Seminář 2: Platforma iOS a Úvod do SwiftUI

Tvorba mobilních aplikací

Roman Vyjídáček

Obsah semináře

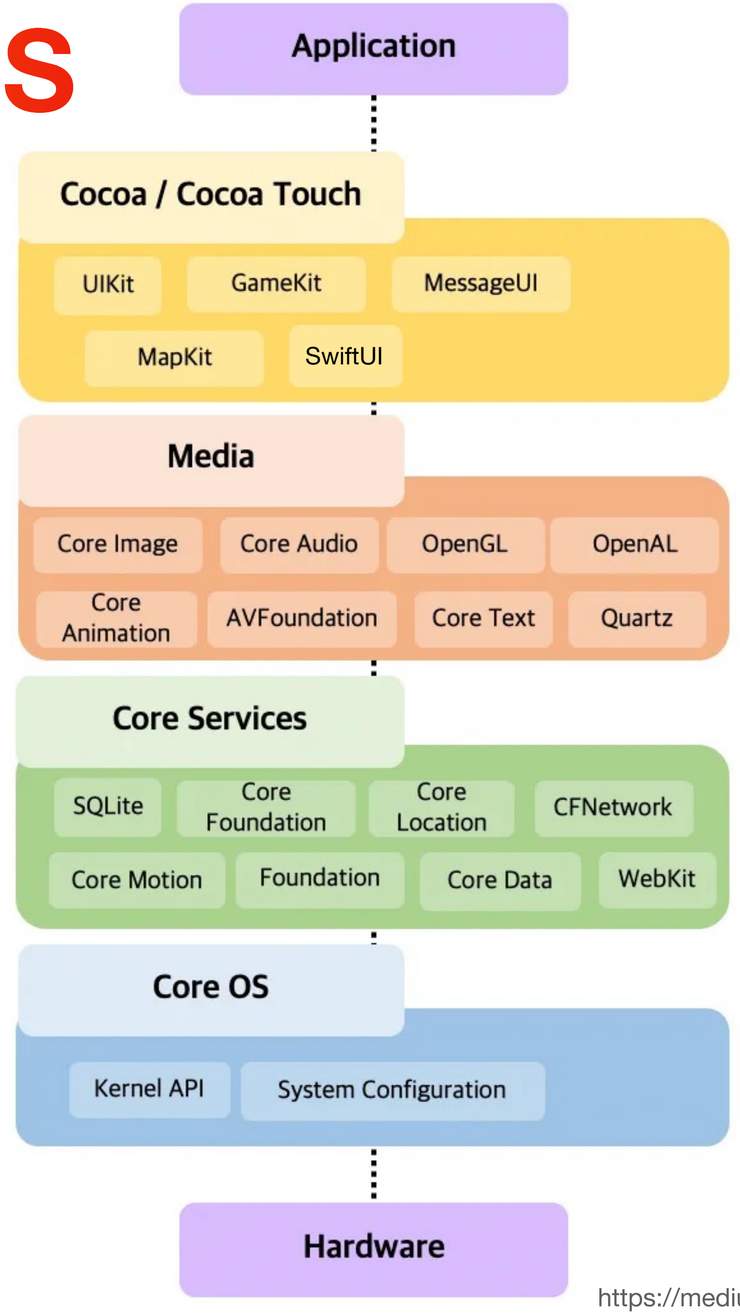
- Platforma iOS
- SwiftUI vs UIKit
- Základní komponenty

Platforma iOS

Operační systém

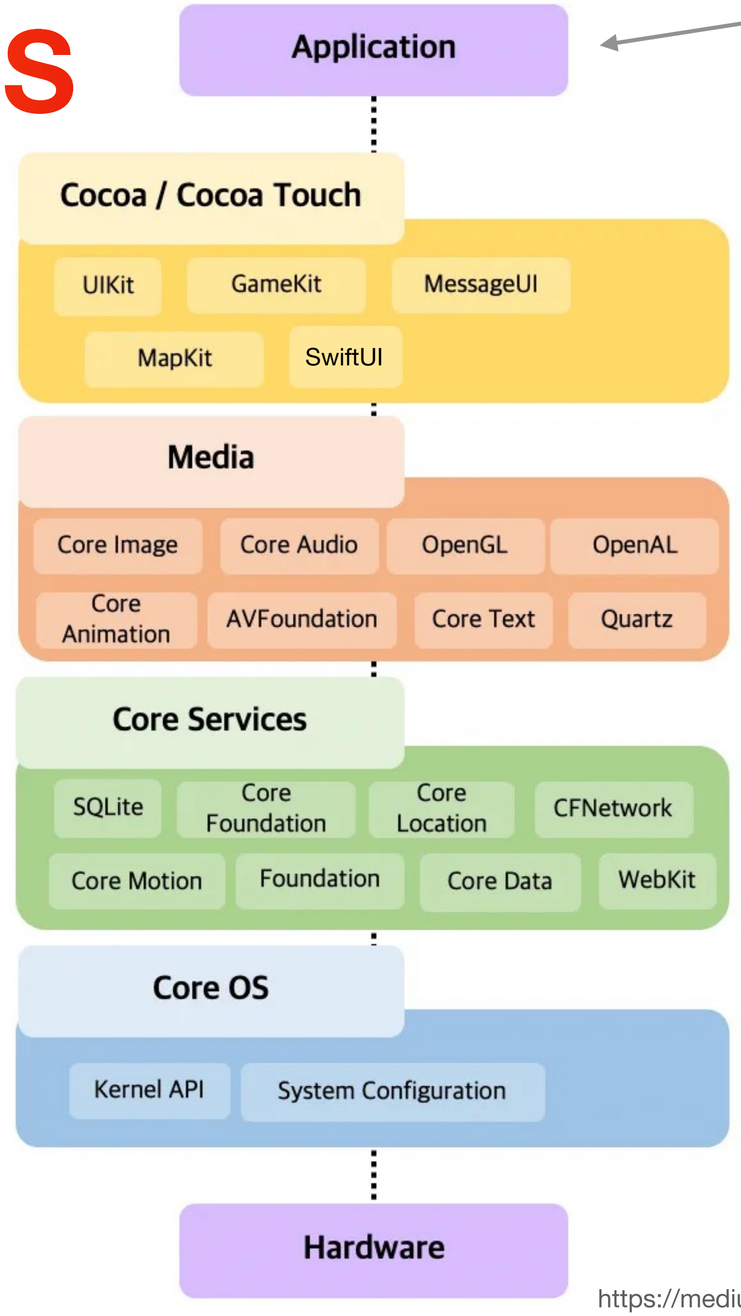
- Vychází z macOS (společný základ)
- Uzavřený ekosystém (kontrolovaný Apple)
- Sandboxovaný model aplikací
- Multitouch jako primární způsob interakce
- Multitasking řízený systémem (důraz na baterii)

Architektura iOS



https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS

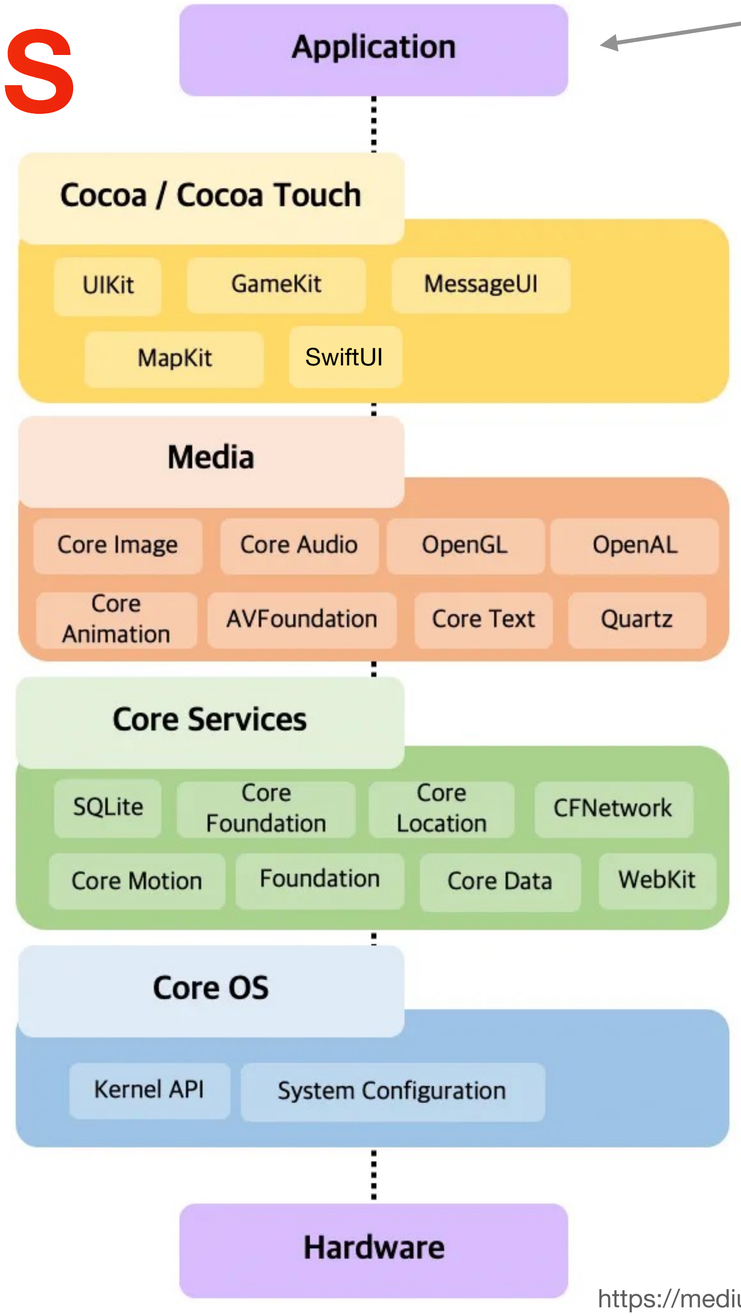


Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS

Cocoa / Cocoa Touch
Framework pro vývoj iOS aplikací, který zajišťuje práci s uživatelským rozhraním a interakcemi. Obsahuje knihovny jako SwiftUI pro zobrazení prvků, GameKit pro herní funkce, MessageUI pro zprávy a MapKit pro mapy.

