFromScreen(createBaseIndexScreen( ), darkMode: true))
}
```

mocks + 1 line test + generated images in repo

`testIndexScreen@3x.png`



SwiftUI issues

+

Костыли

Customizing List parameters, without affecting whole application

```
List {  
    Section { ... }  
    Section { ... }  
}.onAppear {  
    UITableView.appearance().tableFooterView = UIView()  
    UITableView.appearance().separatorStyle = .none  
    UITableViewHeaderFooterView.appearance().tintColor = UIColor.clear  
    UITableView.appearance().backgroundColor = UIColor.themeGroupedTableViewBackground.color  
}
```


SwiftUI works great unless
you find a bug

Проблема с ZStack + ViewModifier

iOS 13.3

10:47

Отмена

Кто полетит?

Пассажиры

Взрослые
18 лет и старше

Дети
От 0 до 17 лет

Салон

☒ Эконом

☐ Премиум-эконом

☐ Бизнес

☐ Первый класс

Готово

iOS 13.1

5:23

```

@available(iOS 13.0.0, *)
extension View {
    @available(iOS 13.0.0, *)
    func bottomSheet<SheetContent: View>(
        presented: Binding<Bool>,
        backgroundColor: Color = BUIColor.themeSecondarySystemBackground.asColor,
        view: @escaping () -> SheetContent ) -> some View {
        modifier(
            SUIBottomSheet(
                presented: presented,
                backgroundColor: backgroundColor,
                view: view
            )
        )
    }
}

```

```

var body: some View {
    ZStack {
        innerBody

        BottomSheetModal(display: $showChildAgePicker) {
            SUICildAgePicker(selectedAgeBinding:
                self.$travellersDetails.childrenAges[self.pickerForChildWithIndex])
                .padding([.trailing, .leading], BUIDoublePaddingUnit)
        }
    }
}

```

```
struct View11: View {
    var body: some View {
        NavigationView {
            Text("You will not see me on iOS 13.1, only on iOS 13.5")
        }
        .modifier11()
    }
}

extension View {
    func modifier11() -> some View {
        modifier(Modifier11())
    }
}

struct Modifier11: ViewModifier {
    func body(content: Content) -> some View {
        ZStack {
            content
        }
    }
}
```

Нежданчик с spacer()

```
private func actionsTopBar() -> some View {
    VStack(alignment: .center) {
        Divider()
        HStack {
            Spacer()

            Button(action: {
                self.showSortOptionsBottomSheet.toggle()
            }) {
                self.actionButton(icon: .sortIcon, title: "Sort")
            }

            Spacer()

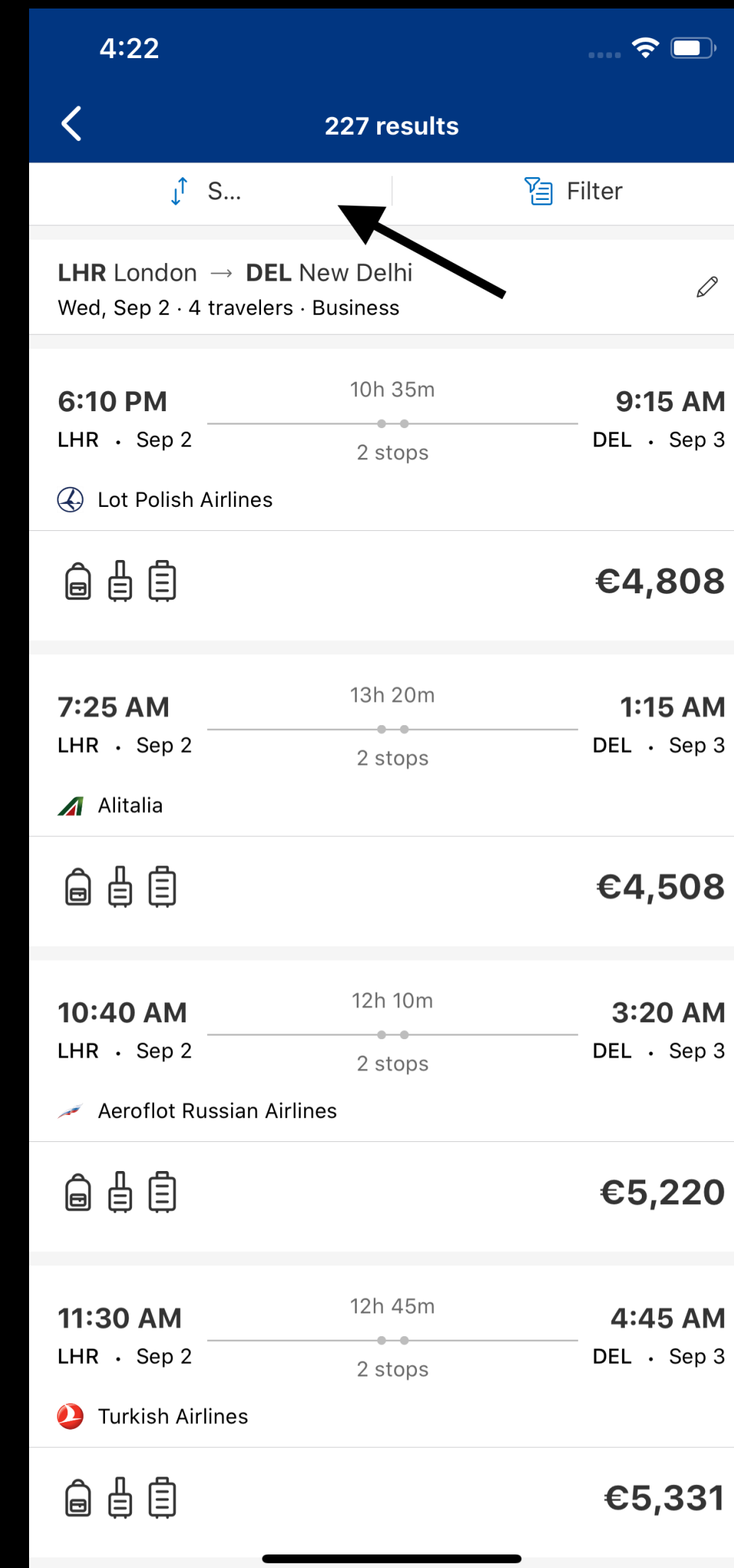
            Divider().frame(width: 1)

