Razvoj mobilnih aplikacija i servisa

Mobilni korisnički interfejs i interakcija

Katedra za računarstvo
Elektronski fakultet u Nišu

Korisnički interfejs mobilne aplikacije

- Korisnički interfejs – *User Interface* – UI
  - Može iznositi i do 80% koda
  - Sofisticiran, sa svim elementima GUI: meniji, *buttons*, *text box*, *listbox*, grafika, itd.
- Razvoj generičkog (opšteg) korisničkog interfejsa
  - Velika raznolikost mobilnih uređaja i mobilnih platformi tako da je neophodno omogućiti prilagođenje korisničkog interfejsa čak i u realnom vremenu
  - Jednostavna izmena korisničkog interfejsa usled novih zahteva
- Neophodan pomeraj u paradigmi projektovanja korisničkog interfejsa sa PC desktop paradigme koju karakterišu ekran, miš, tastatura, na mobilnu paradigmu: mali ekran, mala tastatura ili *keypad*, stylus, glas, itd.
- Multikanalni *(Multichannel*) i multimodalni *(Multimodal)* – korisnički interfejs omogućava interakciju korisnika sa mobilnom aplikacijom na različite načine (npr. glas, GUI, itd.), pri čemu korisnik može da izabere najpogodniji način interakcije na datom mestu i trenutku

Interakcija korisnika sa mobilnom aplikacijom

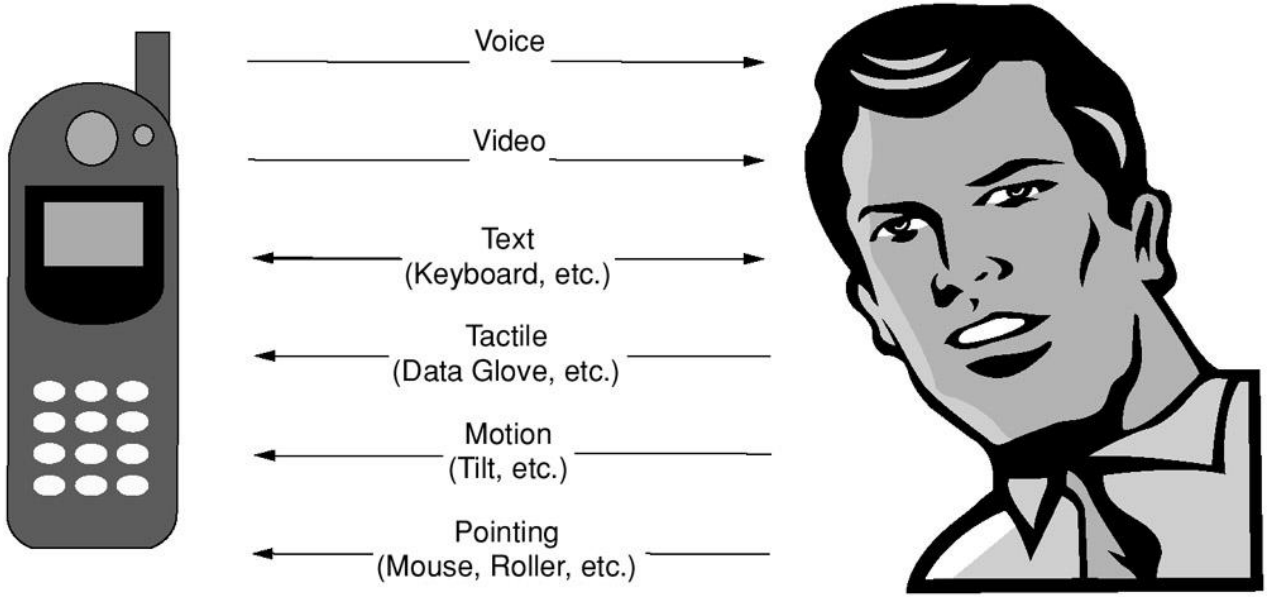
- Mobilni korisnik se kreće, i tako menja lokaciju, uslove i okruženje u kome koristi uređaj i odgovarajuću mobilnu aplikaciju
- Mobilni korisnik uglavnom nije fokusiran na aplikaciju, jer je koristi u pokretu radeći više drugih poslova istovremeno (vozi, šeta, razgleda okolinu, itd.)
- Mobilni korisnik često zahteva visok stepen interaktivnosti i brzi odziv aplikacije
- Mobilni korisnik menja aktivnosti često i/ili iznenadno; tipično to nisu aktivnosti/zadaci koji uključuju veliku količinu podataka i duge transakcije.
- Mobilni korisnik želi da pristupa aplikaciji na svakom mestu u svakom trenutku preko najpogodnijeg načina (moda) za interakciju; izbor omogućava fleksibilnost i upotrebljivost aplikacije

Mobilni korisnički interfejs - značajni aspekti

- Vreme pristupa aplikaciji i vreme odziva
- Jasan i efikasan korisnički interfejs
- Održavanje konzistentnosti između različitih korisničkih interfejsa i različitih tipova uređaja prilikom pristupa aplikaciji
- Sposobnosti i ograničenja ljudskih čula (vida, sluha, dodira)
- Adaptivnost na lokaciju korisnika i njegovo okruženje
- Mogućnost da se definišu prioriteti elemenata korisničkog interfejsa
- Generički korisnički interfejs - zbog velike raznolikosti mobilnih uređaja i platformi i različitih modova interakcije korisnika sa mobilnom aplikacijom

Elementi mobilnog korisničkog interfejsa

- Kanali/ za interakciju/komunikaciju korisnika sa uređajem/aplikacijom i tipovi kanala
  - Medijum za razmenu poruka (komunikaciju) korisnika i aplikacije (preuzeto iz terminologije telekomunikacija)
  - Postoje ulazni i izlazni kanali
  - Zasnivaju se na čulima vida, sluha, dodira, (mirisa?) i aktivnostima korisnika



Načini mobilne interakcije

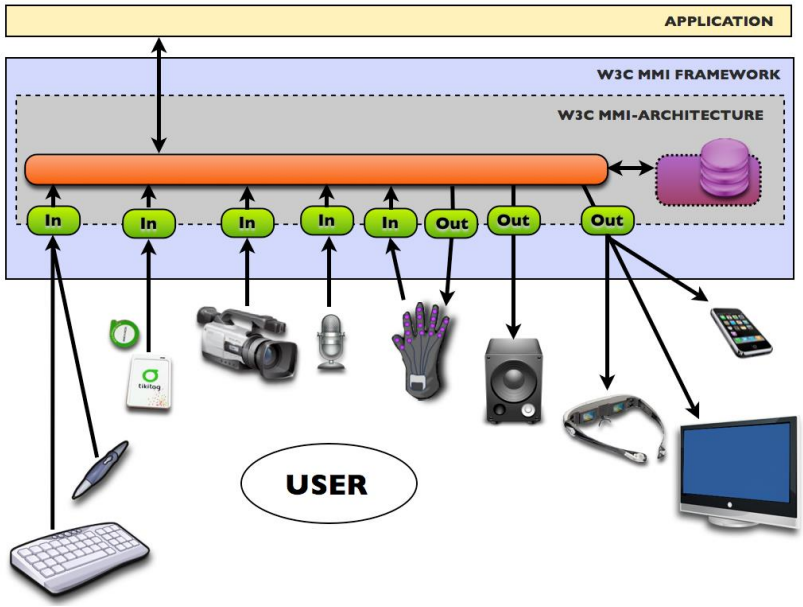
- Tastatura, miš i monitor (unos teksta i GUI prikaz)
- Štampani papir (izlaz u obliku teksta i grafike)
- *Stylus* (prepoznavanje pisanih znakova i unos dodirom)
- *Touch-Screen* (unos dodirom, *multi-touch* i GUI prikaz)
  - Prepoznavanje gestova (*gesture recognition*)
- Mikrofon-zvučnik (unos na osnovu prepoznavanja govora i glasovni/zvučni odgovor)
- Pomeranje uređaja (unos pomeranjem, trešenjem i postavljanjem uređaja u određenu poziciju)
- Odziv na osnovu dodira (*Haptic feedback, vibration*)
- Specijalni uređaji
  - Specijalna rukavica (*dataglove*) - unos putem dodira

Multimodalni korisnički interfejs

- Bira se najpogodniji kanal za komunikaciju/interakciju u skladu sa kontekstom korisnika
- Multimodalni sistemi se karakterišu inerfejsom čoveka i računara koja prevazilazi tradicionalne načine putem tastature, miša i ekrana.
  - Obezbeđuje različite načine (modove) interakcije sa aplikacijom, poput glasa ili gestova
  - Mora da prepozna ulaze sa različitih modova ulaza (ulaznih kanala), kombinuje ih i interpretira
- Multimodalni UI omogućava različite načine interakcije korisnika sa aplikacijom u okviru istog korisničkog interfejsa
  - UI se adaptira kontekstu upotrebe aplikacije
  - U skladu je sa korisnikovim trenutnom situacijom, ciljevima, potrebama i ograničenjima

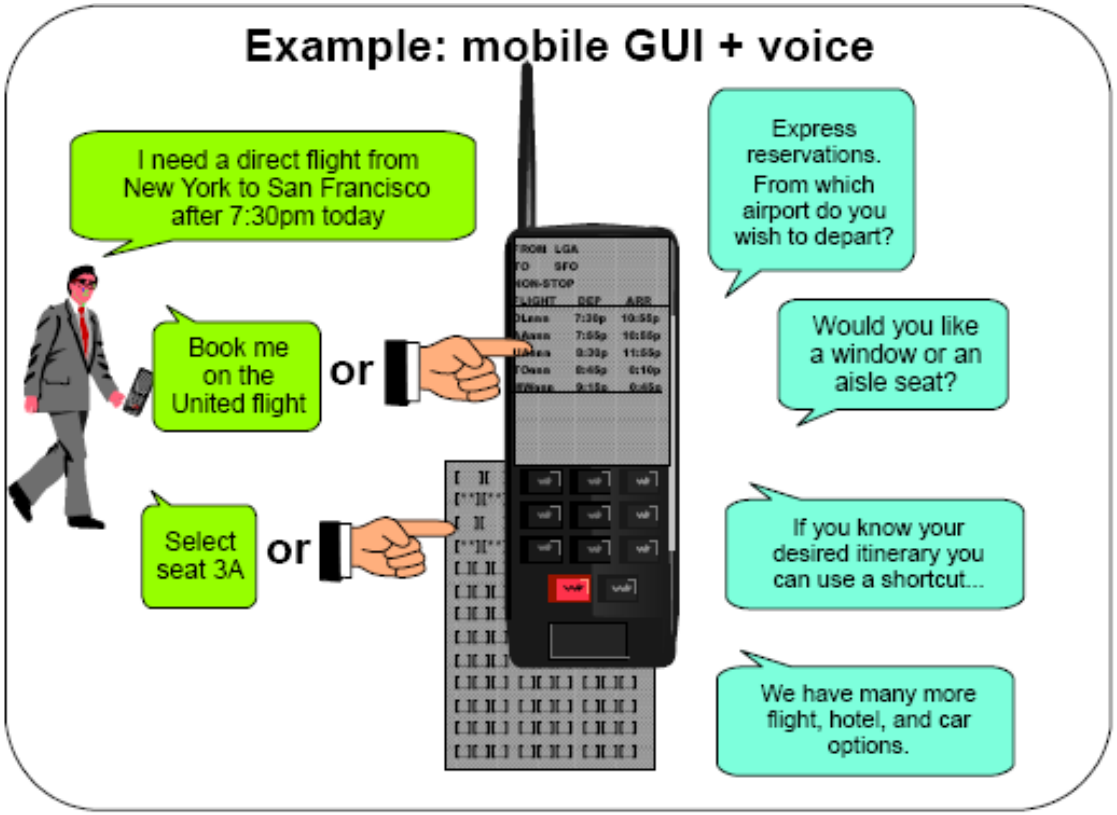
Multimodalni UI

- Multimodalni ulaz
  - Glas, olovka, tastatura, dodir, manuelni gestovi, pogled, pokreti glave, tela, kamera, ...
- Multimodalni izlaz
  - Ekran, zvuk (glas), dodir (vibracija), (miris?)
- Multimodalna fuzija ulaza i fisija izlaza