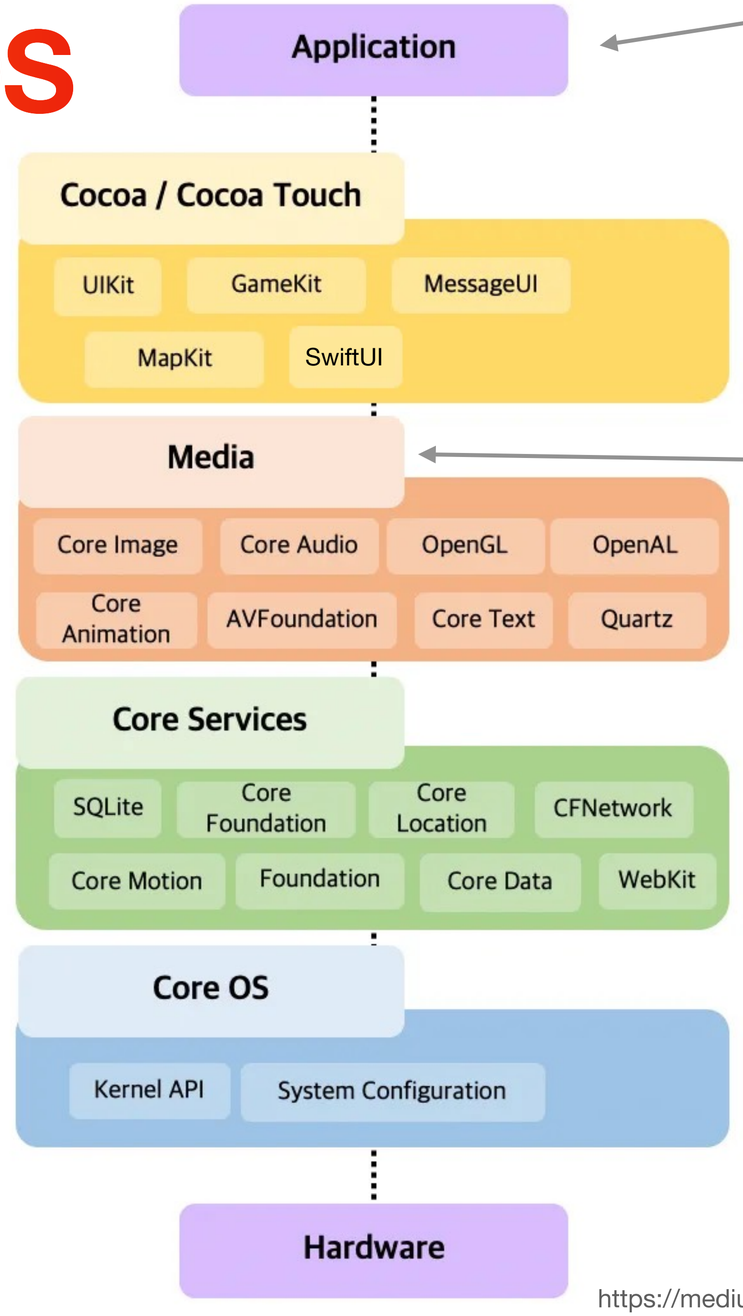


Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS

Cocoa / Cocoa Touch
Framework pro vývoj iOS aplikací, který zajišťuje práci s uživatelským rozhraním a interakcemi. Obsahuje knihovny jako SwiftUI pro zobrazení prvků, GameKit pro herní funkce, MessageUI pro zprávy a MapKit pro mapy.



Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

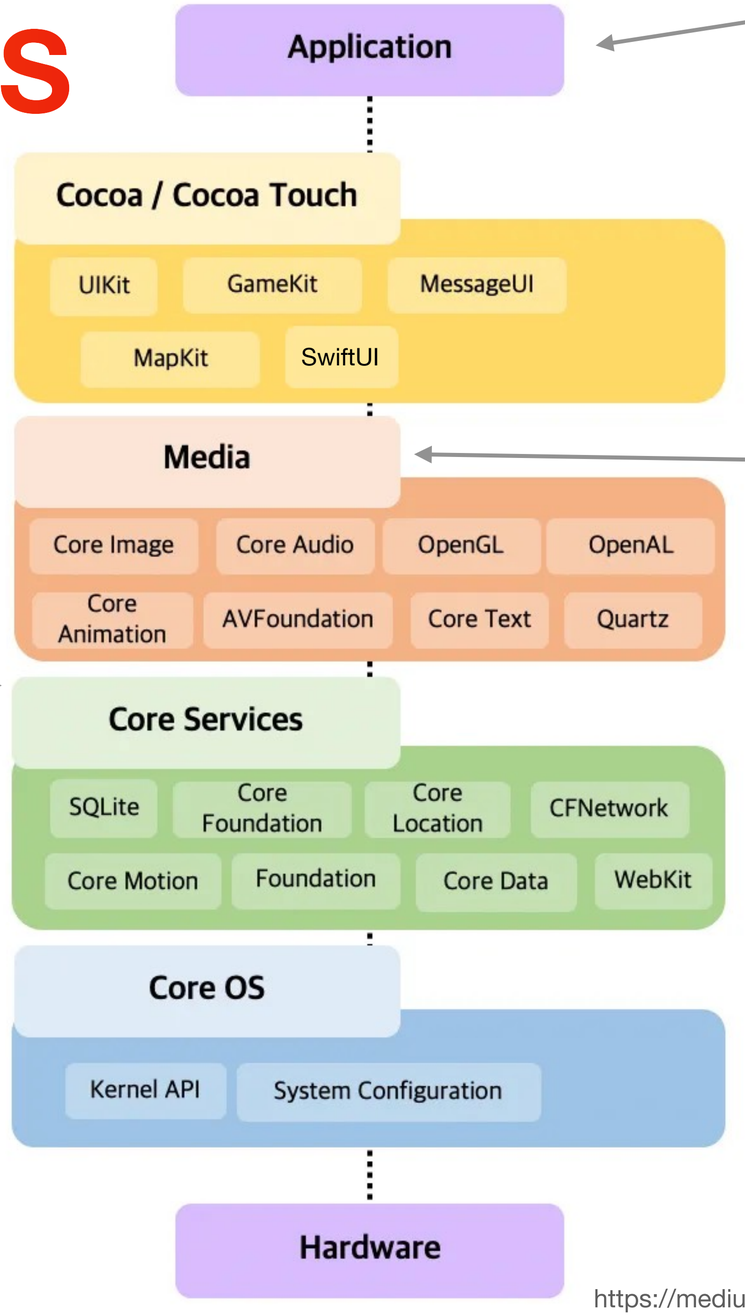
Media Layer
Vrstva zpracovávající multimédia – zvuk, video, grafiku a animace. Obsahuje technologie jako Core Image pro úpravy obrázků, Core Audio pro zvuk, OpenGL pro 3D grafiku a AVFoundation pro práci s videem.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS

Cocoa / Cocoa Touch
Framework pro vývoj iOS aplikací, který zajišťuje práci s uživatelským rozhraním a interakcemi. Obsahuje knihovny jako SwiftUI pro zobrazení prvků, GameKit pro herní funkce, MessageUI pro zprávy a MapKit pro mapy.

Core Services
Zajišťuje klíčové služby pro aplikace, jako je správa databází (SQLite, Core Data), síťová komunikace (CFNetwork), GPS lokalizace (Core Location) a senzorická data (Core Motion). Tato vrstva umožňuje aplikacím efektivně pracovat s daty a hardwarem.



Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

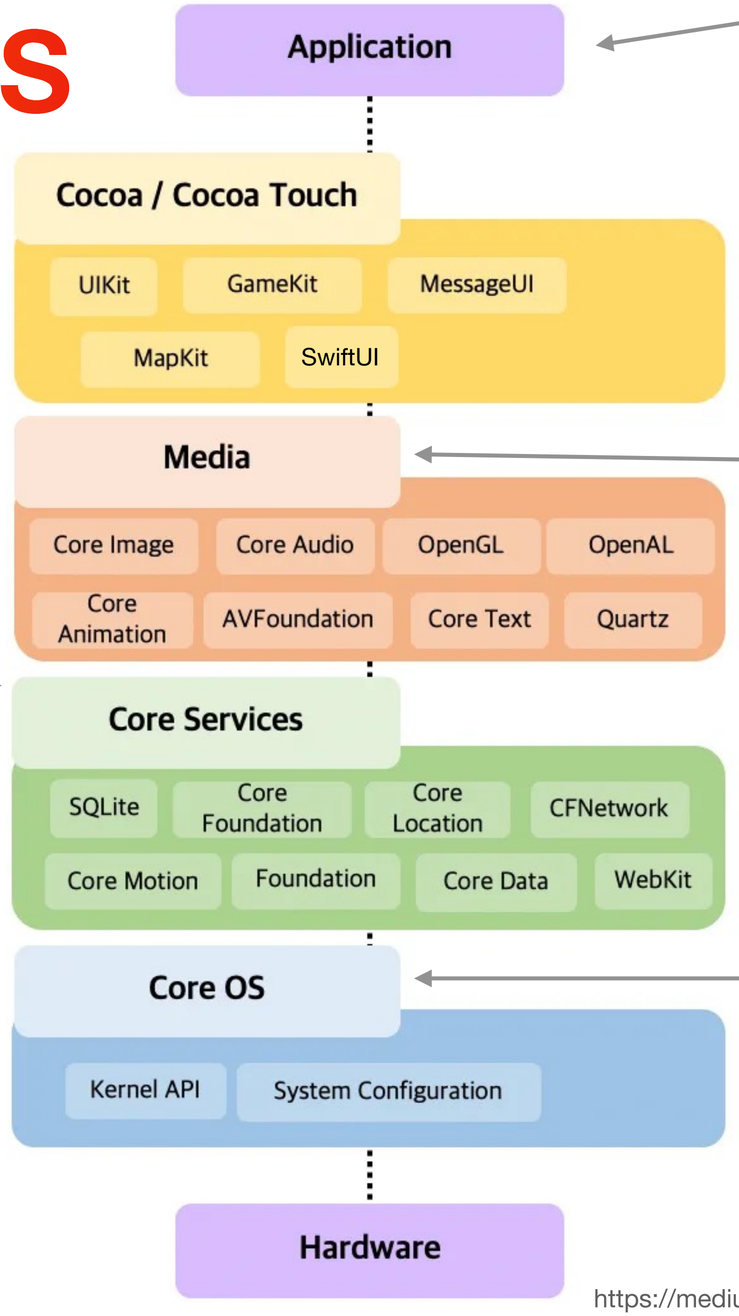
Media Layer
Vrstva zpracovávající multimédia – zvuk, video, grafiku a animace. Obsahuje technologie jako Core Image pro úpravy obrázků, Core Audio pro zvuk, OpenGL pro 3D grafiku a AVFoundation pro práci s videem.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS

Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

Cocoa / Cocoa Touch
Framework pro vývoj iOS aplikací, který zajišťuje práci s uživatelským rozhraním a interakcemi. Obsahuje knihovny jako SwiftUI pro zobrazení prvků, GameKit pro herní funkce, MessageUI pro zprávy a MapKit pro mapy.



