

## Osnova

- Codable z minula
- async/await
- Domácí úkol
- ? Otázky?

## Async/await

Nový přístup k psaní asynchronního kódu

Řeší spoustu věcí za programátora

Jednoduchý nativní styl psaní

```
let request = URLRequest(...)
// Old way
let task = URLSession.shared.dataTask(with: request) { data, response, error in
    if let data {
        do {
            let model = try JSONDecoder().decode(Model.self, from: data)
        } catch {
            print(error)
task.resume()
// New way
do {
    let (data, response) = try await URLSession.shared.data(for: request)
    let model = try JSONDecoder().decode(Model.self, from: data)
} catch {
    print(error)
```

### Async funkce

```
func methodName() async -> ReturnType {
}
```

Klíčové slovo async za argumenty

Potřeba zavolat z async kontextu, nelze volat přímo

# Task

Vytvoření asynchronního kontextu = naplánování async funkce

Není potřeba na něj držet referenci, systém ho odklidí když skončí

## @MainActor

Označení části kódu, která bude vždy spuštěna / použita na hlavním vlákně

Můžeme označit jak funkce, tak objekty

Opět všechno za nás na pozadí vyřešeno



#### 1. domácí úkol

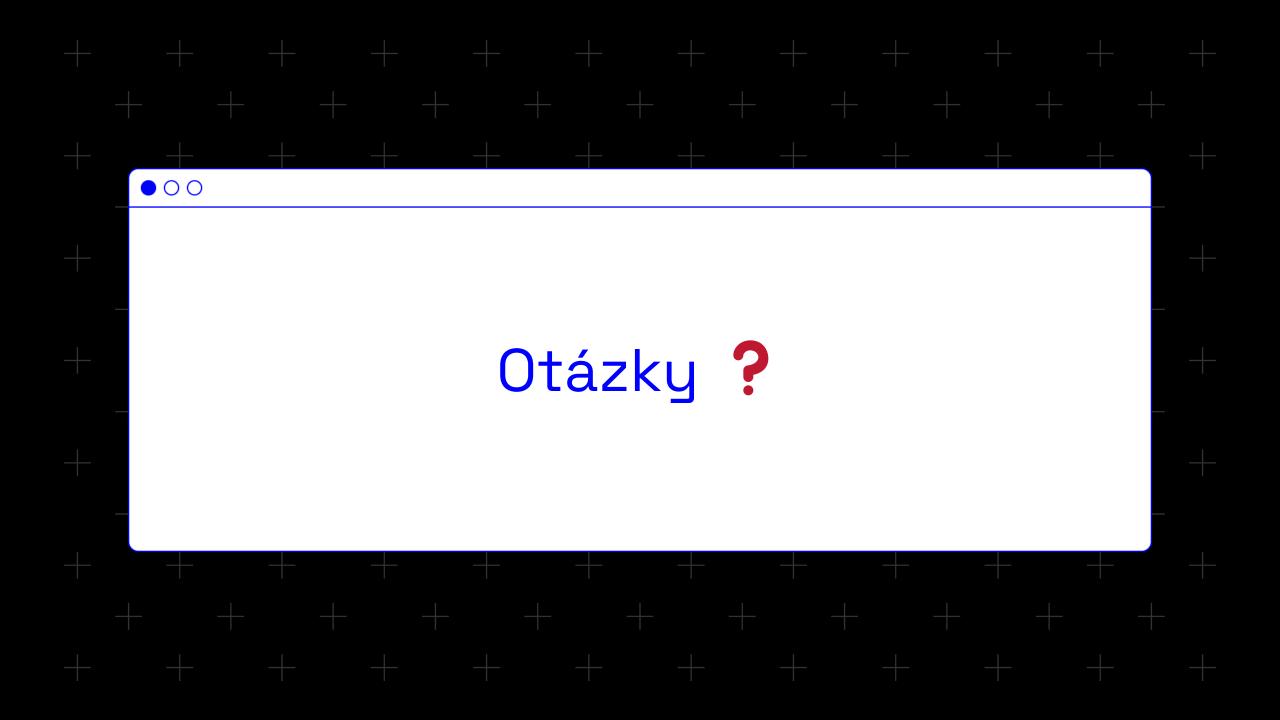
Udělat detail postu

- 🔲 může mít více fotek ( 💡 PageView )
- 🔲 nějaké hezké UI (fotky jsou vidět celé)

Načtení a zobrazení komentářů včetně nějakého UI (třeba nějak přes fotku)

Zobrazení informací o uživateli / postu s nějakým pěkným Ul

Bonus: Po tapnutí na nějaké tlačítko se skryje všechno kromě fotek



## Materiály

https://www.swiftbysundell.com/articles/connecting-async-await-with-other-swift-code/