Multimodalni interfejs – primer Rezervacija avio karata

- Korisnik može izabrati najpogodniji način interakcije
- Korisnik nije ograničen tokom interakcije aktivnostima definisanim za određeni kanal



Interakcija između korisnika i mobilne aplikacije

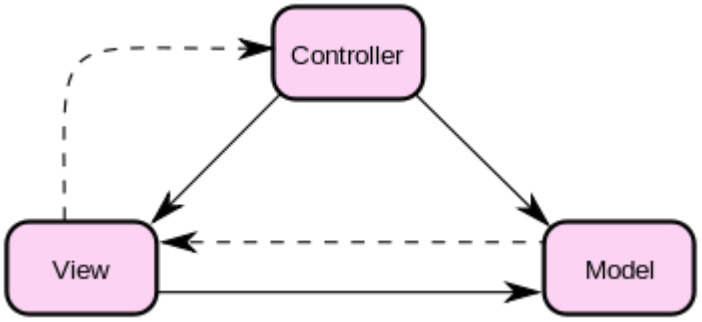
- Komunikacija između korisnika i aplikacije predstavlja interakciju koja se sastoji od poruka koje razmenjuju korisnik i aplikacija
- Interakcija može biti atomična (unos teksta u textbox) ili kompozitna, sastavljena od više atomičnih interakcija u formi dijaloga
- Elementi interakcije
  - Kontrolne poruke – upravljaju tokom izvršenja aplikacije
  - Prompt-ovi – aplikacija zahteva unos od strane korisnika
  - Odgovori – od strane korisnika ili aplikacije
- Tipovi interakcije
  - Komande – predefinisana lista tekstualnih komandi
  - Meniji – selekcija jedne ili više ponuđenih opcija
  - Forme – GUI kontrole i widget-i
  - Prirodni jezik
  - Touch gestovi
  - Kombinacija različitih tipova interakcije
- Kontekst
  - Sve informacije koje karakterišu korisnika, njegov mobilni uređaj, bežičnu mrežu i okruženje u kome se nalazi značajni za korišćenje aplikacije

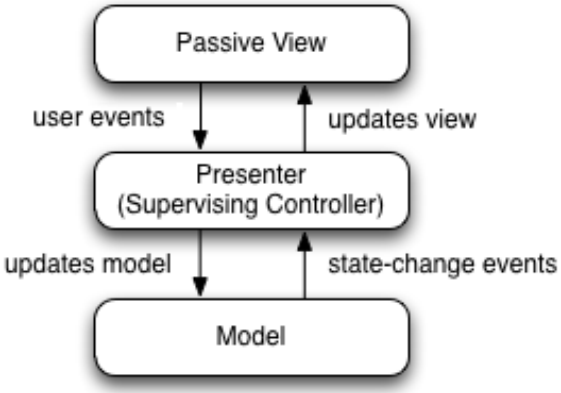
Primer: Navigacija vozila

- Zahtevi multimodalne aplikacije za navigaciju
  - Da omogući korisniku da dobije instrukcije za navigaciju od sistema korišćenjem browser HTML interfejsa, glasovnog interfejsa, vizeulno (na ekranu)
  - Da obezbedi načine za brzo dobijanje navigacionih instrukcija nezavisno od korisničkog interfejsa za ekspertske korisnike
  - Da obezbedi pomoć i uputstva za nove korisnike
- Tri tipa dijaloga korisnika i aplikacije i njihova kombinacija
  - Zasnovan na formama i unosu u forme
  - Zasnovan na govornom jeziku
  - Touch gestovi
  - Kombinacija unosa u forme, touch gestova i govornog jezika

Arhitekturni obrasci mobilnog korisničkog interfejsa

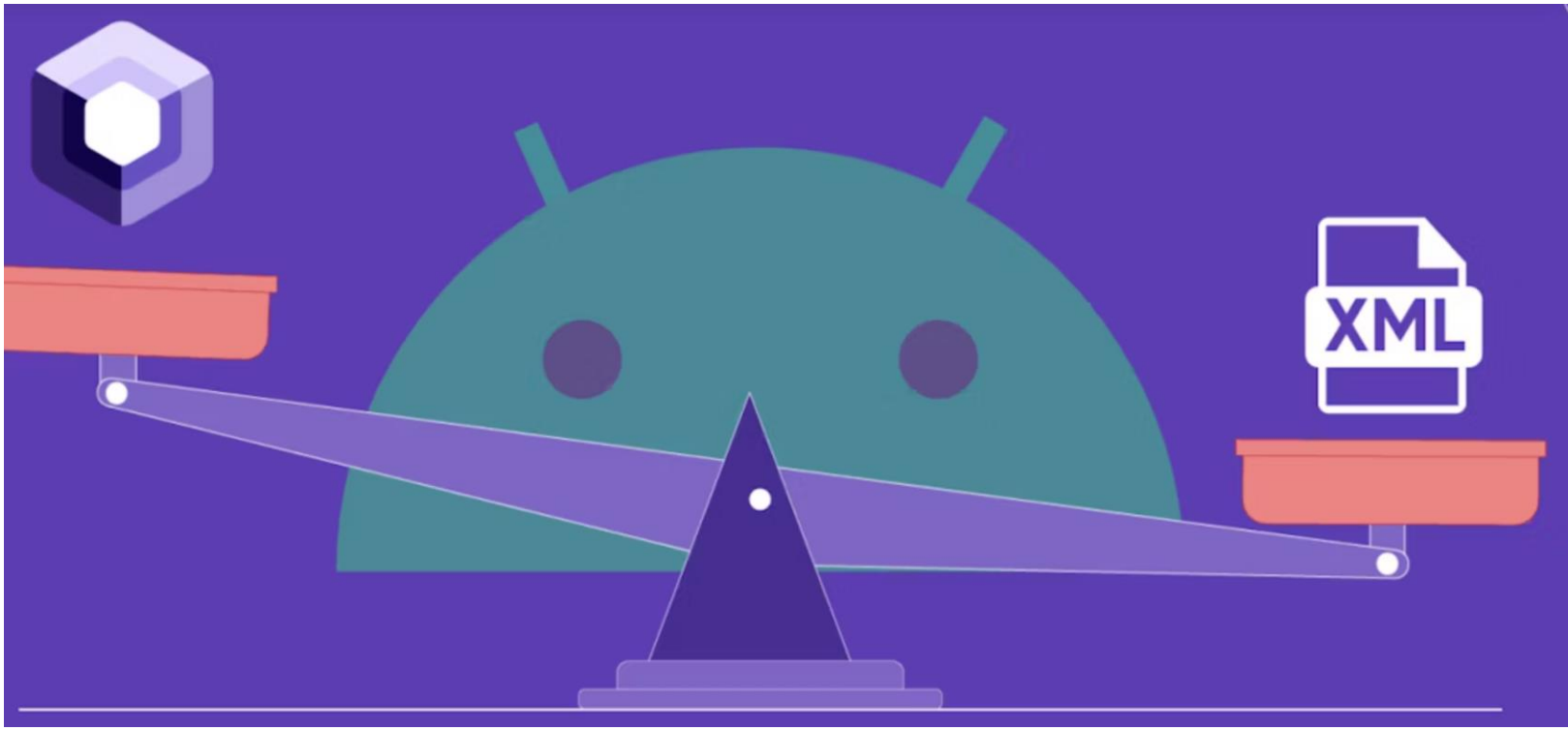
- MVC (Model-View-Controller)
  - Široko implementiran u desktop aplikacijama i Web aplikacijama sa tankim klijentom, gde postoji jedan tip kontrolera i jedan tip pogleda
  - U mobilnim aplikacijama više modova interakcije zahteva više pogleda i više kontrolera za svaki od tih pogleda (održavanje konzistentnosti između različitih pogleda i kontrolera)
- Model-View-ViewModel
- Model–View–Presenter





Mobilni UI design pattern-i

- Obrasci (*pattern*-i) za dizajn korisničkog interfejsa mobilnih aplikacija
  - *Navigation*
  - *Forms*
  - *Search, Sort & filter*
  - *Tools*
  - *Invitations*
  - *Feedback & Affordance*
  - *Anti-Pattern-i*
- *Design patterns for mobile apps*, Ivano Malavolta, University of L'Aquila (Italy)
- *Mobile Design - Strategic Solutions* – Theresa Nail



ANDROID UI

Android UI

- Dva pristupa
  - **XML**
    Tradicionalni pristup dostupan od nastanka Android-a
    Hijerarhijska struktura
    Implementira *separation-of-concern* koncept
  - **Jetpack Compose**
    Objavljen na Google I/O 2019.
    Deklarativni pristup
    Reaktivno programiranje
    Dinamički data-binding

Android UI - XML

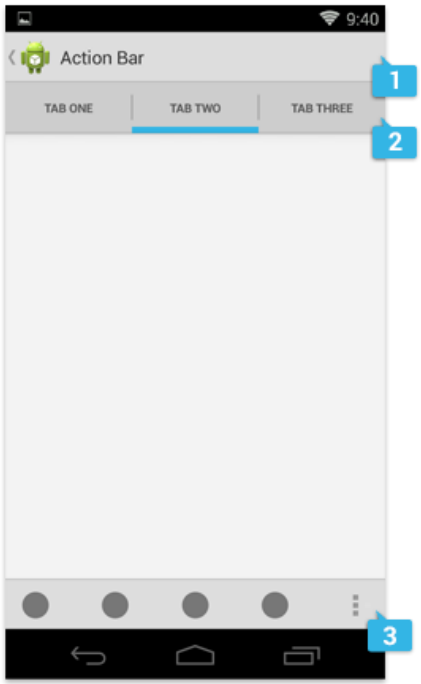
- Koristi tagove i atribute
- Veliki ekosistem alata i komponenti
- Glomazan i nejasan za složene UI
- Zahteva boilerplate programski kod