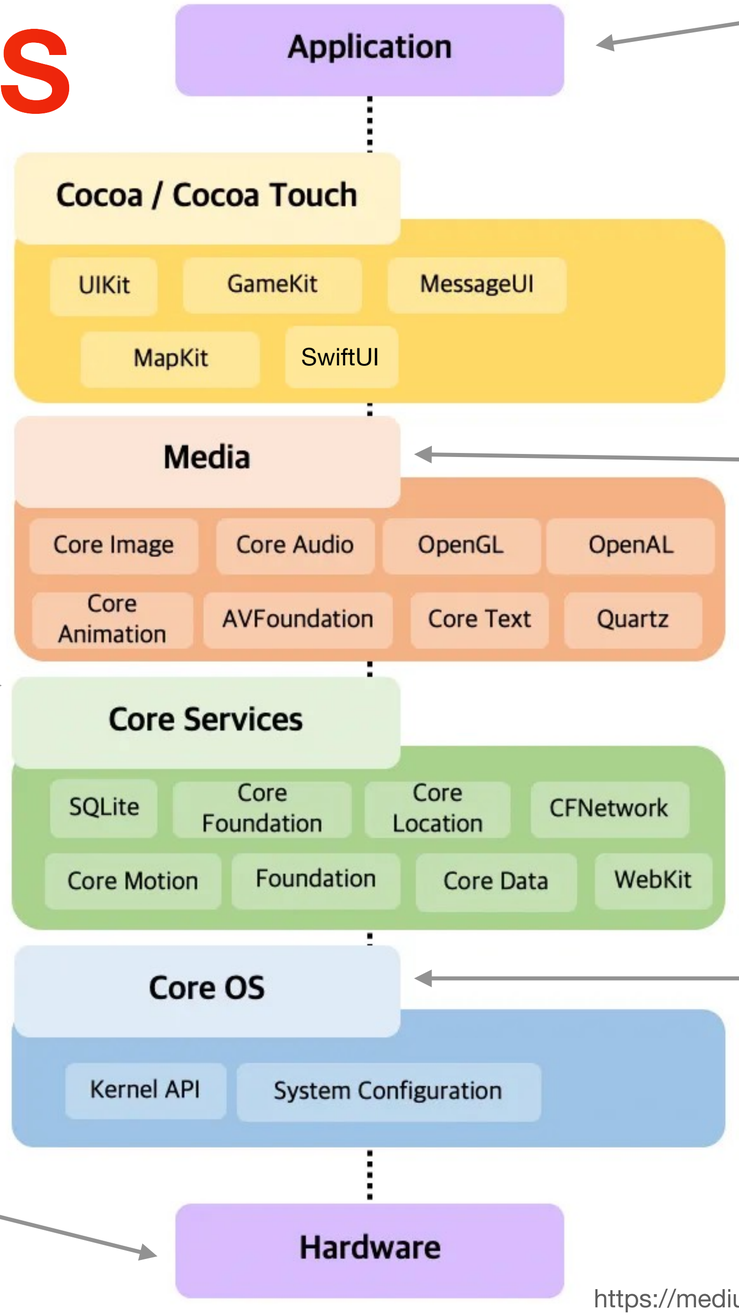
Media Layer
Vrstva zpracovávající multimédia – zvuk, video, grafiku a animace. Obsahuje technologie jako Core Image pro úpravy obrázků, Core Audio pro zvuk, OpenGL pro 3D grafiku a AVFoundation pro práci s videem.

Core Services
Zajišťuje klíčové služby pro aplikace, jako je správa databází (SQLite, Core Data), síťová komunikace (CFNetwork), GPS lokalizace (Core Location) a senzorická data (Core Motion). Tato vrstva umožňuje aplikacím efektivně pracovat s daty a hardwarem.

Core OS
Nejnižší softwarová vrstva, která komunikuje přímo s hardwarem. Obsahuje jádro operačního systému (Kernel API), které spravuje paměť, procesy a bezpečnost. System Configuration zajišťuje správu síťových nastavení a systémových oprávnění.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

Architektura iOS



Application Layer (Vrstva aplikací)
Tato vrstva představuje samotné aplikace, které uživatelé spouštějí na svých zařízeních. Komunikuje s nižšími vrstvami pomocí systémových frameworků a poskytuje uživatelské rozhraní a funkčnost aplikací.

Cocoa / Cocoa Touch
Framework pro vývoj iOS aplikací, který zajišťuje práci s uživatelským rozhraním a interakcemi. Obsahuje knihovny jako SwiftUI pro zobrazení prvků, GameKit pro herní funkce, MessageUI pro zprávy a MapKit pro mapy.

Media Layer
Vrstva zpracovávající multimédia – zvuk, video, grafiku a animace. Obsahuje technologie jako Core Image pro úpravy obrázků, Core Audio pro zvuk, OpenGL pro 3D grafiku a AVFoundation pro práci s videem.

Core Services
Zajišťuje klíčové služby pro aplikace, jako je správa databází (SQLite, Core Data), síťová komunikace (CFNetwork), GPS lokalizace (Core Location) a senzorická data (Core Motion). Tato vrstva umožňuje aplikacím efektivně pracovat s daty a hardwarem.

Core OS
Nejnižší softwarová vrstva, která komunikuje přímo s hardwarem. Obsahuje jádro operačního systému (Kernel API), které spravuje paměť, procesy a bezpečnost. System Configuration zajišťuje správu síťových nastavení a systémových oprávnění.

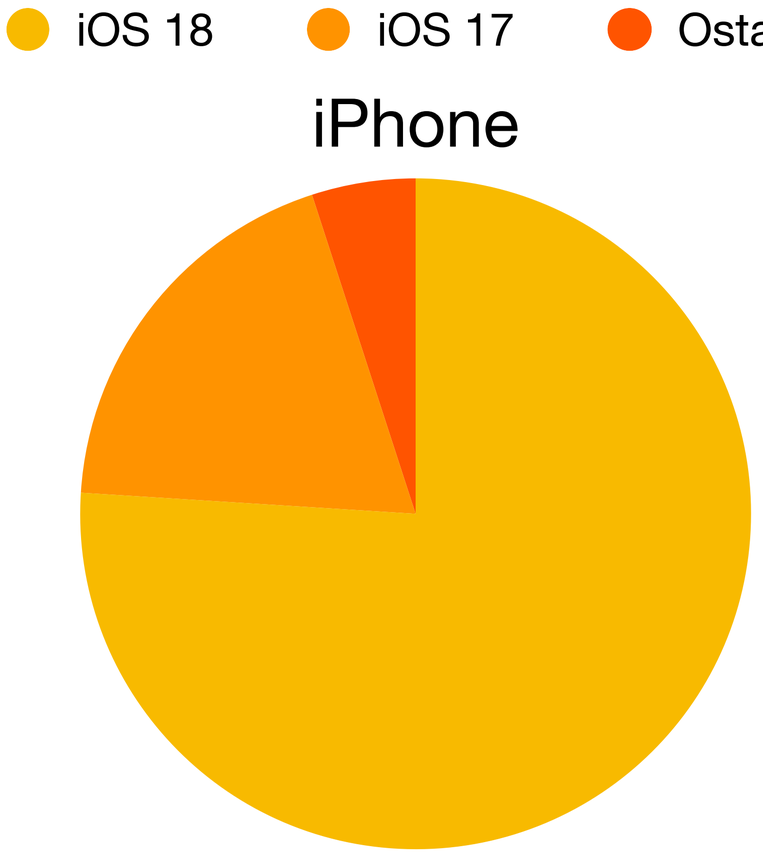
Hardware
Fyzická zařízení, na kterých běží iOS. Patří sem iPhone, iPad nebo Apple Watch. Tato vrstva spolupracuje s Core OS pro přímý přístup k procesoru, paměti, baterii a senzorům.

https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c

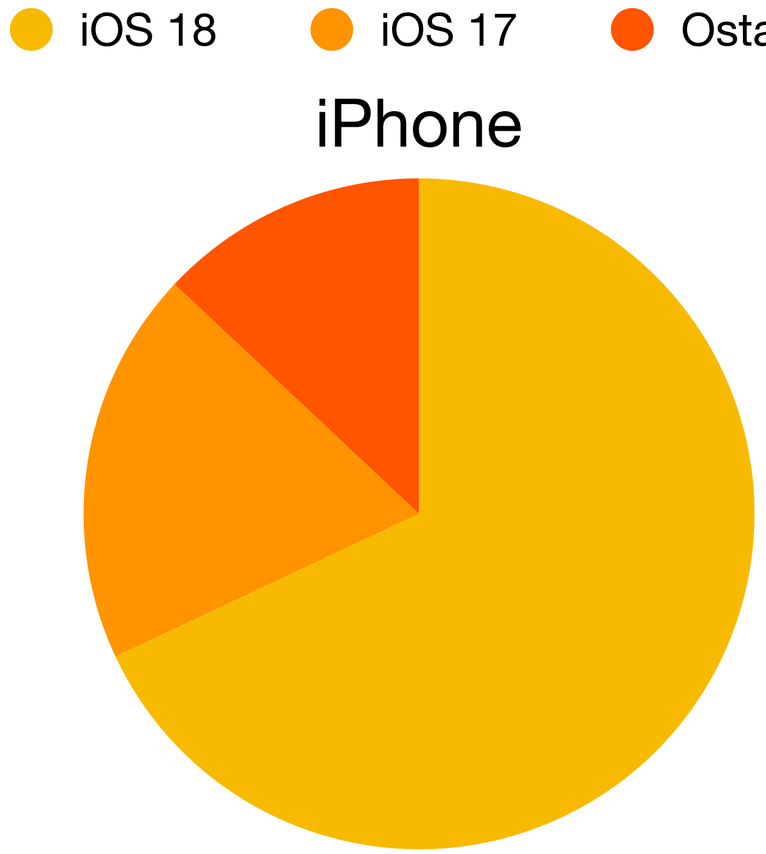
Životní cyklus iOS

- Nová verze iOS je představena na WWDC (červen) a vydána veřejně na podzim
- Během roku vycházejí minor a patch verze (opravy chyb, bezpečnostní aktualizace)
- Apple poskytuje podporu zařízení a OS typicky po dobu 5 let
- Nové verze iOS mají velmi rychlou adopci mezi uživateli

Zastoupení verzí iOS



Max. 4 roky staré zařízení



Zastoupení u všech zařízení

Zdroj: https://developer.apple.com/support/app-store/

Distribuce aplikací

- App Store (Developer Program za $99).
  - Provize Applu 30% (15%).
  - Každá aplikace musí projít review.
  - Je možnost si vyžádat i zrychlené review
- TestFlight, interní a externí testeři, public link.
- Interně (Enterprise Program za $299).