            Spacer()

            Button(action: {
                self.showFiltersScreen.toggle()
            }) {
                self.actionButton(icon: .filterIcon, title: "Filter")
            }

            Spacer()
        }
        .padding(.bottom, BUISinglePaddingUnit)
    }
}
```

iOS 13.5 iPhone 11 Max



Не забывайте про Optiona Link Binary

Twitter - место куда жалуются iOSники



Demo

Some tips about SwiftUI

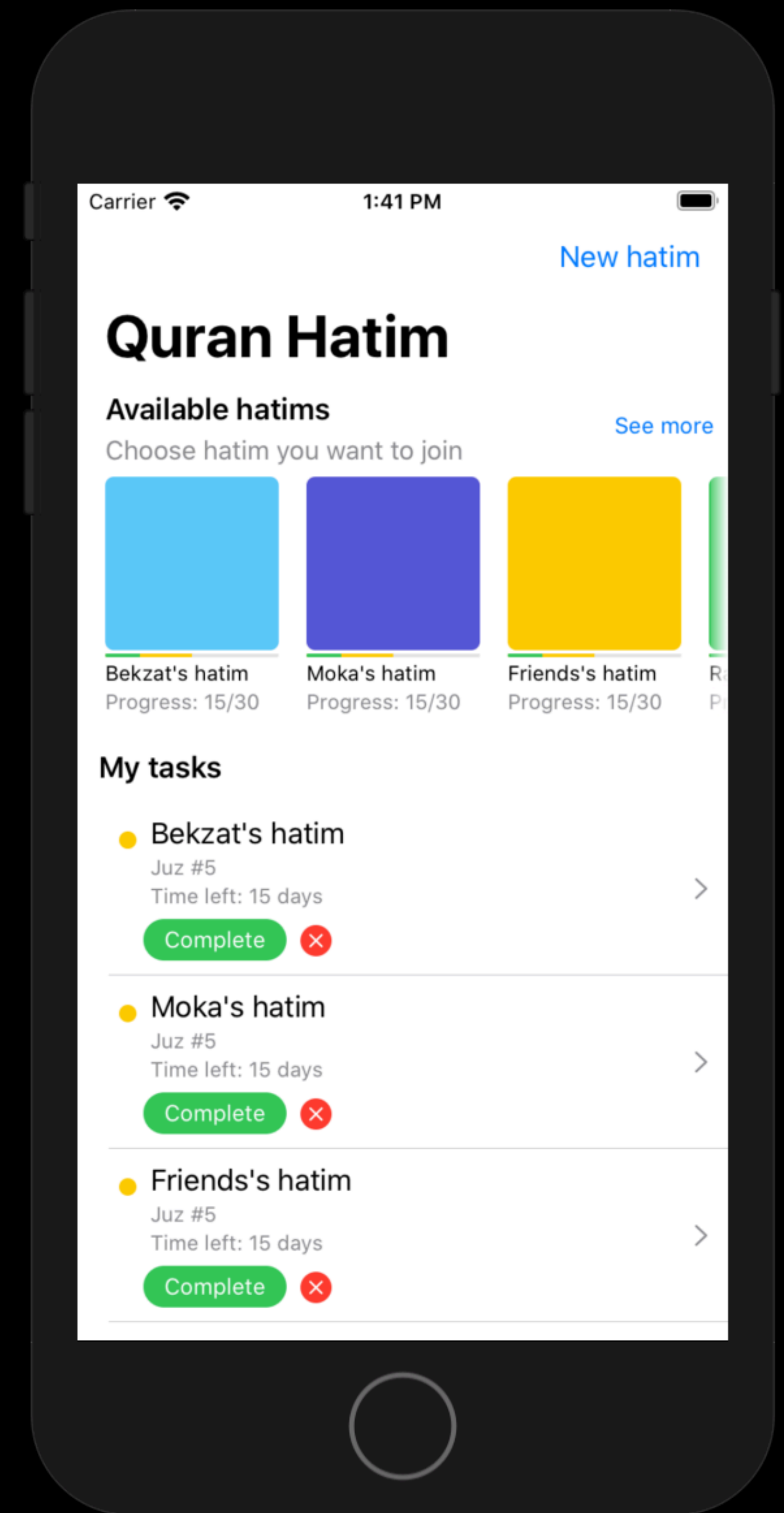
SwiftUI even with bugs is not problem

- It's solution

Некоторые pet-projects раньше было тяжело начинать

из-за большого boilerplate кода, которое отбивает желание

SwiftUI должен убрать этот барьер и позволить снова любить разработку



New to iOS ?

SwiftUI or UIKit ?

UIKit 100%

Useful resources:

- WWDC videos from 2019
- Thinking in SwiftUI ? (not sure)
- <https://www.swiftbysundell.com>
- <https://www.hackingwithswift.com/100/swiftui>
- <https://swiftui-lab.com/>
- <https://medium.com/@nalexn>
- <https://www.vadimbulavin.com/>
- <https://github.com/bigmountainstudio/About-SwiftUI>