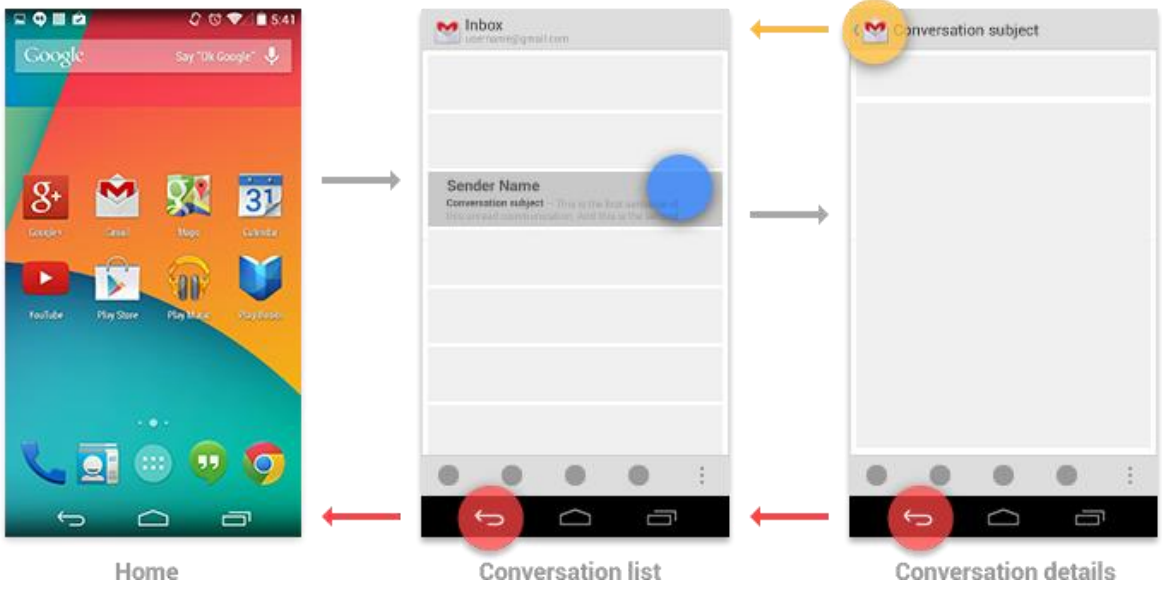
```
<!-- Example XML Layout for a Login Screen -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- ... (Previous XML layout code here) ... -->

</LinearLayout>
```

Android UI

- Status Bar
- Navigation bar

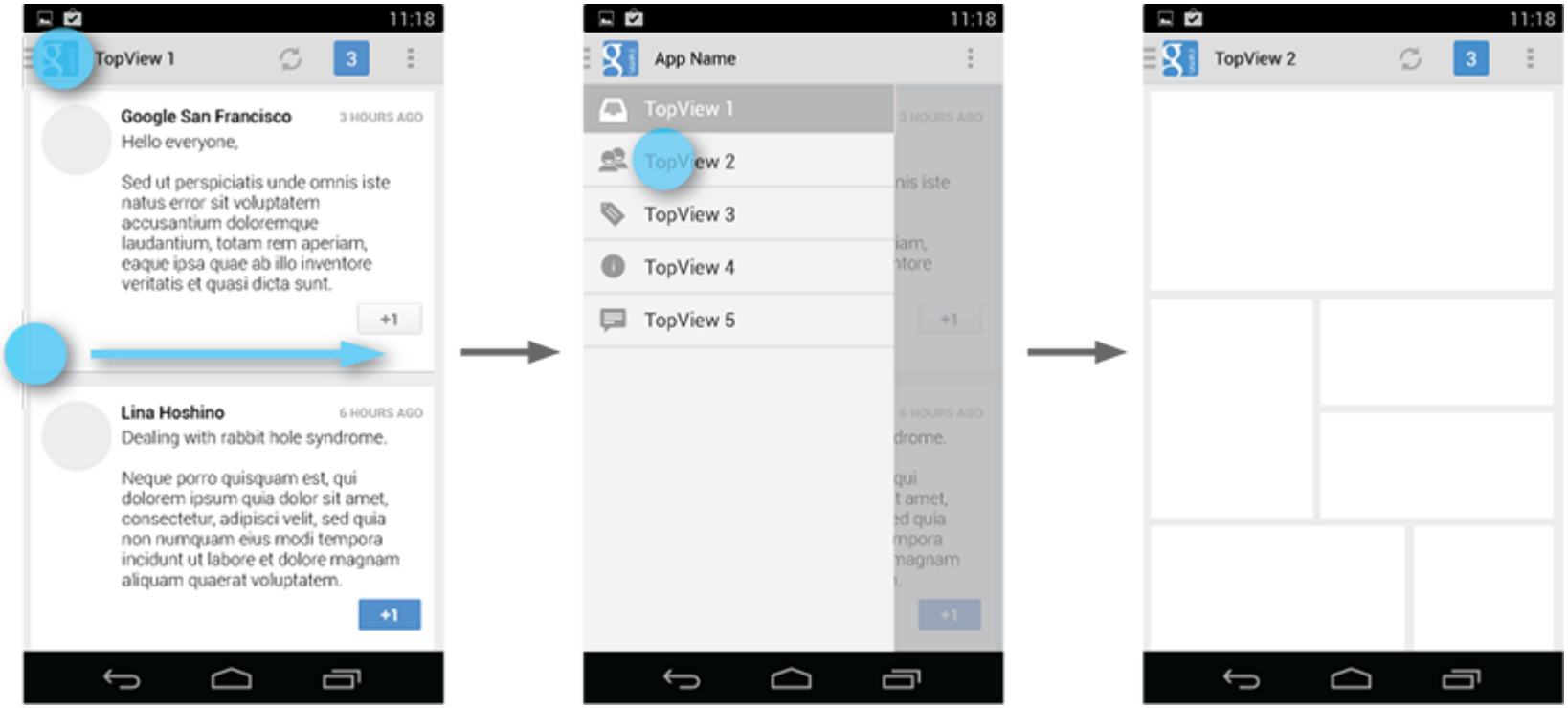
- Action (app) bar





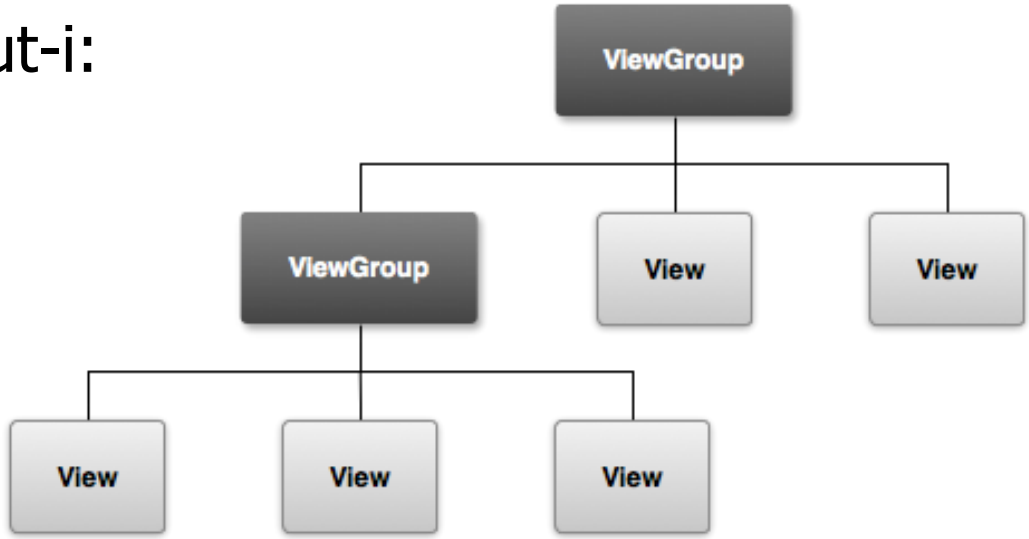
Android UI

- Navigation Drawer



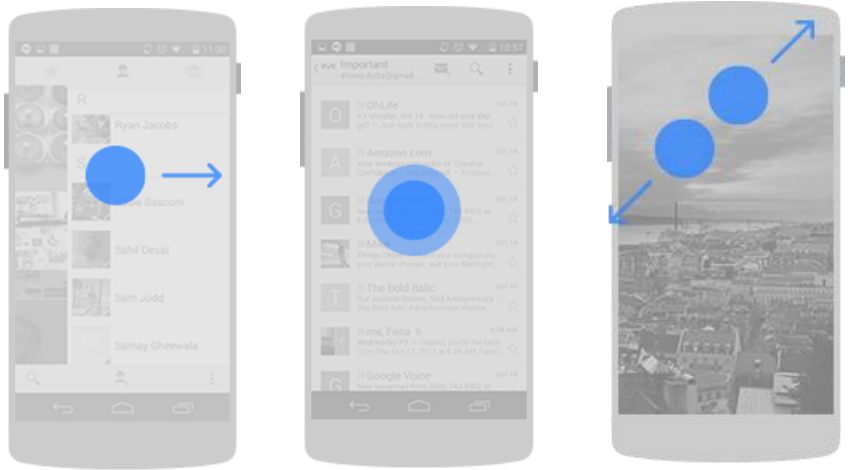
Android UI

- Hijerarhija *View & View Group*
- Najčešće korišćeni layout-i:
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - ScrollView
  - WebView
  - ListView
  - RecyclerView
  - GridView, ...
- GUI control-e (*widget-i*)
  - Button, TextView, EditText, CheckBox, RadioButton, Picker, Spinner, Slider, ...



Android touch gestovi

- Gestovi
  - Touch, Long press, Swipe/drag, Long press drag, Double touch, Double touch drag, Pinch open, Pinch close, ...
- Klase i interfejsi
  - Activity/View callback metod *onTouchEvent(MotionEvent)*
  - Android *GestureDetector* class
  - Interfejsi *GestureDetector.OnGestureListener* *.OnDoubleTapListener*



Android – *Material Design*

- *Material design* – Android 5.0
  - *Comprehensive guide for visual, motion, and interaction design across platforms and devices*.
- Elementi Material design-a
  - *Material* tema
  - Widget-i za kompleksne *view-e (cards, lists)*
  - Senke view-a u zavisnosti od *z* vrednosti view-a i isecanje view-a (*shadows and view clipping*)
  - Nove karakteristike objekata koji se iscrtavaju (*drawables*)
    - vektorski, slike, ekstrakcija boja
  - *Custom* animacije

Jetpack Compose

- Kombinuje reaktivno programiranje sa konciznošću Kotlin programskog jezika
- Potpuno je deklarativan kao Flutter, SwiftUI, React Native
- Princip je da se UI opisuje pozivom funkcija koje transformišu podatke u UI
- Sa promenom podataka framework automatski poziva funkcije i ažurira UI
- Composable funckija se označava *@Composable* anotacijom