Způsoby monetizace

- Zdarma nebo placené.
- Jednorázové nákupy v aplikaci
- Předplatné

Přístupnost

- Guided Access
- VoiceOver
- Dynamic Type
- Grafika (tmavý režim, vysoký kontrast, inverzní barvy, snížená průhlednost, snížená barevnost, omezené animace)

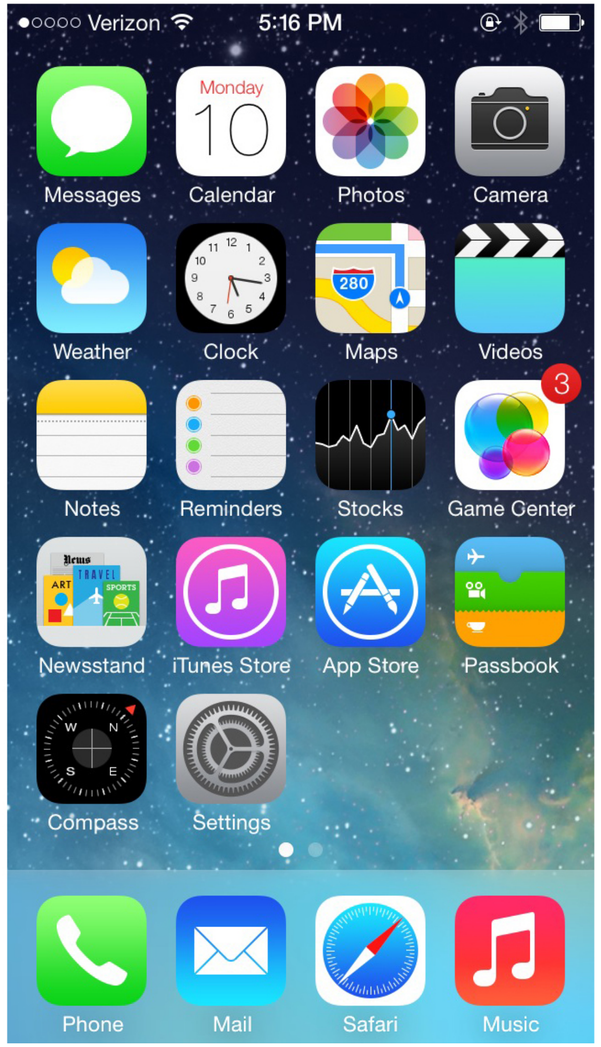
Bezpečnost

- Sandboxing
- Potvrzování přístupů jednotlivě
- Keychain
- App Transport Security (2015)
- Passcode, TouchID (2013), FaceID (2017)
- Šifrování obsahu (v HW)
- Apple ID účet s 2FA

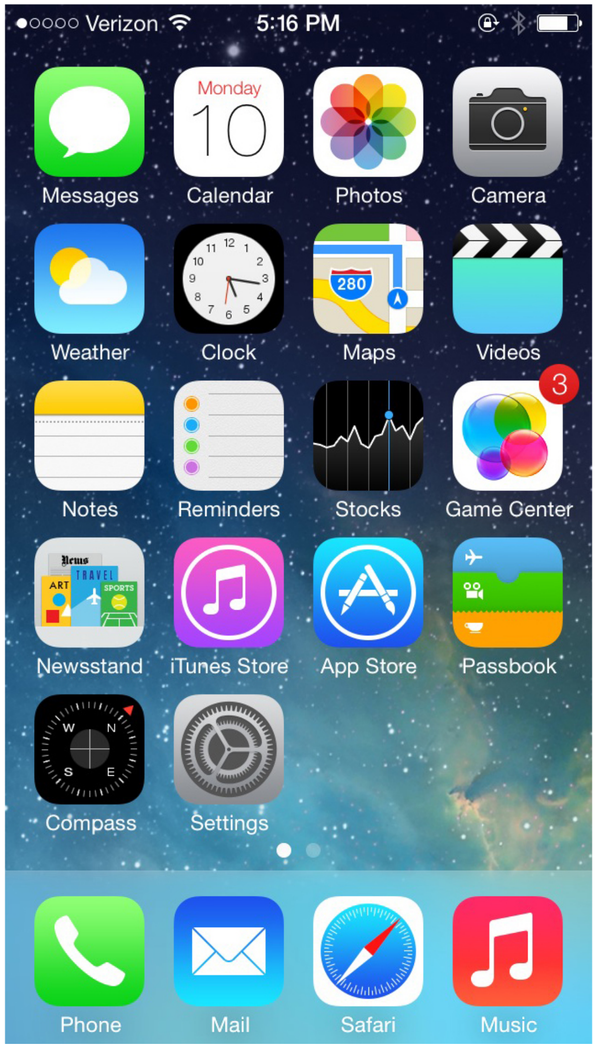
Uživatelské rozhraní

- Human Interface Guidelines
- Preference systémových komponent
- Primárně dotykové, lze mít periferie
- Widgety
- Notifikace
- Skeuomorphism → Flat design (iOS 7) → Liquid Glass (iOS 26)

Skeuomorphism → Flat design



Flat design→ Liquid Glass



SwiftUI vs UIKit

Co je UIKit?

- UIKit je imperativní framework pro tvorbu uživatelského rozhraní na platformě iOS.
- UI je tvořeno pomocí tříd a objektů, které vývojář explicitně vytváří a upravuje.
- Vývojář ručně řídí životní cyklus, rozložení prvků a aktualizaci UI.
- UIKit byl hlavní framework pro vývoj iOS aplikací od první verze iOS.
- Stále se používá v existujících aplikacích

Co je SwiftUI?

- SwiftUI je deklarativní framework pro tvorbu uživatelského rozhraní pro iOS, macOS, watchOS a tvOS
- Umožňuje popsat UI pomocí dat a stavu, nikoliv pomocí explicitních kroků
- Využívá Swift jako primární jazyk a integruje návrh UI přímo do kódu
- SwiftUI existuje vedle UIKit a postupně se stává preferovaným řešením
- UI je popsáno jako hierarchie View
- Změna dat automaticky aktualizuje UI

Deklarativní vs. Imperativní přístup

Rozdíl mezi popisem chování UI a popisem výsledného stavu UI

SwiftUI (Deklarativní)	UIKit (Imperativní)
Používá struktury (structs)	Používá třídy (classes)
UI je funkcí dat a stavu	Stav UI řízen explicitně vývojářem
Framework řeší aktualizace za vývojáře	Ruční aktualizace UI
Menší množství kódu	Více řádků kódu, nutná synchronizace

UIKit vs SwiftUI

```swift
import UIKit

class ViewController: UIViewController {
    override func loadView() {
        super.loadView()

        let label = UILabel()
        label.text = "Ahoj, světe!"
        label.font = UIFont.systemFont(ofSize: 34)
        label.translatesAutoresizingMaskIntoConstraints = false

        view.addSubview(label)
        view.backgroundColor = .white

        NSLayoutConstraint.activate([
            label.centerXAnchor.constraint(
               equalTo: view.centerXAnchor
            ),
            label.centerYAnchor.constraint(
               equalTo: view.centerYAnchor
            )
        ])
    }
}
```

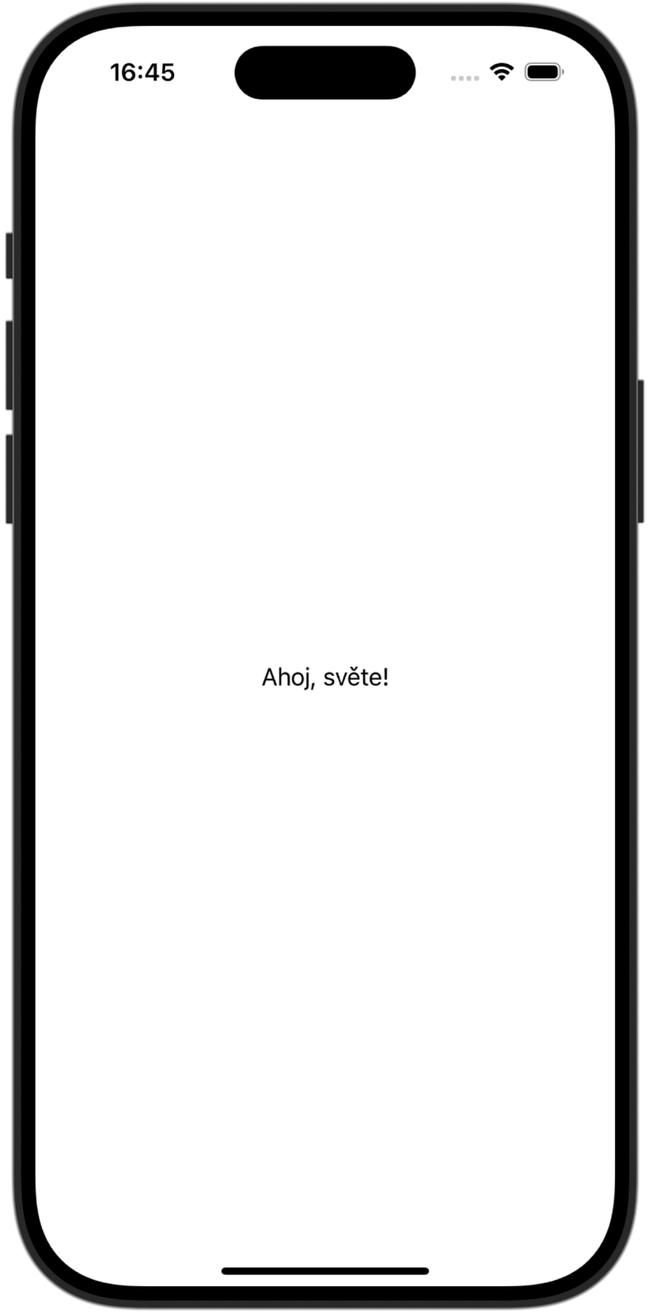
UIKit vs SwiftUI

```swift
import SwiftUI

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
            .font(.largeTitle)
            .padding()
            .frame(maxWidth: .infinity, maxHeight: .infinity)
            .background(Color.white)
    }
}
```

