Projeto Pokedex em SwiftUI

```
Pokedex/
├── Models/
│   ├── Pokemon.swift
│   ├── PokemonStats.swift
│   └── ElementType.swift
├── Data/
│   └── PokemonData.swift
├── Views/
│   ├── ContentView.swift
│   ├── PokedexView.swift
│   ├── PokemonCardView.swift
│   └── StatsView.swift
├── Helpers/
│   └── TypeIconHelper.swift
├── Assets.xcassets/
│   └── (Imagens dos pokémons e ícones dos tipos)
├── PokedexApp.swift
```

Vou criar a estrutura completa do projeto Pokedex com SwiftUI. Aqui está o código organizado conforme a estrutura de arquivos solicitada:

1. Arquivos de Modelos

`Models/ElementType.swift`

```swift
import Foundation

enum ElementType: String, CaseIterable, Codable {
    case grass
    case poison
    case fire
    case flying
    case water
    case bug
```

```
    case normal
    case electric
    case ground
    case fairy
    case psychic
    case rock
    case ice
    case dragon
    case dark
    case steel
    case fighting
    case ghost
}
```

Models/Pokemon.swift

```
import Foundation

struct Pokemon: Identifiable, Codable {
    var id: Int
    var name: String
    var types: [ElementType]

    var capitalizedName: String {
        name.capitalized
    }

    var imageName: String {
        String(format: "%03d", id)
    }
}
```

Models/PokemonStats.swift

```
import Foundation

struct PokemonStats: Codable {
    var hp: Int
    var attack: Int
    var defense: Int
    var specialAttack: Int
    var specialDefense: Int
    var speed: Int
```

```
}
```

2. Arquivo de Dados

`Data/PokemonData.swift`

```
import Foundation

let pokemons: [Pokemon] = [
    Pokemon(id: 1, name: "bulbasaur", types: [.grass, .poison]),
    Pokemon(id: 2, name: "ivysaur", types: [.grass, .poison]),
    // ... (todos os outros pokémons da lista fornecida)
    Pokemon(id: 151, name: "mew", types: [.psychic])
]
```

3. Views

`Views/ContentView.swift`

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        NavigationStack {
            PokedexView()
                .navigationTitle("Pokédex")
        }
    }
}

#Preview {
    ContentView()
}
```

`Views/PokedexView.swift`

```
import SwiftUI

struct PokedexView: View {
    @State private var searchText = ""
```

```
    @State private var displayedPokemons: [Pokemon] = []
    @State private var currentPage = 1
    @State private var isLoading = false
    @State private var showLoadButton = false

    private let pageSize = 30
    private var totalPages: Int {
        Int(ceil(Double(filteredPokemons.count) / Double(pageSize)))
    }

    // Filtra os Pokémon baseado no texto de busca
    var filteredPokemons: [Pokemon] {
        if searchText.isEmpty {
            return pokemons
        } else {
            return pokemons.filter {
$0.name.localizedCaseInsensitiveContains(searchText) }
        }
    }


 // Carrega mais Pokémon quando necessário (scroll automático)
    private func loadMorePokemonsIfNeeded(currentItem item: Pokemon) {
        let thresholdIndex =
displayedPokemons.index(displayedPokemons.endIndex, offsetBy: -5)
        if displayedPokemons.firstIndex(where: { $0.id == item.id }) ==
thresholdIndex {
            loadMorePokemons()
        }
    }

    // Carrega o próximo lote de Pokémon
    private func loadMorePokemons() {
        guard !isLoading && displayedPokemons.count < filteredPokemons.count
else { return }

        isLoading = true

        // Simula um pequeno delay para carregamento (opcional)
        DispatchQueue.main.asyncAfter(deadline: .now() + 0.5) {
            let startIndex = (currentPage - 1) * pageSize
            let endIndex = min(startIndex + pageSize, filteredPokemons.count)
```

```
            let newPokemons = Array(filteredPokemons[startIndex..<endIndex])
            displayedPokemons.append(contentsOf: newPokemons)
            currentPage += 1

            // Atualiza a visibilidade do botão de carregar mais
            showLoadButton = displayedPokemons.count < filteredPokemons.count
            isLoading = false
        }
    }

    // Reinicia a paginação quando o texto de busca muda
    private func resetPagination() {
        displayedPokemons = []
        currentPage = 1
        loadMorePokemons()
    }

    // Carrega uma página específica
    private func loadPage(_ page: Int) {
        guard page >= 1 && page <= totalPages else { return }

        currentPage = page
        let startIndex = (page - 1) * pageSize
        let endIndex = min(startIndex + pageSize, filteredPokemons.count)

        displayedPokemons = Array(filteredPokemons[startIndex..<endIndex])
        showLoadButton = endIndex < filteredPokemons.count
    }

    var body: some View {
        ScrollView {
            // Contador de resultados
            HStack {
                Text("Showing \(displayedPokemons.count) of
\(filteredPokemons.count) Pokémon")
                    .font(.caption)
                    .foregroundColor(.secondary)
                Spacer()
            }
            .padding(.horizontal)

            // Grid de Pokémon
```

```
            LazyVGrid(columns: [GridItem(.adaptive(minimum: 150))], spacing:
16) {
                ForEach(displayedPokemons) { pokemon in
                    NavigationLink(destination: PokemonDetailView(pokemon:
pokemon)) {
                        PokemonCardView(pokemon: pokemon)
                            .onAppear {
                                loadMorePokemonsIfNeeded(currentItem: pokemon)
                            }
                    }
                    .buttonStyle(PlainButtonStyle())
                }
            }
            .padding()