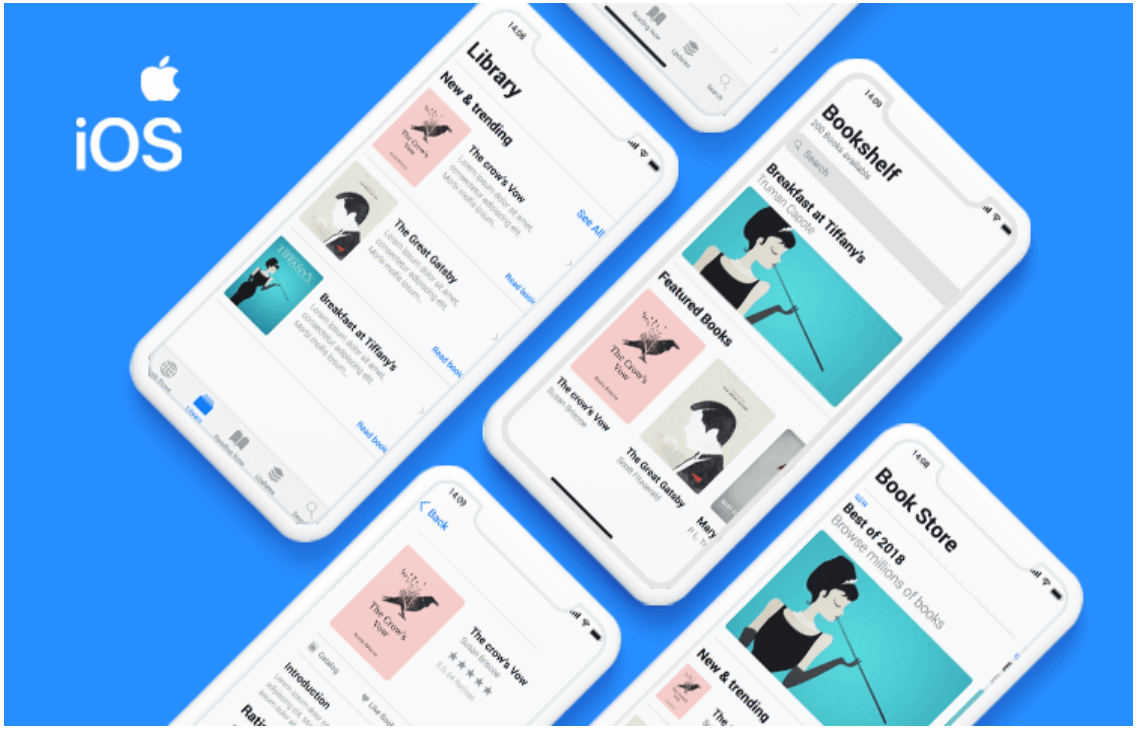
Jetpack Compose

- Primer Login ekrana

```
//Example XML Layout for a Login Screen
@Composable
fun LoginScreen() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        // ... (Previous Jetpack Compose code here) ...
    }
}
```

Android - UI dizajn

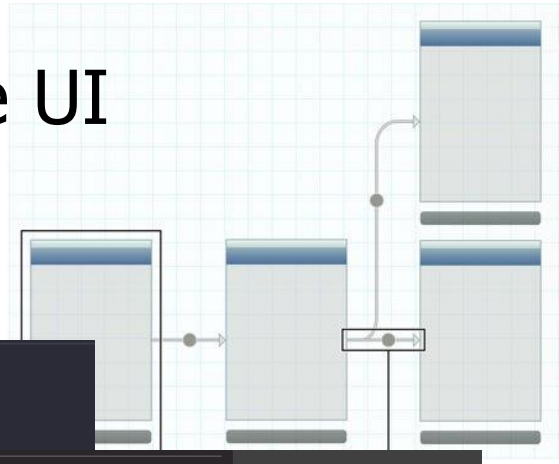
- Android User interface
  - http://developer.android.com/guide/topics/ui/index.html
- Android Design
  - http://developer.android.com/design/index.html
- Design patterns
  - http://developer.android.com/design/patterns/index.html
- Material design
  - http://developer.android.com/design/material/index.html
- Best Practices for Interaction and Engagement
  - http://developer.android.com/training/best-ux.html
- Best Practices for User Interface
  - http://developer.android.com/training/best-ui.html
- Best Practices for User Input
  - http://developer.android.com/training/best-user-input.html


iOS
Library
New & trending
The crow's Vow
The Great Gatsby
Breakfast at Tiffany's
Bookshelf
Featured Books
Book Store
Best of 2018
Browse millions of books

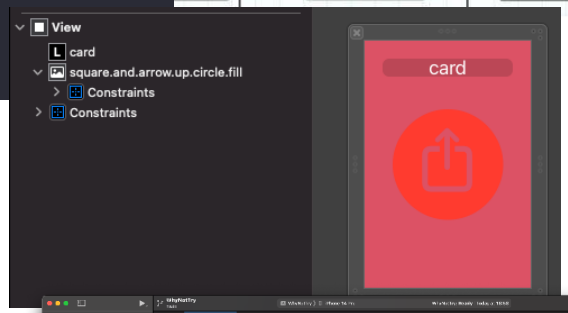
iOS UI

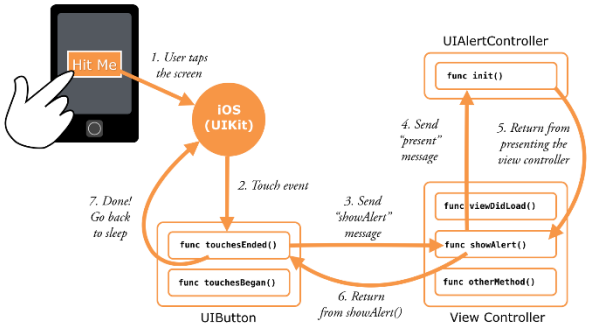
iOS pristupi izgradnji UI

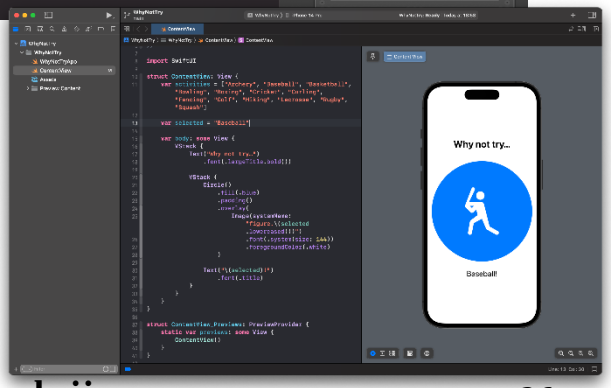
- Xcode nudi nekoliko načina izgradnje UI
  - UIkit
  - Storyboard
  - Programski
  - NIB
  - SwiftUI



```
// MARK: - Views
private lazy var tableView: UITableView = {
    let view = UITableView()
    view.translatesAutoresizingMaskIntoConstraints = false
    view.delegate = self
    view.dataSource = self
    return view
}()
```

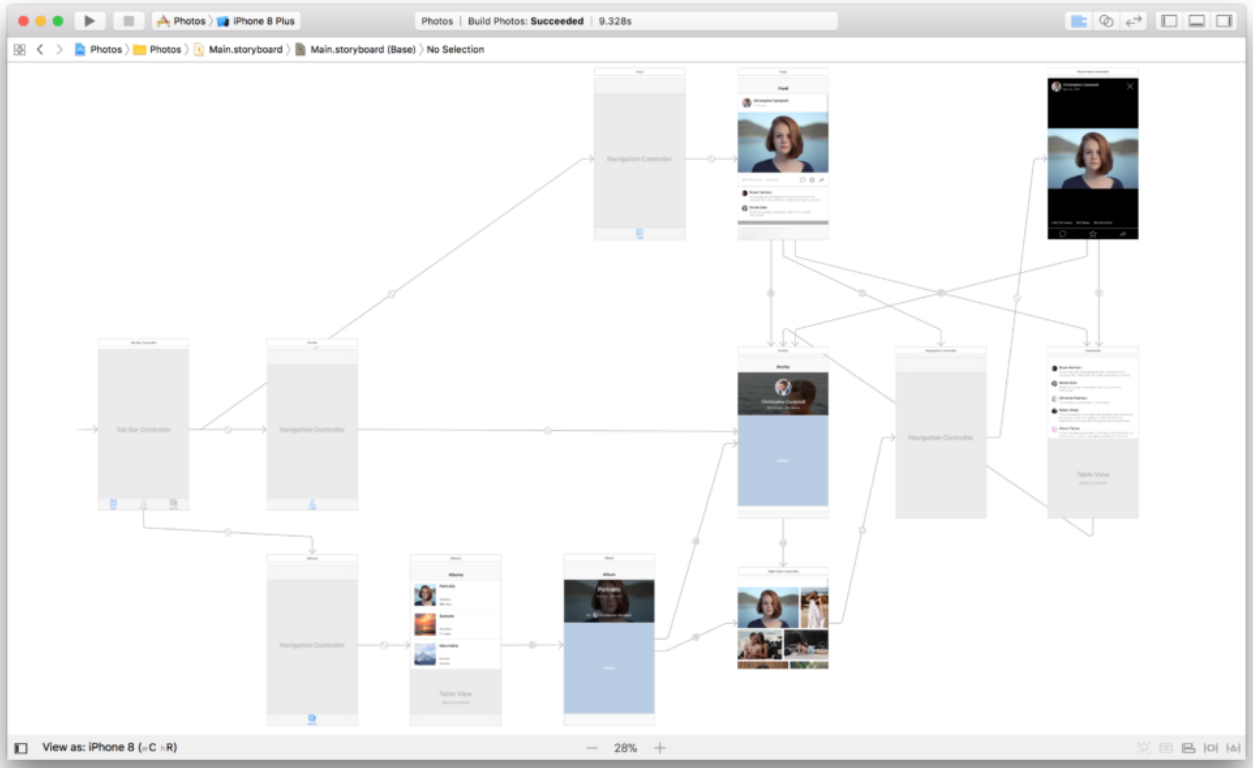






Storyboard

- Vizuelni alat koji omogućava uvezivanje većeg broja ekrana aplikacije (views) i navigaciju između njih

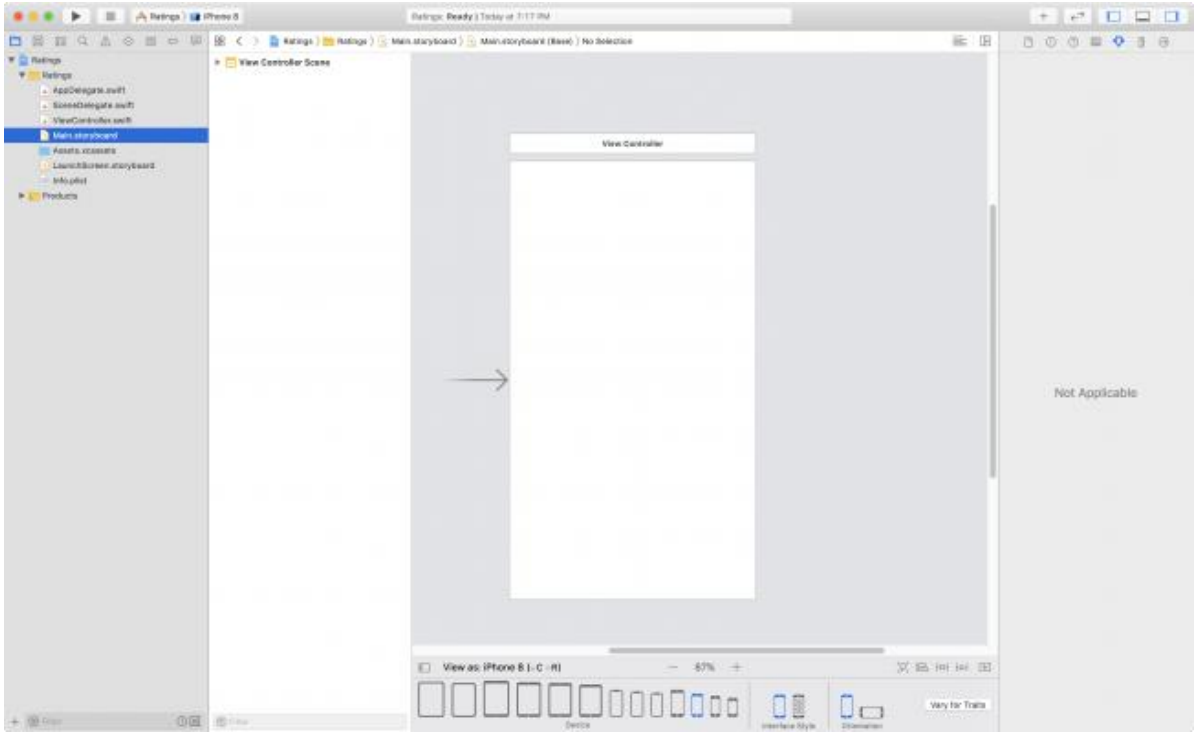


Storyboard

- Prednosti
  - Jednostavno i verno vizuelizira izglede ekrana i veza između njih
  - Dobro integrisan sa Xcode razvojnim okruženjem sa auto-layout opcijom
- Mane
  - Storyboard se konvertuje u složen XML
  - Reuse je složen
  - Kolaborativni rad dovodi do merge konflikata
  - Storyboard obrađuje navigaciju između ekrana ali se ne bavi prenosom podataka što je potrebno realizovati manuelno