Základní komponenty

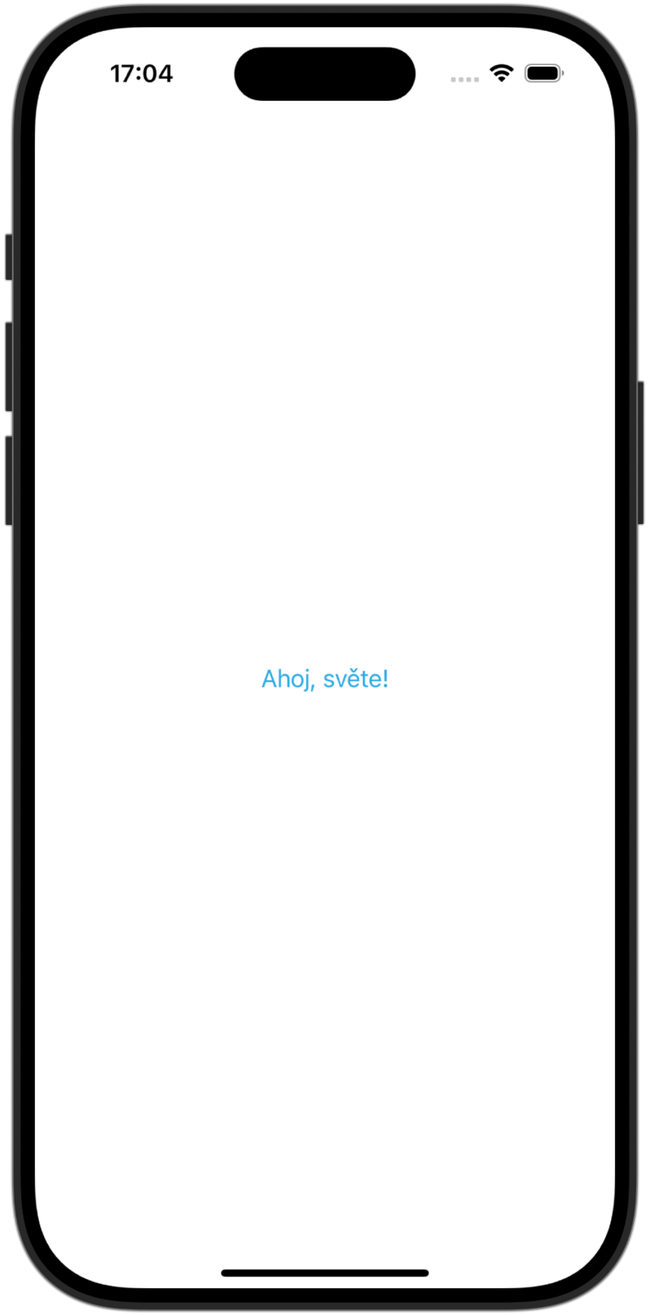
Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
    }
}
```



Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
//            Nastavíme barvu textu
            .foregroundStyle(.cyan)
    }
}
```



Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
//            Nastavíme barvu textu
            .foregroundStyle(.cyan)
//            Zvětšíme font
            .font(.largeTitle)
    }
}
```

Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
//            Nastavíme barvu textu
            .foregroundStyle(.cyan)
//            Zvětšíme font
            .font(.largeTitle)
//            Nastavíme barvu pozadí
            .background(.yellow)
    }
}
```

Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
//            Nastavíme barvu textu
            .foregroundStyle(.cyan)
//            Zvětšíme font
            .font(.largeTitle)
//            Nastavíme padding
            .padding()
//            Nastavíme barvu pozadí
            .background(.yellow)

    }
}
```

Text

```
// - Slouží k zobrazení statického textu

struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
//            Nastavíme barvu textu
            .foregroundStyle(.cyan)
//            Zvětšíme font
            .font(.largeTitle)
//            Nastavíme padding
            .padding()
//            Nastavíme barvu pozadí
            .background(.yellow)
//            Nastavíme zakulacení rohů
            .cornerRadius(16)
    }
}
```

Button

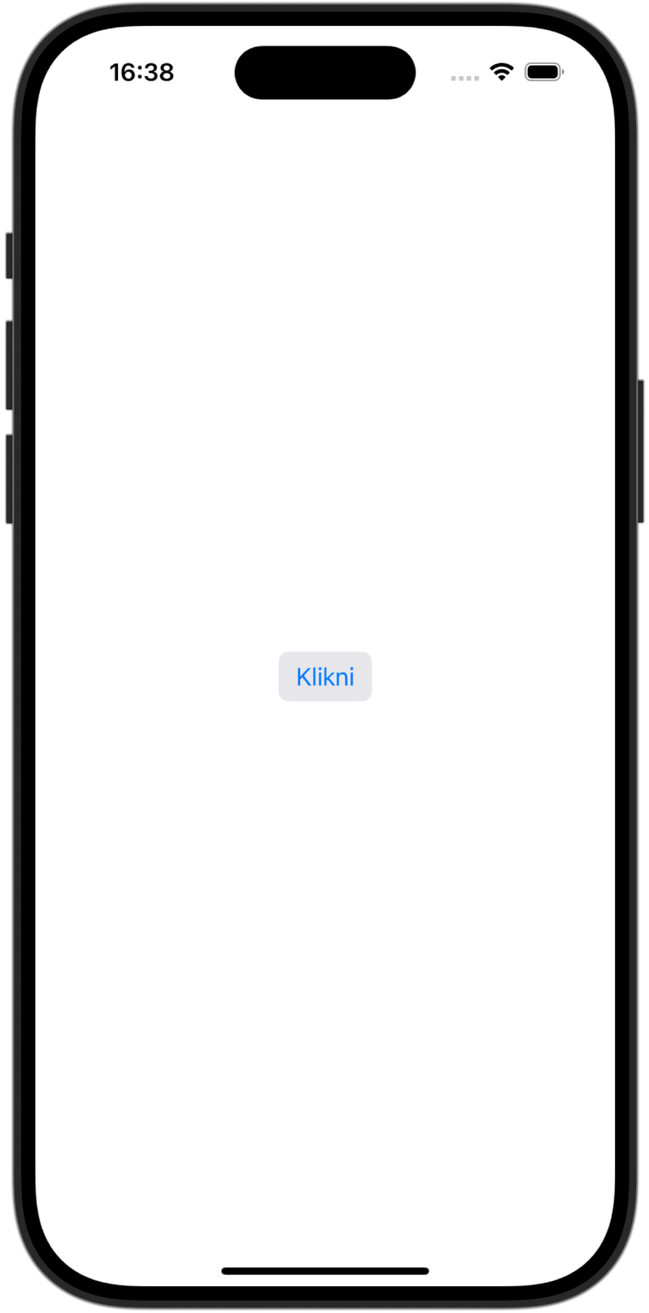
- Komponenta Button vytváří interaktivní tlačítko.

Základní parametry:

- *action*: Closure, která se vykoná po stisknutí tlačítka.
- *label*: Zobrazuje obsah tlačítka, obvykle Text, ale může obsahovat libovolné View.
- title: Text tlačítka

Button

```
struct ContentView: View {

    var body: some View {
        Button("Klikni") {
            print("Tlačítko stisknuto")
        }
//          UI styl tlačítka
        .buttonStyle(.bordered)
    }
}
```



Button

```
struct ContentView: View {

    var body: some View {
        Button(action: {
            print("Tlačítko stisknuto")
        }) {
            Text("Klikni zde")
                .font(.headline)
                .padding()
                .background(Color.green)
                .foregroundColor(.white)
                .cornerRadius(10)
        }
    }
}
```

Asset Catalog

- Slouží ke správě zdrojů aplikace
- Obsahuje obrázky, barvy a další assety
- Automaticky řeší různé velikosti obrazovek a režimy zobrazení

Image

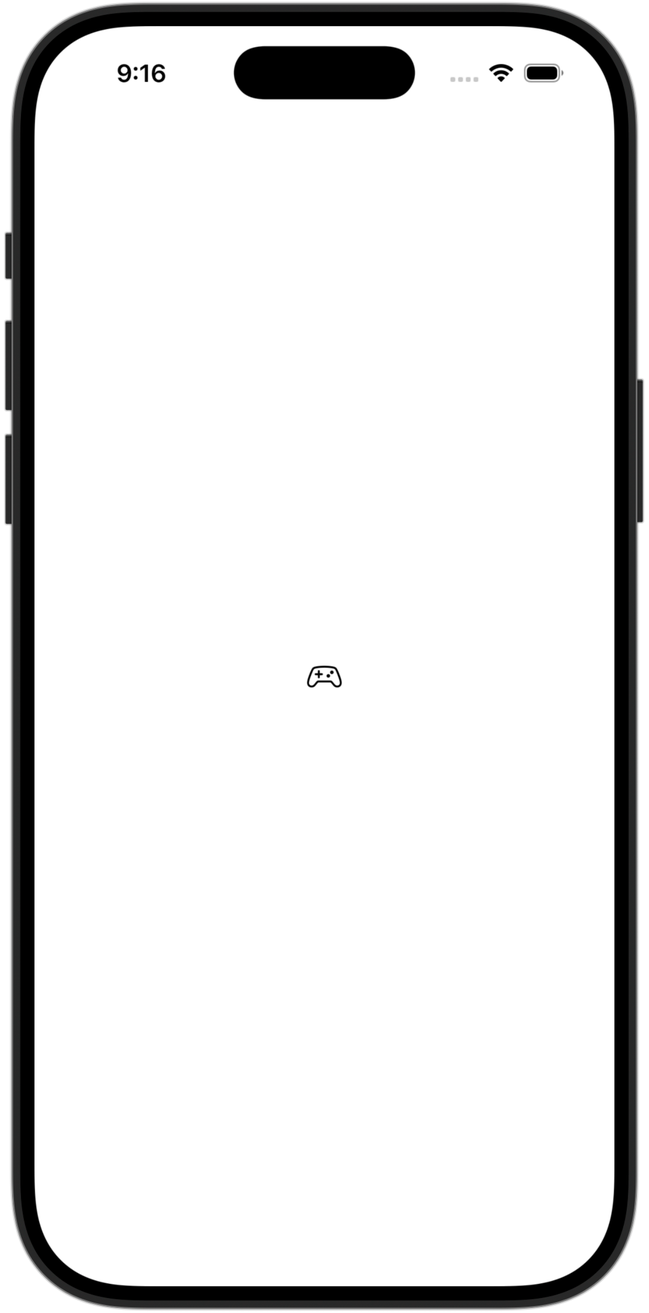
- Slouží k zobrazení obrázků v uživatelském rozhraní
- Podporuje obrázky z assets i systémové ikony (SF Symbols)

Základní parametry:

- *name*: Název obrázku z Asset Catalogu
- *systemName*: Název systémového obrázku (SF Symbols)

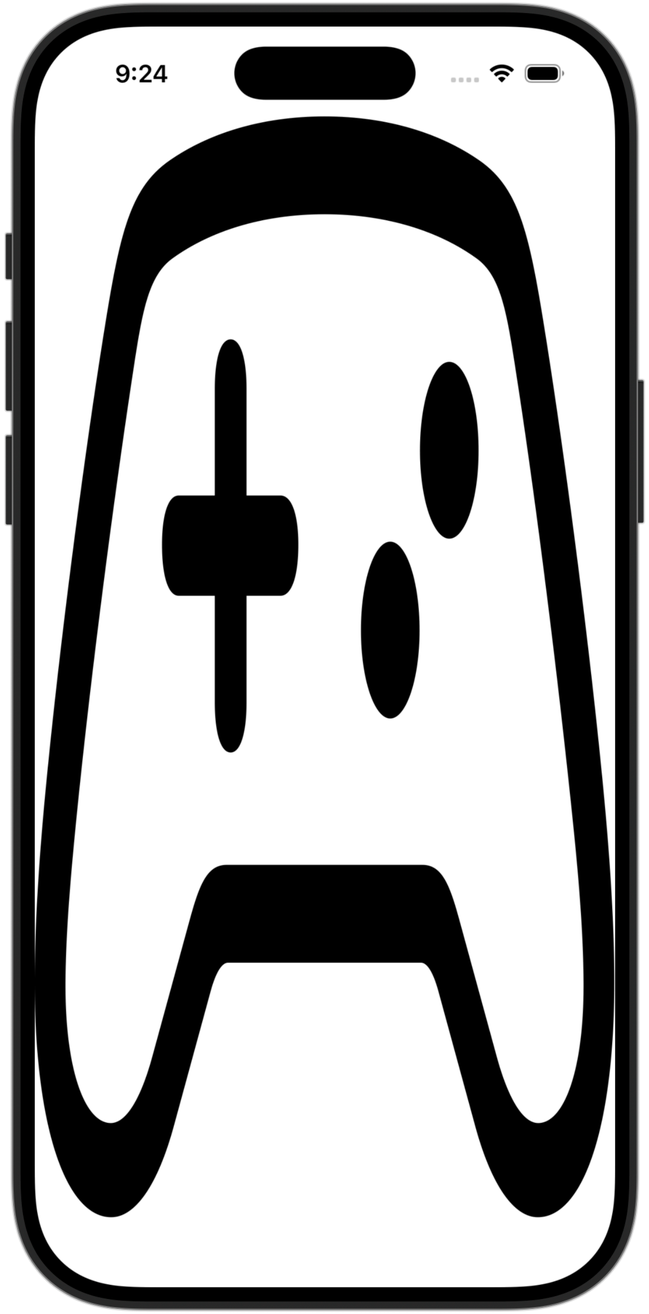
Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
    }
}
```



