Projeto Pokémon

...

Foundation iOS - 25JUL02M

Integrantes

Miguel Baricelli

Gabriel Neman Silva

Metas

- Desenvolver um aplicativo iOS nativo e funcional usando SwiftUI;
- Consumir dados de uma API pública;
- Criar uma interface interativa e dinâmica, permitindo interação com o usuário;
- Listar os pokémons junto com seus respectivos nomes e imagens;
- Listar as estatísticas da pokédex.

Wireframe

- Interface intuitiva com padrão HIG da Apple;
- Design fácil e claro;
- Contraste entre as cores do App;

Home

Estatística

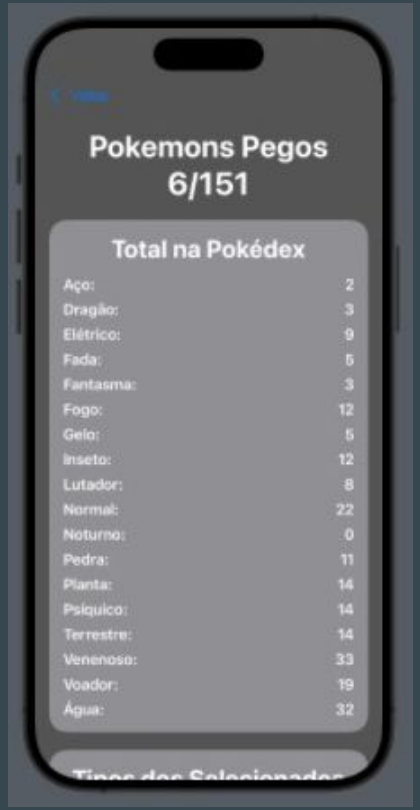
Funcionalidades Principais

- **Visualização em Grade:** Exibe todos os 151 Pokémon em uma grade
- **Seleção Múltipla:** Permite ao usuário "capturar" Pokémon, marcando-os com uma cor diferente
- **Distribuição por Tipos:** Divisão a partir dos tipos entre os Pokémon totais e os capturados.

Home

Funcionalidades Principais

- **Visualização em Grade:** Exibe todos os 151 Pokémon em uma grade
- **Seleção Múltipla:** Permite ao usuário "capturar" Pokémon, marcando-os com uma cor diferente
- **Distribuição por Tipos:** Divisão a partir dos tipos entre os Pokémon totais e os capturados.



Estatística

Trecho importante do código

async/await: Permite que as 151 chamadas de rede aconteçam em segundo plano;

JSONDecoder & Codable : Converte os dados da internet (JSON) em objetos Pokemon;

DispatchQueue.main : Garante que a interface do usuário seja atualizada.

```swift
func fetchTodosPokemon() async {
    for i in 1...151 {
        guard let url = URL(string: "https://pokeapi.co/api/v2/pokemon/\(i)") else {
            continue
        }

        do {
            let (data, _) = try await URLSession.shared.data(from: url)
            let decodedPokemon = try JSONDecoder().decode(Pokemon.self, from: data)

            DispatchQueue.main.async {
                todosPokemon.append(decodedPokemon)
            }
        } catch {
            print("Erro ao buscar Pokémon com id \(i): \(error.localizedDescription)")
        }
    }
}
```

Fetch da pokeApi

Trecho importante do código

Codable: Este protocolo converte automaticamente os dados da internet (JSON) para a estrutura do nosso aplicativo.

Modelo de Dados da API: o código imita o formato da resposta da API, garantindo que os dados sejam lidos corretamente.

selecionado : É uma variável extra, que não vem da API, criada para controlar a lógica de seleção dos botões verdes na tela.

```
struct TypeContainer: Codable {
    let type: PokemonType
}

struct PokemonType: Codable {
    let name: String
}

class Sprites: Codable {
    let front_default: String
}

struct Pokemon: Codable, Identifiable {
    let id: Int
    let name: String
    let sprites: Sprites
    let types: [TypeContainer]

    var selecionado: Bool = false

    enum CodingKeys: String, CodingKey {
        case id
        case name
        case sprites
        case types
    }
}
```

Estrutura da API

Desafios Principais

- Implementar API ao nosso aplicativo.
- Imagens nos botões
- Filtro de tipos existentes na pokédex
- Padrões HIG
- Tempo

Reflexão sobre o projeto

Em três dias, desenvolvemos um app funcional aplicando SwiftUI, consumo de APIs e gerenciamento de informação.

A experiência fortaleceu nossa lógica para resolver problemas, resultando em uma base sólida e funcional para futuras melhorias.