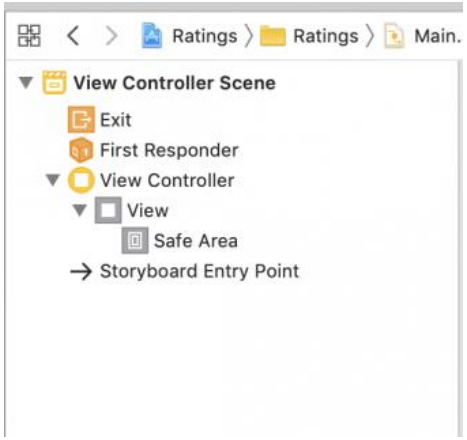
Storyboard

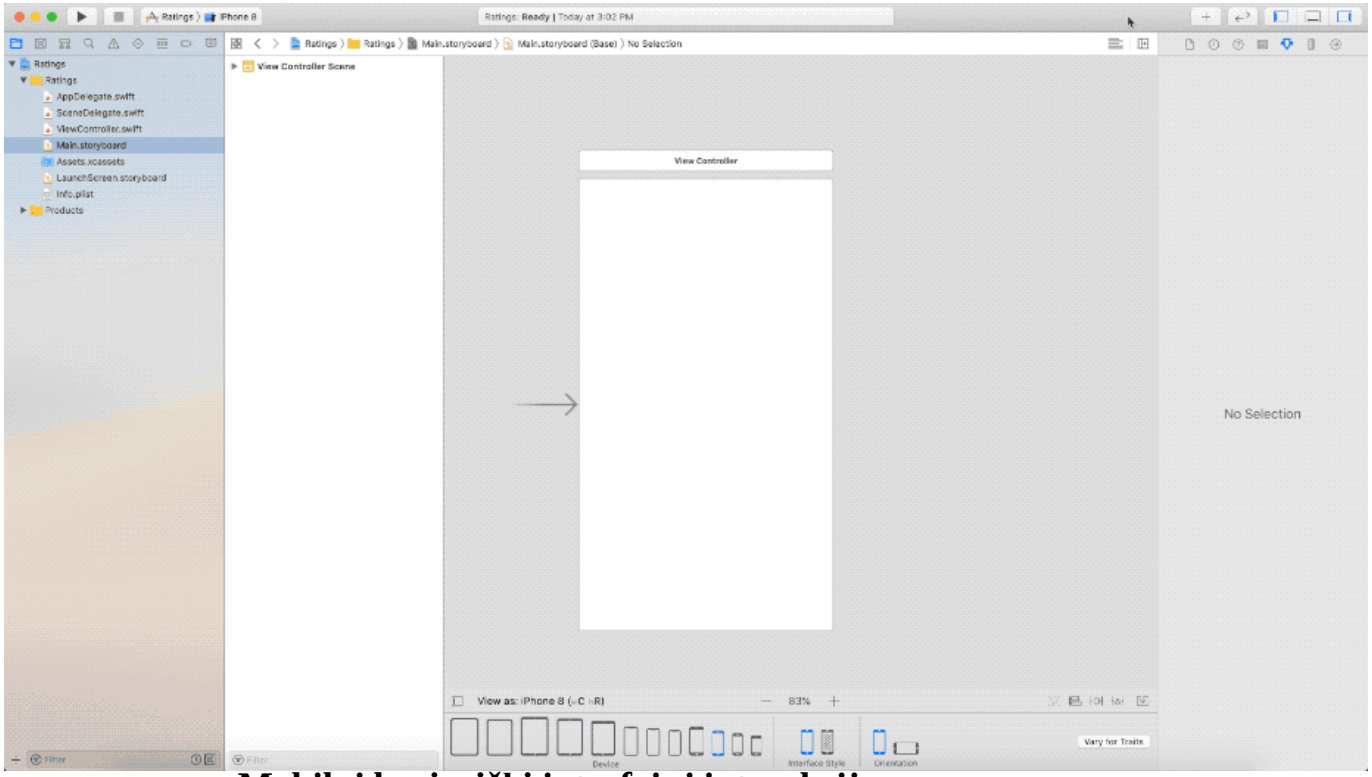
- Prvi put uveden u iOS 5
- Omogućava vizuelno kreiranje View/Controller-a
- Jedan view/controller je inicijalni (strelica pokazuje)



Storyboard

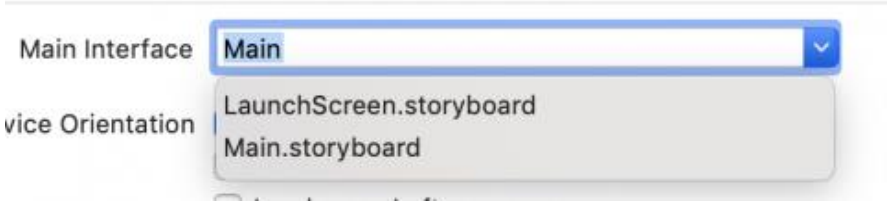
- Za dizajn UI-a se koristi drag&drop kontrola iz **Object Library**
- Struktura scena (view/controller) je prikazana u **Document Outline**



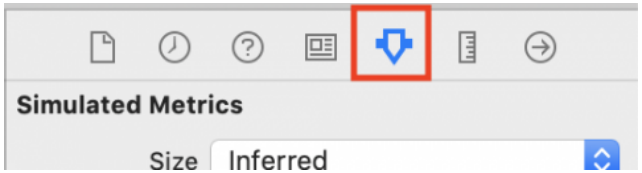


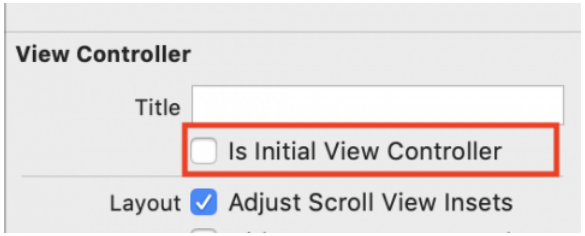
Storyboard

- Projekat definiše jedan storyboard kao **Main interface**



- U storyboard-u jedan View/Controller se bira kao inicijalni

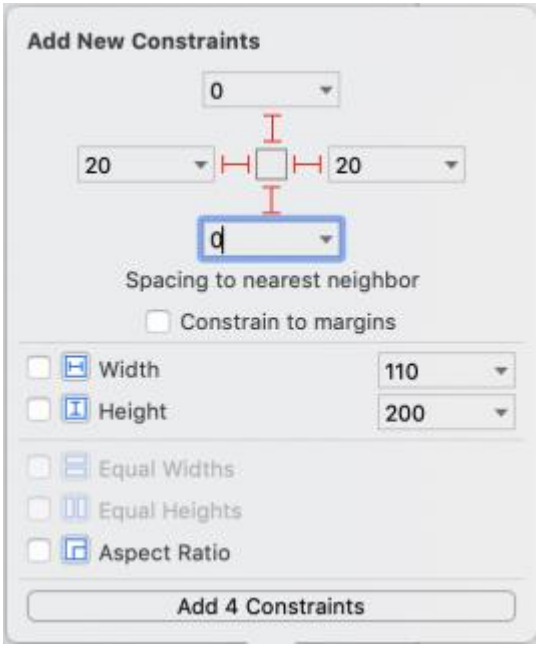




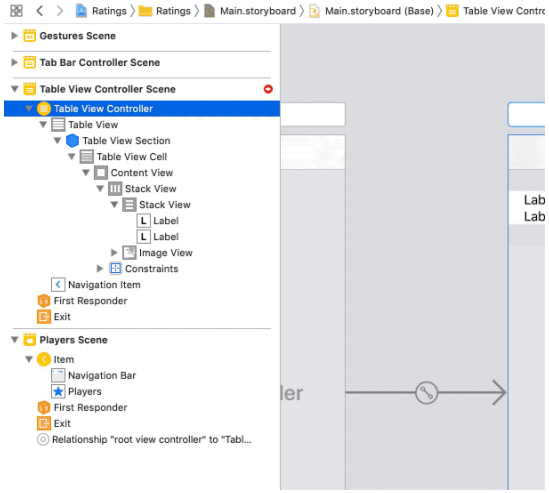
- Inicijalni view/controller je označen strelicom sa leve strane

Storyboard

- Različiti formati displeja se rešavaju Auto Layout ograničenjima

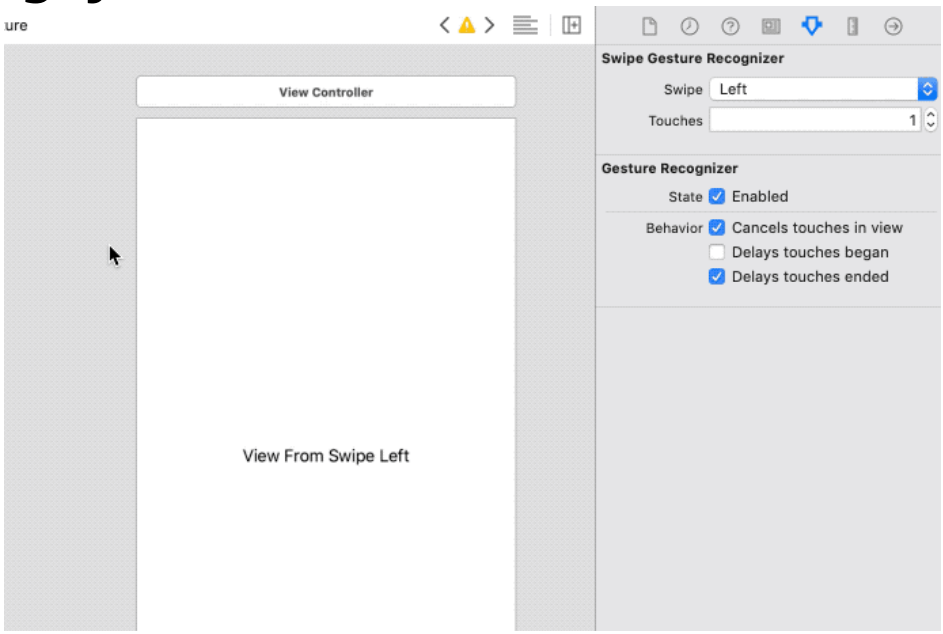


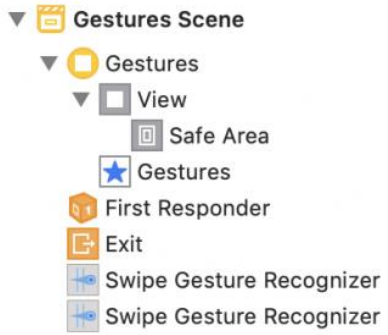
- Document outline prepoznaje greške u layout-u i najčešće nudi rešenja