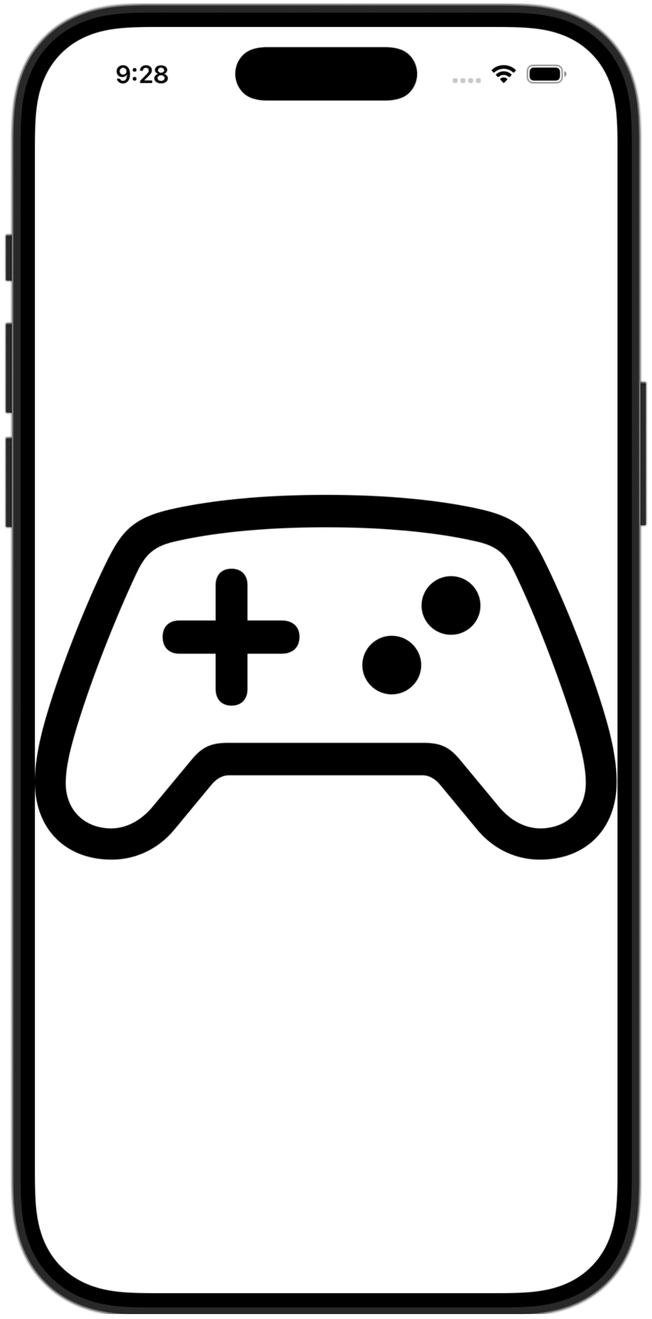
Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
    }
}
```



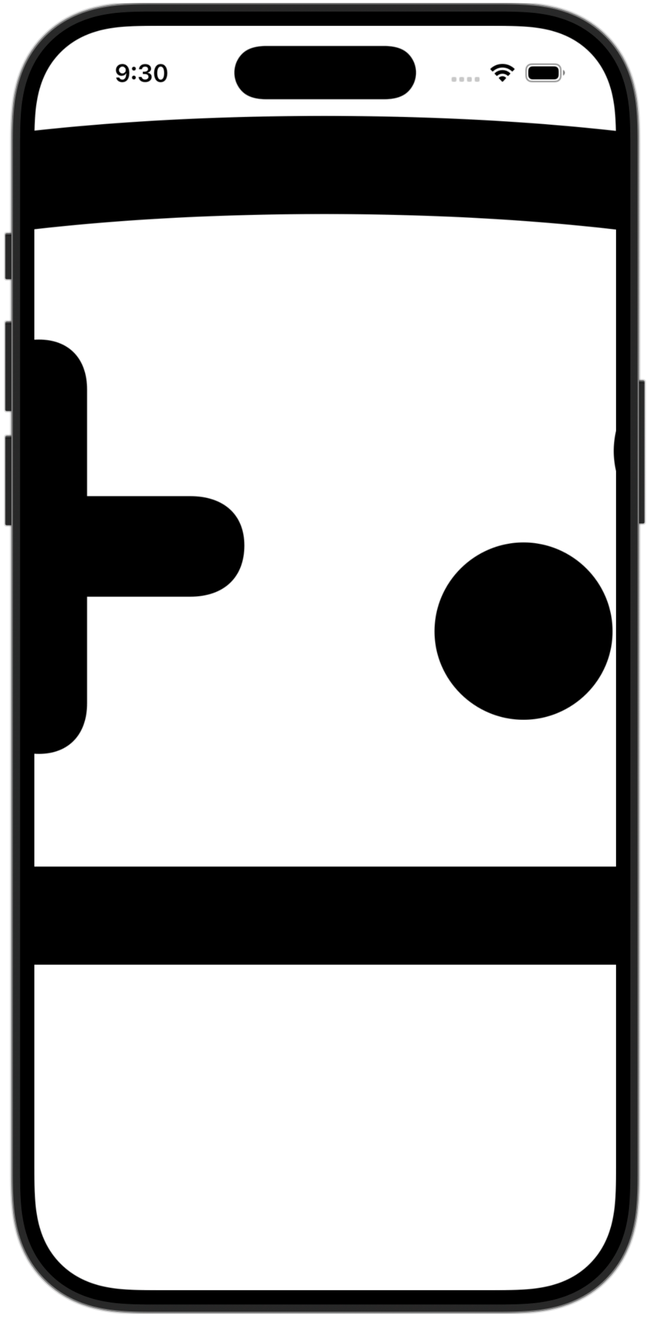
Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
            .scaledToFit()
    }
}
```



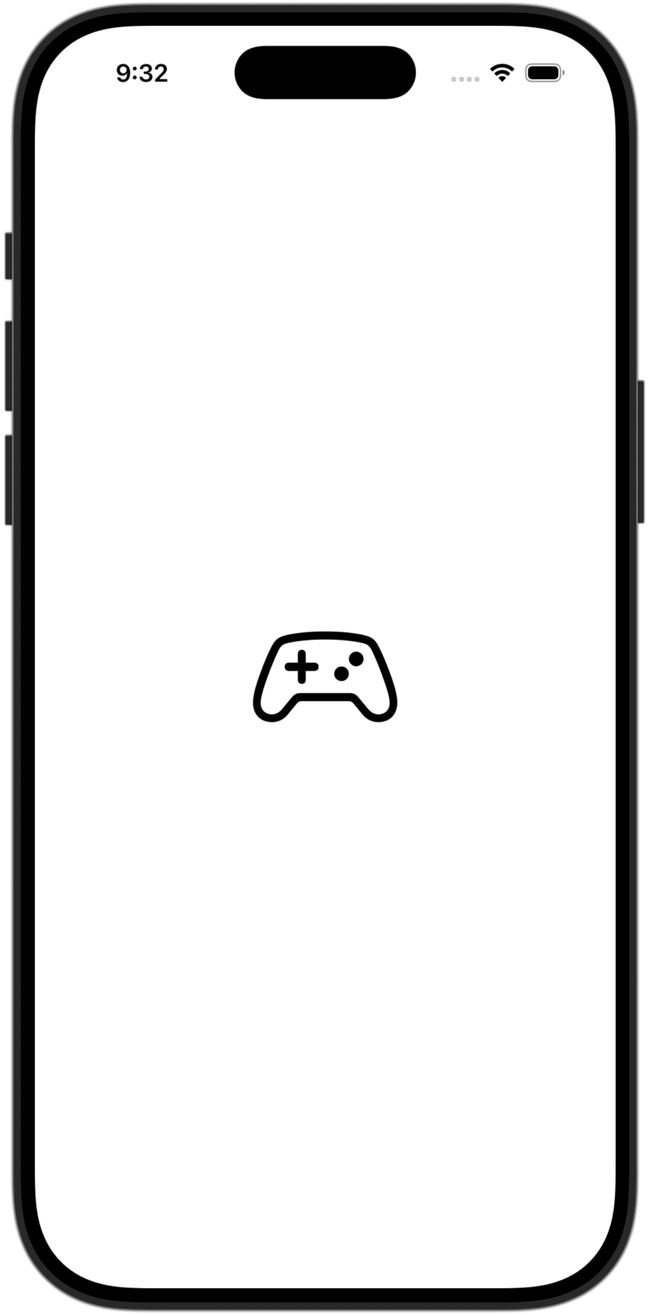
Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
            .scaledToFill()
    }
}
```



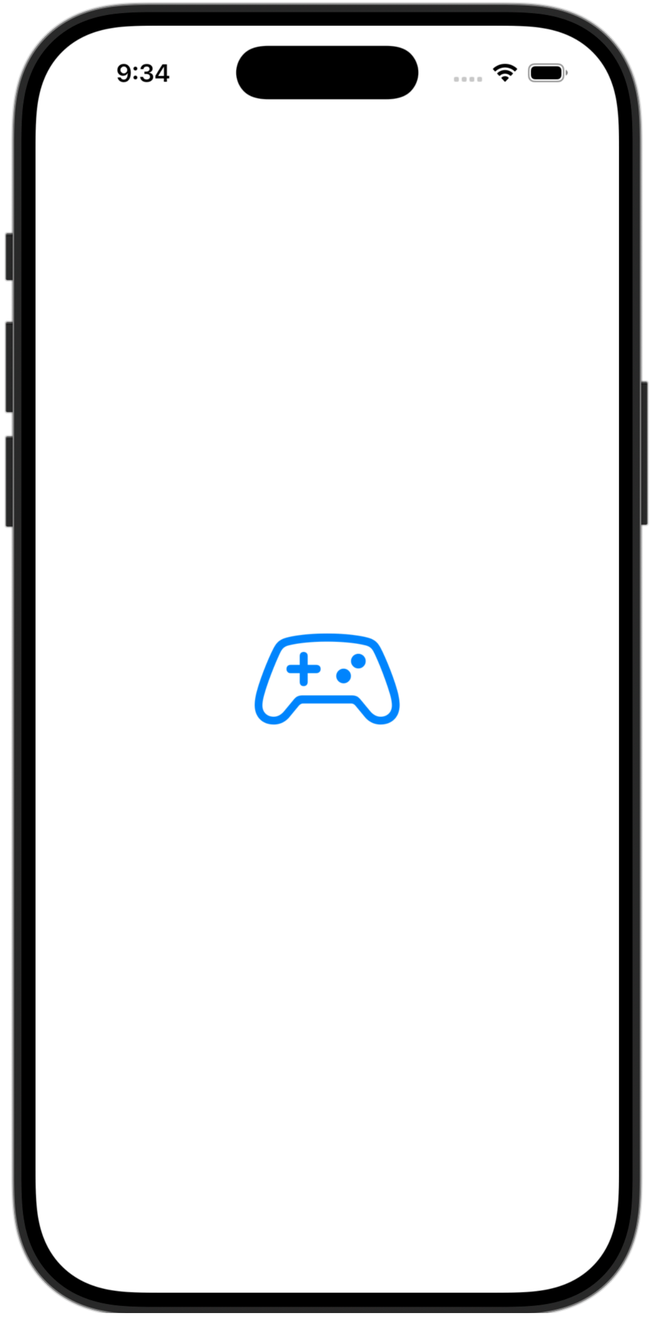
Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
            .scaledToFit()
            .frame(width: 100, height: 100)
    }
}
```