Storyboard

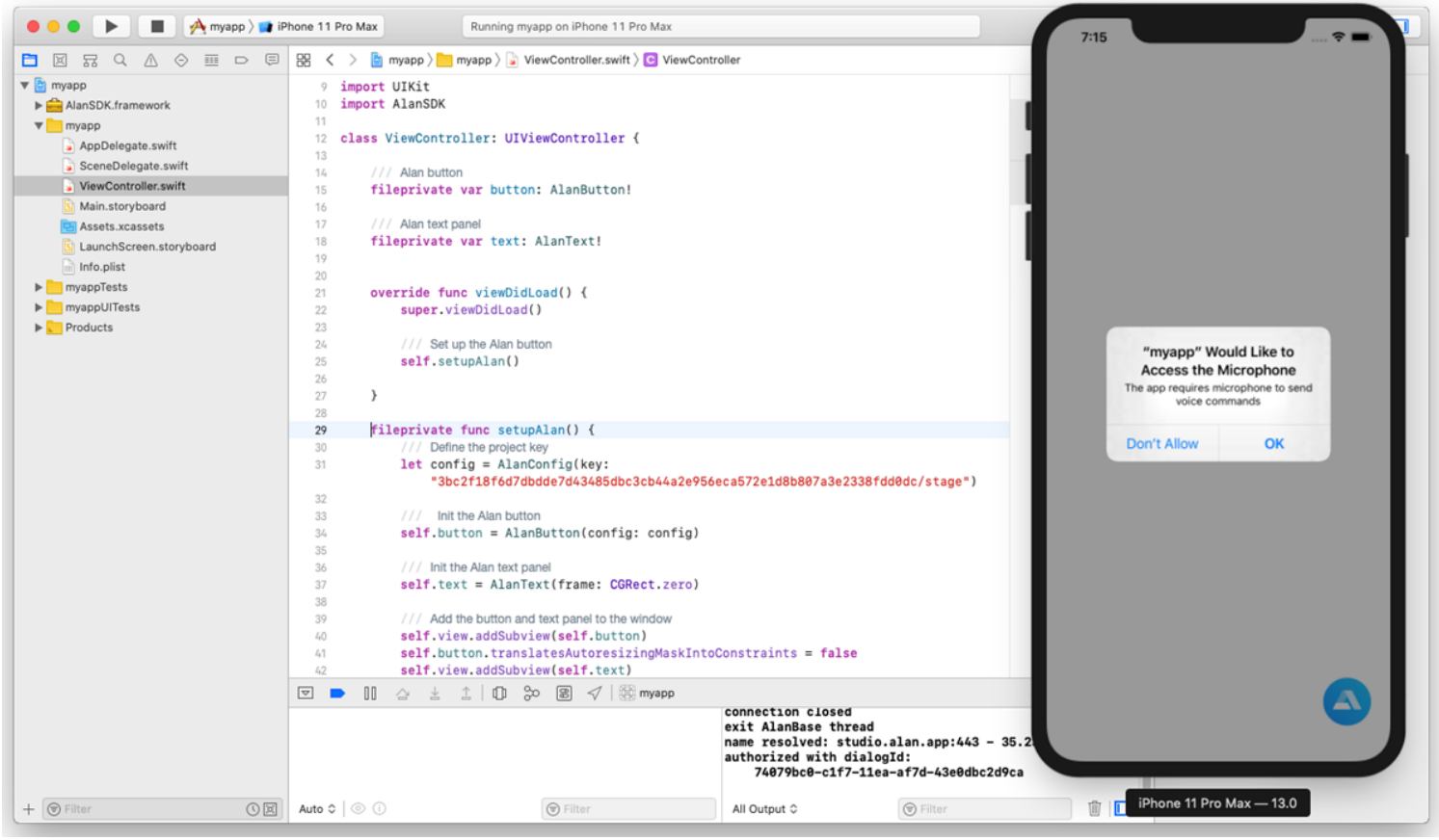
- Korišćenje gestova u navigaciji
- U scenu se dodaju odgovarajući Gesture Recognizers
- Sa Gesture Recognizer se povezuje aktivirana veza (triggered segue) ka drugoj sceni





Programski

- Ne koristi se nikakav GUI alat već se sve definiše iz programskog koda



Programski

- Prednosti
  - Jednostavniji kolaborativni rad, merge konflikti u programskom kodu se lakše rešavaju
  - Korišćenjem Stroyboard-ova ili Nib-ova oni moraju da se učitaju i prsiraju pre instanciranja objekata, što zahteva neko vreme, programski pristup je brži
  - Maksimalna fleksibilnost, GUI alati kao Storyboard ili NIB mogu imati neka ograničenja šta je moguće uraditi
- Mane
  - Nije WYSIWYG pristup zato što se rezultat izmena vidi tek nakon startovanja
  - Osim što se kontroler komponenta piše programski i view komponenta se piše u programskom kodu što nije tako intuitivno

Programski

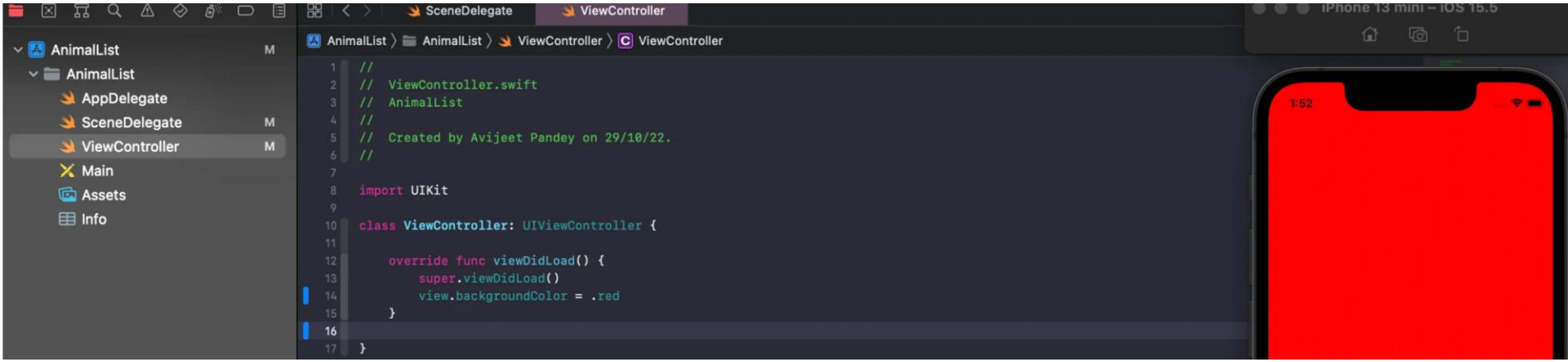
- 3 koraka u programskom kreiranju UI
  - Kreiranje View-eva u programkom kodu
  - Dodavanje View-eva u parent View
  - Definisanje ograničenja programski
- Nakon uklanjanja referenci na Storyboard u funkciji *scene* u *SceneDelegate.swift* treba definisati početnu scenu

```
func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions) {
    // Use this method to optionally configure and attach the UIWindow `window` to the provided UIWindowScene `scene`.
    // If using a storyboard, the `window` property will automatically be initialized and attached to the scene.
    // This delegate does not imply the connecting scene or session are new (see `application:configurationForConnectingSceneSession`
        instead).
    guard let _ = (scene as? UIWindowScene) else { return }

}
```

```
func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions) {
    guard let scene = (scene as? UIWindowScene) else { return }
    window = UIWindow(windowScene: scene)
    let nav = UINavigationController(rootViewController: ViewController())
    window?.rootViewController = nav
    window?.makeKeyAndVisible()
}
```

Programski

- Izmene u ViewController-u (na promer menjamo boju pozadine)



Programski

- Programsko kreiranje view-eva u ViewController.swift (na primer TableView)

```
// MARK: - Views
private lazy var tableView: UITableView = {
    let view = UITableView()
    view.translatesAutoresizingMaskIntoConstraints = false
    return view
}()
```

- Dodavanje view-a u roditeljski kontejner view (u metodi *viewDidLoad*)

```
// MARK: - Lifecycle
override func viewDidLoad() {
    super.viewDidLoad()
    view.addSubview(tableView)
}
```

Programski

- Dodavanje ograničenja

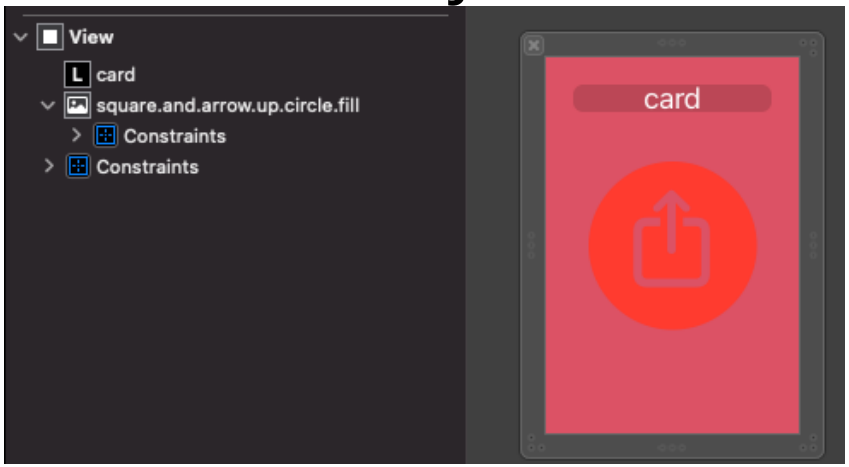
```
// MARK: - Lifecycle
override func viewDidLoad() {
    super.viewDidLoad()
    view.addSubview(tableView)
    NSLayoutConstraint.activate([
        tableView.topAnchor.constraint(equalTo: view.topAnchor),
        tableView.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 16),
        tableView.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -16),
        tableView.bottomAnchor.constraint(equalTo: view.bottomAnchor)
    ])
}
```

- Povezivanje delegata i izvora podataka sa kreiranim pogledom (*TableView*)

```
// MARK: - Views
private lazy var tableView: UITableView = {
    let view = UITableView()
    view.translatesAutoresizingMaskIntoConstraints = false
    view.delegate = self
    view.dataSource = self
    return view
}()
```

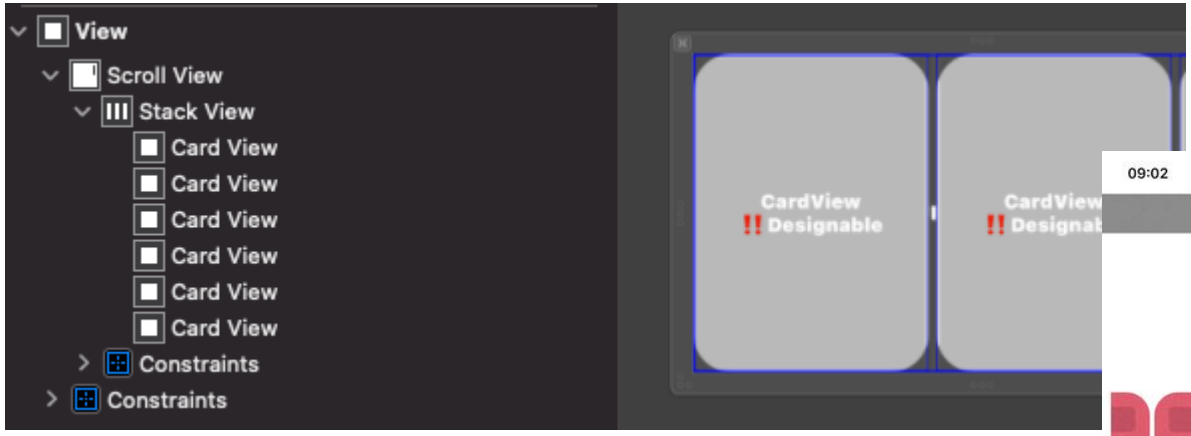
NIB - XIB

- NIB (NeXTSTEP Interface Builder)
- XIB (Xcode Interface Builder)
- Oba koncepta su vezana samo za UIkit i Objective-C
- Ideja je da omoguće kreiranje reupotrebljivih UI komponenti
- *.xib* fajl sadrži jedan pogled i može se koristiti u *Interface Builder-u* za vizuelno kreiranje UI



NIB - XIB

- Praktična upotreba je za kreiranje custom komponenti koje će se koristiti na više mesta



- Inicijalizacija

```
func standartInit() {
    let nib = UINib(nibName: CardsScrollView.indentifier, bundle: .main)

    let view = nib.instantiate(withOwner: self, options: nil).first as? UIView ?? UIView()
```

NIB - XIB

- Prednosti
  - Vrlo slično XML pristupu u Android-u
    XML + Activity vs. *XIB + Controller*
  - Mogu da postoje različiti XIB-ovi za različite lokalizacije
  - NIB-ovi koriste lazy-load pa ne zauzumaju memoriju dok nisu potrebni
  - NIB-ovi koriste vizuelni alat za kreiranje
- Mane
  - Komplikovano je koristiti NIB-ove za prikaz dinamičkog sadržaja
  - Kolaborativni rad na istom XIB-u je komplikovan (merge conflicts)
  - Sporije od programskog kreiranja zbog lazy-load pristupa
  - Čuva se u složenim XML fajlovima pa je reuse komplikovan

UIKit framework

- Elementi iOS UI mogu se podeliti u 4 glavne kategorije:
- *Bar-ovi* – statusne trake
  - Statusna traka
  - Navigaciona traka
  - Traka sa alatima
  - Tab traka
- Prikaz sadržaja
- Kontrole
- Privremeni prikaz sadržaja (dijalog box-ovi)

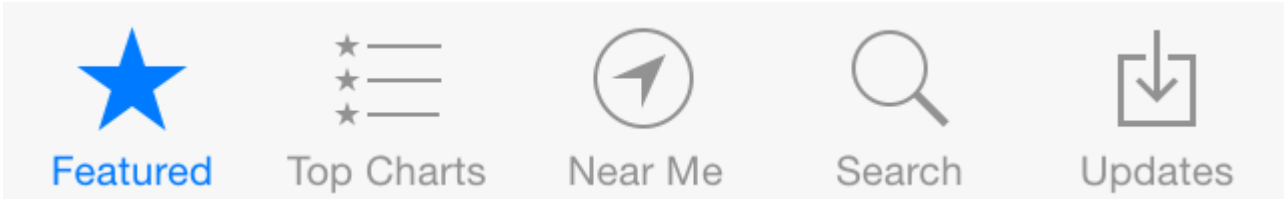
Bar-ovi (statusne trake)

Status bar

Navigation bar

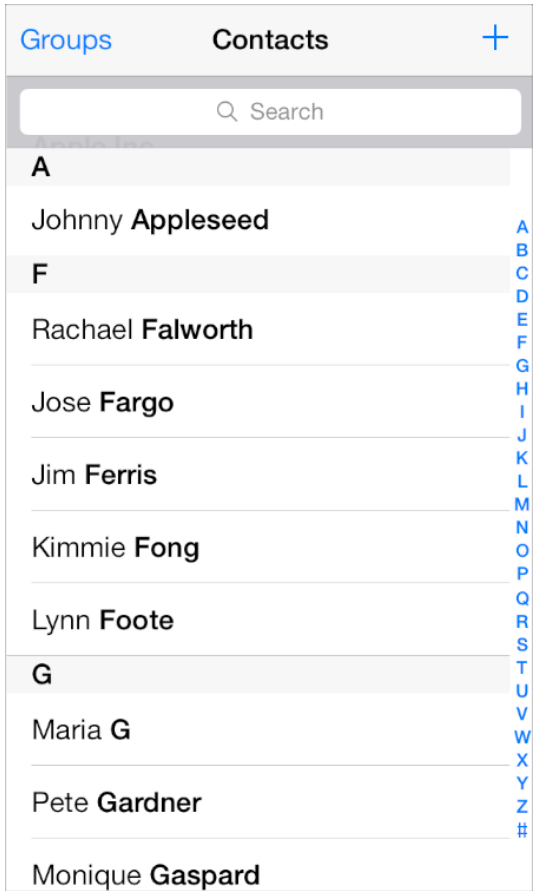
Tool bar

Tab bar

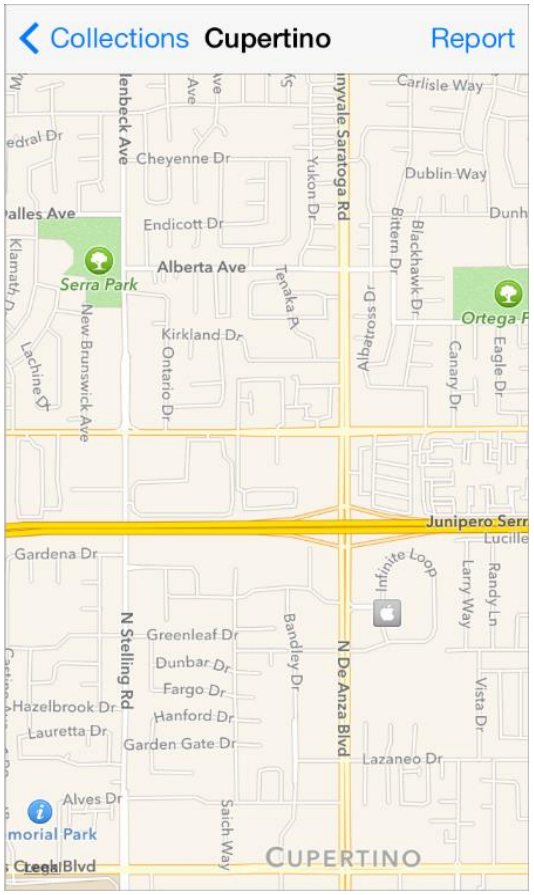


Prikaz sadržaja

- Definisan je mnogim podkategorijama:
  - Aktivnosti
  - Prikaz kolekcija
  - Prikaz slika
  - Prikaz mape
  - Prikaz skrol funkcije
  - Prikaz tabele
  - itd.



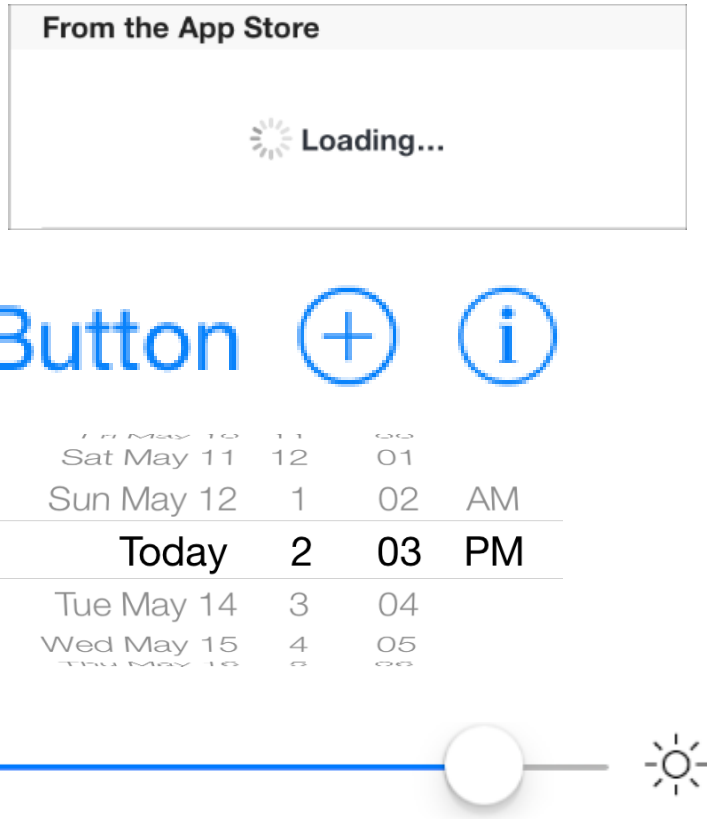
Prikaz tabele



Prikaz mape

UI kontrole

- Kontrole na UI su određene predefinisanim elementima:
  - Indikator aktivnosti
  - Dugmići
  - Izbornik datuma
  - Naslovi
  - Indikator mrežne aktivnosti
  - Kontrola strane
  - Traka progresa
  - Slajder
  - Prekidač
  - ...



iOS - new UI

- SwiftUI
  - https://developer.apple.com/xcode/swiftui/
  - Approach to developing user interfaces for all Apple platforms from watchOS up to tvOS for large TV displays.
  - SwiftUI uses a declarative syntax which means it helps to describe what happens after inputs and UI respond to these actions.
- Interface Builder
  - https://developer.apple.com/xcode/interface-builder/
  - Within Xcode makes it simple to design a full user interface without writing any code.
  - Simply drag and drop windows, buttons, text fields, and other objects onto the design canvas to create a functioning user interface.
- Catalyst
  - https://developer.apple.com/mac-catalyst/
  - Unites the mobile and the desktop platforms supposed to make it easy to port iPadOS apps to macOS by adapting the user interface and the interpretation of the inputs.
  - For example, touch gestures are automatically translated into mouse interactions.