Image

```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
            .scaledToFit()
            .frame(width: 100, height: 100)
            .foregroundStyle(Color.blue)
    }
}
```



Image

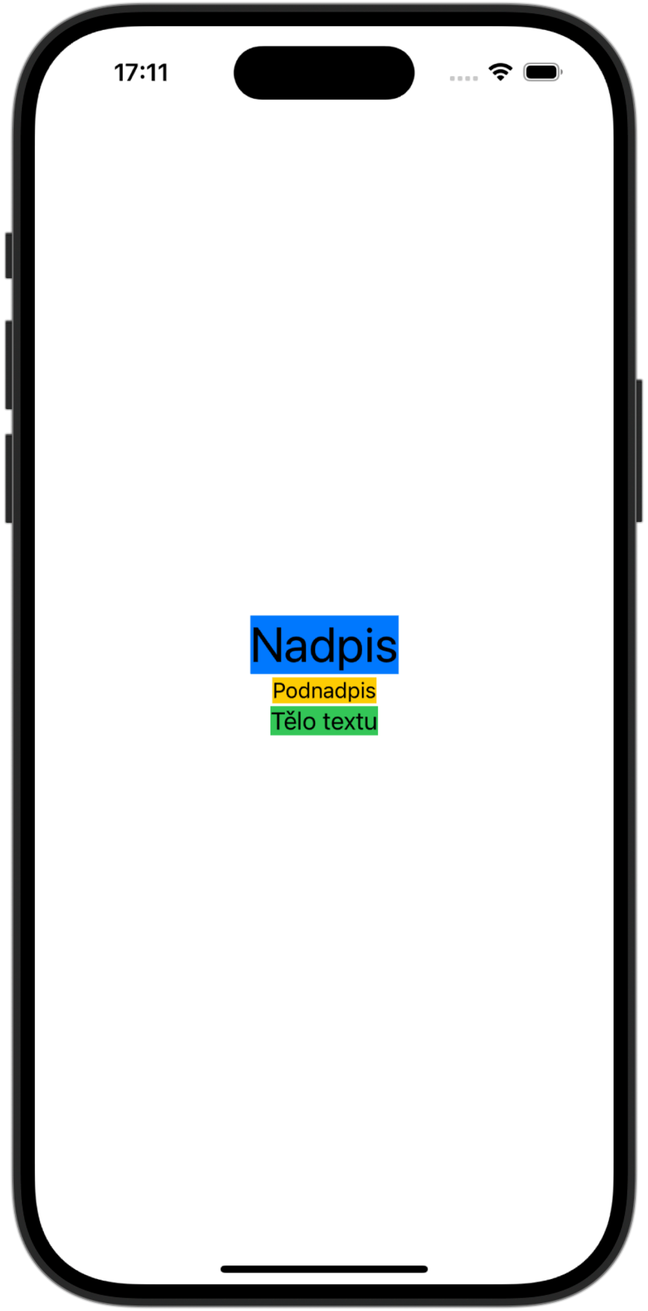
```
struct ContentView: View {

    var body: some View {
        Image(systemName: "gamecontroller")
            .resizable()
            .scaledToFit()
            .frame(width: 100, height: 100)
            .foregroundStyle(Color.blue)
            .padding(30)
            .background {
                Circle()
                    .fill(Color.gray.opacity(0.2))
            }
    }
}
```

Stacks (HStack, VStack, ZStack)

- Stacky slouží k uspořádání více Views do horizontální (HStack), vertikální (VStack) nebo překrývající se (ZStack) vrstvy.
- Základní parametry:
- *alignment:* Zarovnání Views (např. .leading, .center, .trailing).
- *spacing:* Odstup mezi podřízenými Views.

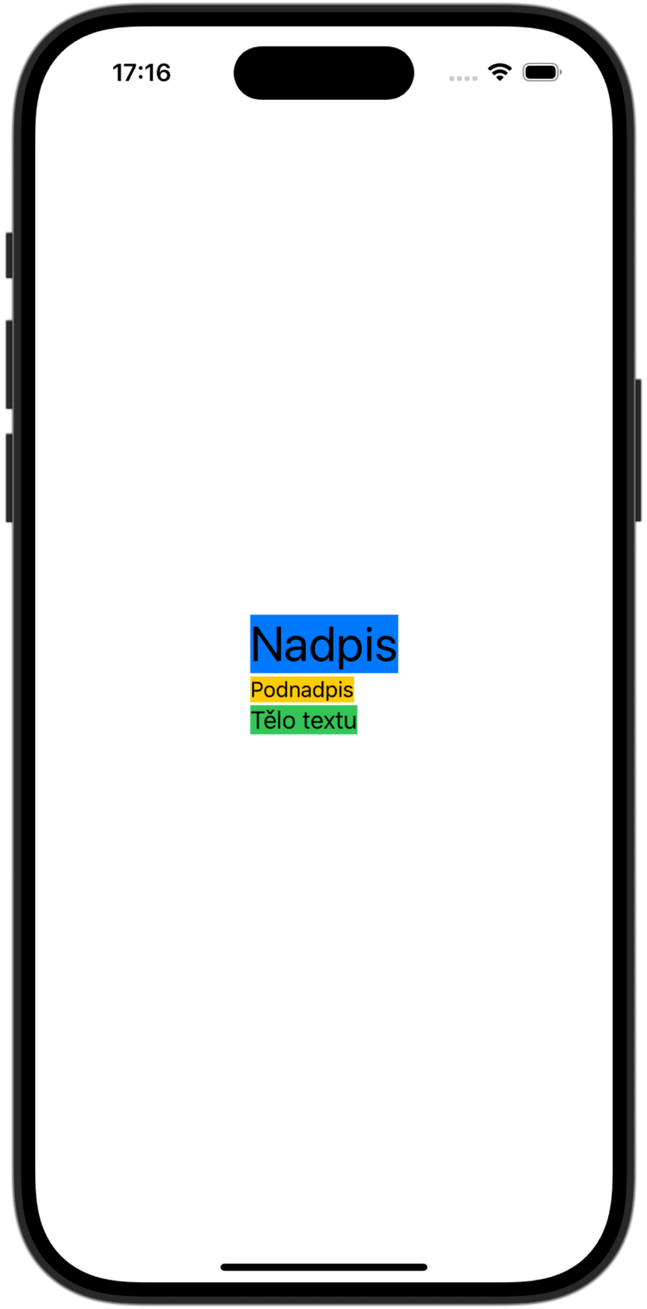
VStack

```
struct ContentView: View {

    var body: some View {
        VStack {
            Text("Nadpis")
                .font(.largeTitle)
                .background(Color.blue)
            Text("Podnadpis")
                .font(.subheadline)
                .background(Color.yellow)
            Text("Tělo textu")
                .font(.body)
                .background(Color.green)
        }
        .padding()
    }
}
```



VStack

```
struct ContentView: View {

    var body: some View {
        VStack(alignment: .leading) {
            Text("Nadpis")
                .font(.largeTitle)
                .background(Color.blue)
            Text("Podnadpis")
                .font(.subheadline)
                .background(Color.yellow)
            Text("Tělo textu")
                .font(.body)
                .background(Color.green)
        }
        .padding()
    }
}
```