SwiftUI

- Koristi deklarativni stil opisa UI-a, slično kao Jetpack Compose na Android-u
- SwiftUI podržava samo iOS 13 ili noviji

Framework	Swift	UIkit
Declarative syntax	✓	✗
Previews and live editing	✓	✗
State and data binding	✓	✗
Reactive programming	✓	✗
View controllers	✗	✓
Interface builder and storyboards	✗	✓
Delegates and data sources	✗	✓

SwiftUI

- SwiftUI je lansiran 2019
- Interaktivan je, pa se izmene vide at design-time
- Deklarativna sintaksa

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundColor(.accentColor)
            Text("Hello, world!")
        }
        .padding()
    }
}
```

SwiftUI

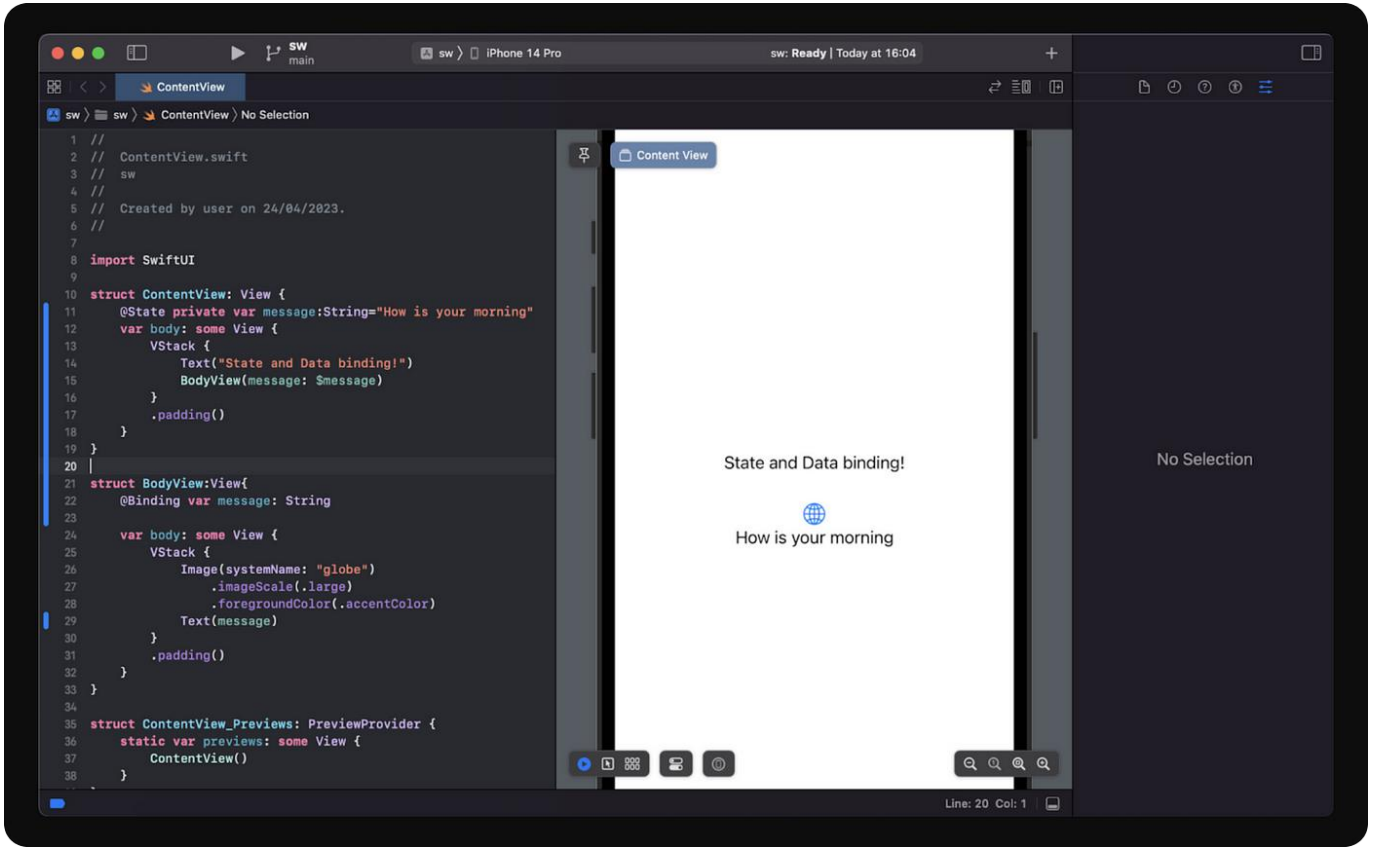
- Uvodi *State*
  - State je wrapper za property čije izmene treba prikazati na UI
- Uvodi *Data binding*
  - Data binding povezuje dva podatka kod kojih izmena jednog utiče na drugi

```
import SwiftUI

struct ContentView: View {
    @State private var message:String="How is your morning"
    var body: some View {
        VStack {
            Text("State and Data binding!")
            BodyView(message: $message)
        }
        .padding()
    }
}
```

```
struct BodyView:View{
    @Binding var message: String

    var body: some View {
        VStack {
            Image(systemName: "globe")
                .imageScale(.large)
                .foregroundColor(.accentColor)
            Text(message)
        }
        .padding()
    }
}
```

SwiftUI

- Preview i Live-editing prikazuje izmene i bez startovanja aplikacije

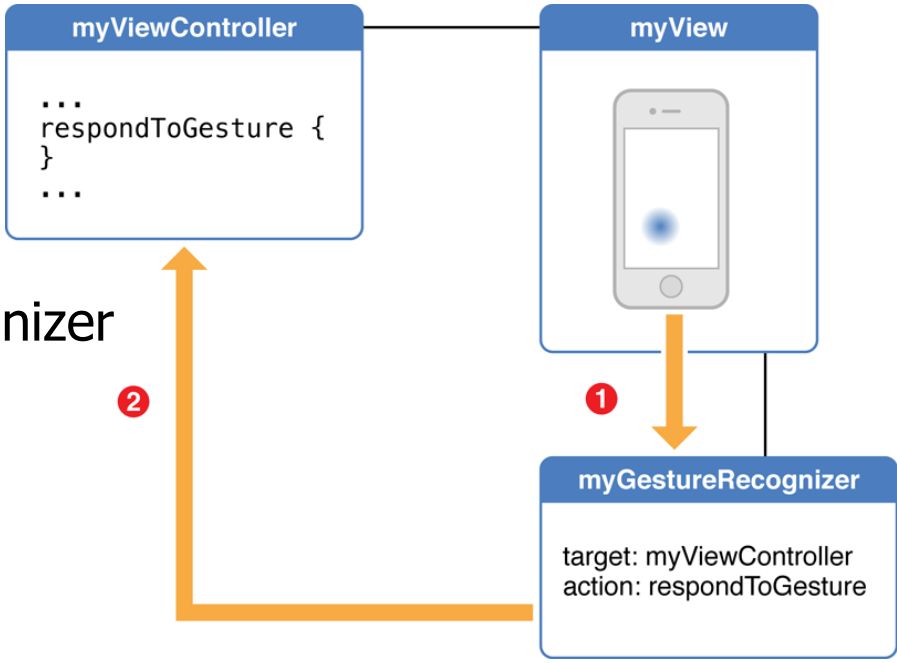


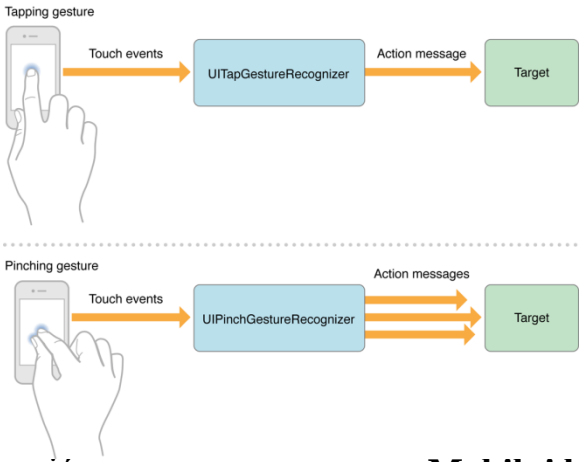
SwiftUI

- Prednosti
  - Unapredjeni programski koncept
  - Preview odmah prikazuje svaku izmenu
  - Lakši za kolaborativni rad sa code-versioning sistemima
  - Kombinacija Storyboard i programskog pristupa
- Mane
  - Radi samo na iOS 13 i novijim
  - Manji broj dostupnih UI kontrola

iOS - Prepoznavanje gestova

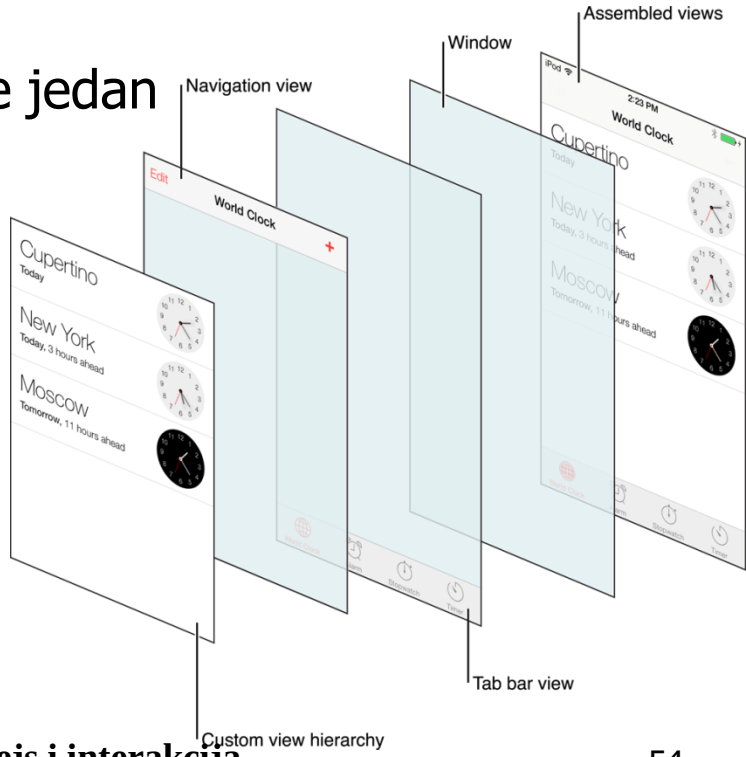
- iOS **UIGestureRecognizer** – apstraktna klasa za konkretne klase za prepoznavanje gestova
  - UI**Tap**GestureRecognizer
  - UI**Pinch**GestureRecognizer
  - UI**Rotation**GestureRecognizer
  - UI**Swipe**GestureRecognizer
  - UI**Pan**GestureRecognizer
  - UI**ScreenEdgePan**GestureRecognizer
  - UI**LongPress**GestureRecognizer





iOS - Prepoznavanje gestova (2)

- Views – gradivni blokovi za konstruisanje UI
- Odgovor na *Touch events*
  - touchesBegan:withEvent: metod kada jedan ili više prstiju dodirne ekran.
  - touchesMoved:withEvent: metod kada se jedan ili više prstiju pokreću.
  - touchesEnded:withEvent: metod kada se jedan ili više prstiju podignu sa ekrana
  - touchesCancelled:withEvent: metod kada se sekvenca dodira prekine nekim sistemskim događajem, npr. telefonskim pozivom.
- Odgovor na *Motion events*
  - motionBegan:withEvent:
  - motionEnded:withEvent:
  - motionCancelled:withEvent:



iOS *touch* gestovi

- Tap - To press or select a control or item.
- Drag - To scroll or pan—that is, move side to side. To drag an element.
- Flick - To scroll or pan quickly.
- Swipe - With one finger, to return to the previous screen, to reveal the hidden view in a split view (iPad only), or the Delete button in a table-view row. With four fingers, to switch between apps on iPad.
- Double tap - To zoom in and center a block of content or an image. To zoom out (if already zoomed in).
- Pinch - Pinch open to zoom in; pinch close to zoom out.
- Touch and hold - In editable or selectable text, to display a magnified view for cursor positioning.
- Shake - To initiate an undo or redo action.

iOS – UI dizajn

- iOS Design Resources
  - https://developer.apple.com/design/
- iOS Human Interface Guidelines
  - https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

Mobilne Web aplikacije -UI dizajn

- Responsive Design
- Google - Mobile Friendly Websites
  - https://developers.google.com/webmasters/mobile-sites/
- Mobile Friendly Test
  - https://search.google.com/search-console/mobile-friendly

Literatura

- *Mobile Developer's Guide To The Galaxy*, 18th Edition, 2019
- *Mobile Developer's Guide to the 5th Dimension*, Wireless Industry Partnership Connector Inc., 2013. http://wip.org/download/Fifth_Dimension_v1.pdf
- *Mobile Design and Development*, Brian Fling, O'Reilly Media; 2009
- *Mobile Design Pattern Gallery*, 2nd Edition, Theresa Neil, O'Reilly Media, 2014

Pitanja i komentari