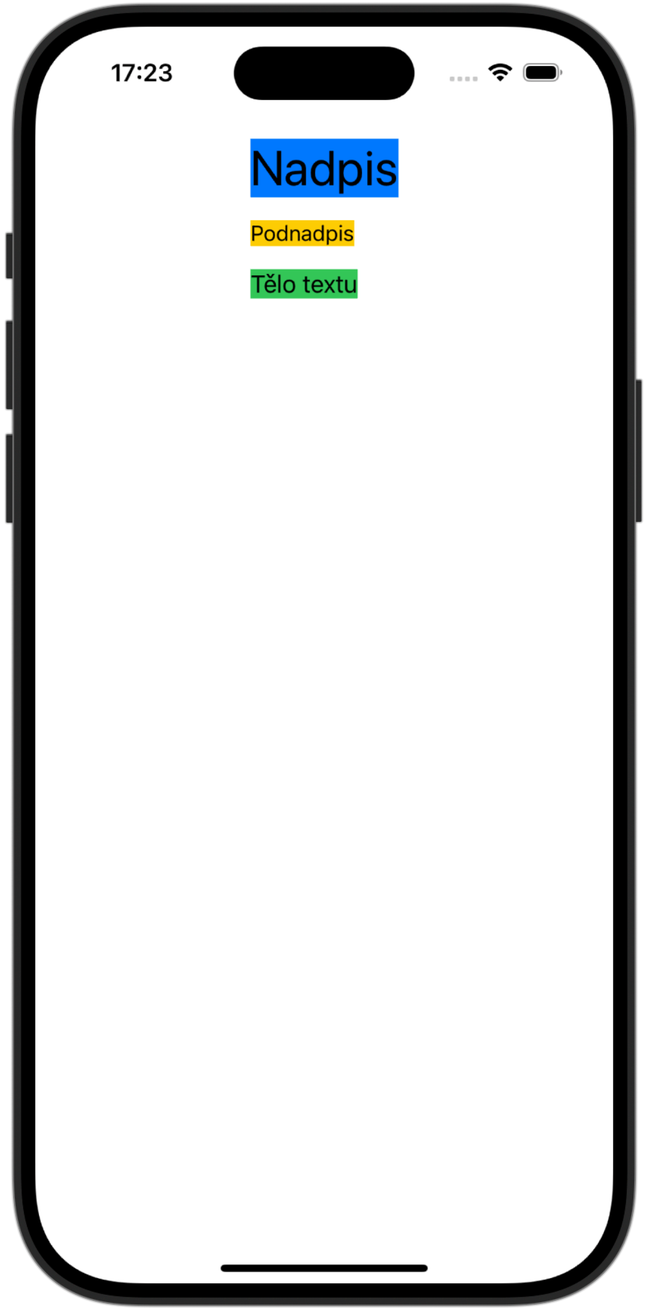


VStack

```
struct ContentView: View {

    var body: some View {
        VStack(alignment: .leading, spacing: 16) {
            Text("Nadpis")
                .font(.largeTitle)
                .background(Color.blue)
            Text("Podnadpis")
                .font(.subheadline)
                .background(Color.yellow)
            Text("Tělo textu")
                .font(.body)
                .background(Color.green)
        }
        .padding()
    }
}
```

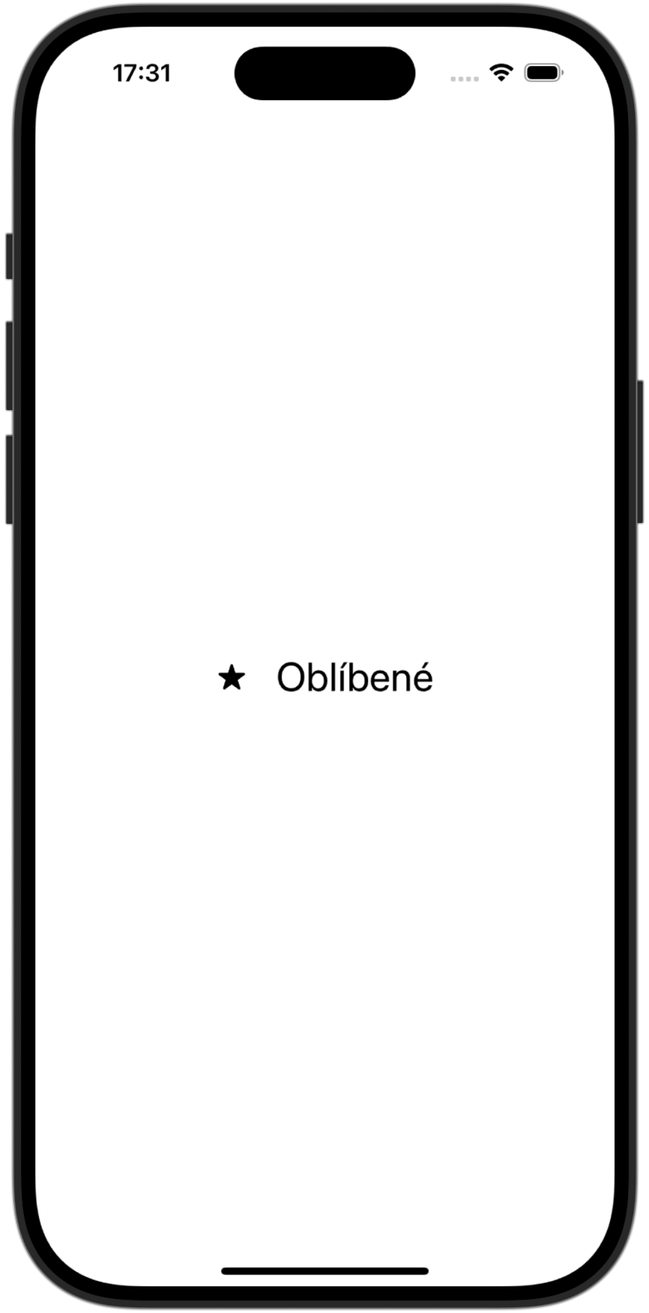
VStack

```
struct ContentView: View {

    var body: some View {
        VStack(alignment: .leading, spacing: 16) {
            Text("Nadpis")
                .font(.largeTitle)
                .background(Color.blue)
            Text("Podnadpis")
                .font(.subheadline)
                .background(Color.yellow)
            Text("Tělo textu")
                .font(.body)
                .background(Color.green)
//              Spacer vyplňuje dostupný prostor
            Spacer()
        }
        .padding()
    }
}
```



HStack

```
struct ContentView: View {

    var body: some View {
        HStack(spacing: 20) {
            Image(systemName: "star.fill")
            Text("Oblíbené")
                .font(.title)
        }
        .padding()
    }
}
```

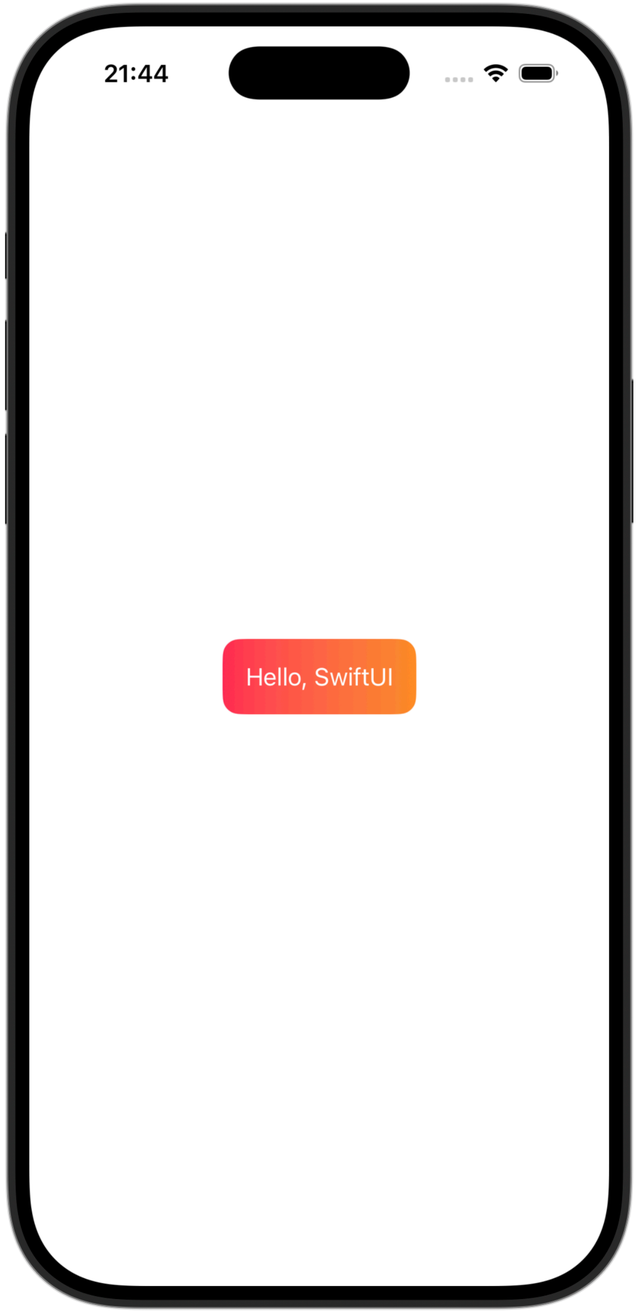


ZStack

```
struct ContentView: View {

    var body: some View {
        ZStack {
            RoundedRectangle(cornerRadius: 16)
                .fill(Color.cyan)
                .frame(width: 300, height: 200)
            Text("Překryvný text")
                .font(.largeTitle)
                .foregroundColor(.white)
                .padding(10)
                .background(Color.black.opacity(0.7))
                .cornerRadius(5)
        }
    }
}
```

LinearGradient

```
struct ContentView: View {

    var body: some View {
        Text("Hello, SwiftUI")
            .foregroundStyle(.white)
            .padding()
            .background(
                LinearGradient(
                    colors: [.pink, .orange],
                    startPoint: .leading,
                    endPoint: .trailing
                )
            )
            .cornerRadius(12)
    }
}
```



Preview

- Umožňují vývojářům zobrazit živý náhled uživatelského rozhraní přímo v Xcode během vývoje.
- Okamžitá zpětná vazba při úpravách kódu.
- Možnost testovat různé konfigurace, například světlý/tmavý režim, různé velikosti textu nebo obrazovek.

```
struct ContentView: View {

    var body: some View {
        Text("Ahoj, světe!")
    }
}

#Preview {
    ContentView()
}
```

View s parametry

```
struct UserRowView: View {
    let name: String
    let role: String
    let iconName: String
    let color: Color

    var body: some View {
        HStack(spacing: 10) {
            Image(systemName: iconName)
                .font(.system(size: 20))
                .foregroundStyle(.white)
                .frame(width: 36, height: 36)
                .background(Circle().fill(color))
            VStack(alignment: .leading, spacing: 2) {
                Text(name)
                    .font(.headline)
                Text(role)
                    .font(.caption)
                    .foregroundStyle(.secondary)
            }
            Spacer()
        }
        .padding(10)
        .background(
            RoundedRectangle(cornerRadius: 12)
                .fill(Color(.secondarySystemBackground))
        )
    }
}
```

Použití v rodičovském view

```
struct UsersListView: View {

    var body: some View {
        VStack(spacing: 8) {
            UserRowView(
                name: "Jimmy",
                role: "Teacher",
                iconName: "person.fill",
                color: .blue
            )

            UserRowView(
                name: "Alice",
                role: "Student",
                iconName: "person.fill",
                color: .pink
            )
        }
    }
}
```

Úkoly

https://vyjidacek.cz/tmai/seminar3.php