            // Indicador de carregamento
            if isLoading {
                ProgressView()
                    .padding()
            }

            // Botão para carregar mais manualmente
            if showLoadButton && !isLoading {
                Button(action: loadMorePokemons) {
                    Label("Load More Pokémon", systemImage:
"arrow.down.circle")
                        .padding()
                        .frame(maxWidth: .infinity)
                        .background(Color.blue.opacity(0.2))
                        .foregroundColor(.blue)
                        .cornerRadius(10)
                        .padding(.horizontal)
                }
            }

            // Seletor de páginas (para listas grandes)
            if filteredPokemons.count > pageSize * 3 {
                pageSelector
                    .padding(.bottom)
            }
        }
        .searchable(text: $searchText, prompt: "Search Pokémon")
        .onChange(of: searchText) { _ in
```

```
            resetPagination()
        }
        .onAppear {
            if displayedPokemons.isEmpty {
                resetPagination()
            }
        }
        .overlay {
            if filteredPokemons.isEmpty {
                ContentUnavailableView.search(text: searchText)
            }
        }
    }

    // Seletor de páginas
    private var pageSelector: some View {
        HStack {
            // Botão anterior
            Button(action: { loadPage(currentPage - 1) }) {
                Image(systemName: "chevron.left")
                    .frame(width: 30, height: 30)
                    .disabled(currentPage == 1)
            }
            .disabled(currentPage == 1)

            // Página atual
            Text("Page \(currentPage)/\(totalPages)")
                .frame(width: 120)

            // Botão próximo
            Button(action: { loadPage(currentPage + 1) }) {
                Image(systemName: "chevron.right")
                    .frame(width: 30, height: 30)
                    .disabled(currentPage == totalPages)
            }
            .disabled(currentPage == totalPages)
        }
        .buttonStyle(.bordered)
    }
}

#Preview {
```

```
    NavigationStack {
        PokedexView()
    }
}
```

Como Usar

1. A lista começa carregando os primeiros 30 Pokémon automaticamente
2. Ao rolar até o final, mais 30 são carregados
3. Use o botão "Load More" para carregar manualmente
4. Use os controles de página para navegação rápida
5. A barra de busca filtra os resultados e reinicia a paginação

`Views/PokemonCardView.swift`

```
import SwiftUI

struct PokemonCardView: View {
    let pokemon: Pokemon

    var body: some View {
        VStack {
            AsyncImage(url: URL(string:
"https://assets.pokemon.com/assets/cms2/img/pokedex/full/\(pokemon.imageName).p
ng")) { image in
                image.resizable()
            } placeholder: {
                ProgressView()
            }
            .frame(width: 120, height: 120)

            Text(pokemon.capitalizedName)
                .font(.headline)
                .foregroundColor(.primary)

            HStack {
                ForEach(pokemon.types, id: \.self) { type in
                    TypeIconView(type: type)
                }
```

```
            }
        }
        .padding()
        .background(Color(.systemBackground))
        .cornerRadius(12)
        .shadow(radius: 5)
    }
}

#Preview {
    PokemonCardView(pokemon: pokemons[0])
        .previewLayout(.sizeThatFits)
        .padding()
}
```

Views/PokemonDetailView.swift

```
import SwiftUI

struct PokemonDetailView: View {
    let pokemon: Pokemon

    var body: some View {
        ScrollView {
            VStack(spacing: 20) {
                // Imagem do Pokémon
                AsyncImage(url: URL(string:
"https://assets.pokemon.com/assets/cms2/img/pokedex/full/\(pokemon.imageName).p
ng")) { image in
                    image.resizable()
                } placeholder: {
                    ProgressView()
                }
                .frame(width: 200, height: 200)

                // Nome e número
                VStack {
                    Text(pokemon.capitalizedName)
                        .font(.largeTitle.bold())
                    Text("#\(String(format: "%03d", pokemon.id))")
                        .font(.title3)
                        .foregroundColor(.secondary)
```

```
                }

                // Tipos
                HStack(spacing: 16) {
                    ForEach(pokemon.types, id: \.self) { type in
                        TypeIconView(type: type, large: true)
                    }
                }

                // Stats (simplificado)
                StatsView(pokemon: pokemon)

                Spacer()
            }
            .padding()
        }
        .navigationTitle(pokemon.capitalizedName)
        .navigationBarTitleDisplayMode(.inline)
    }
}

#Preview {
    NavigationStack {
        PokemonDetailView(pokemon: pokemons[0])
    }
}
```

Views/StatsView.swift

```
import SwiftUI

struct StatsView: View {
    let pokemon: Pokemon

    // Valores fictícios para demonstração
    var stats: [Stat] {
        [
            Stat(name: "HP", value: Int.random(in: 30...100),
            Stat(name: "Attack", value: Int.random(in: 30...100),
            Stat(name: "Defense", value: Int.random(in: 30...100),
            Stat(name: "Sp. Atk", value: Int.random(in: 30...100),
            Stat(name: "Sp. Def", value: Int.random(in: 30...100),
```

```
            Stat(name: "Speed", value: Int.random(in: 30...100))
        ]
    }

    var body: some View {
        VStack(alignment: .leading, spacing: 12) {
            Text("Stats")
                .font(.title2.bold())

            ForEach(stats) { stat in
                HStack {
                    Text(stat.name)
                        .frame(width: 80, alignment: .leading)

                    ProgressView(value: Float(stat.value), total: 100)
                        .tint(typeColor(for: pokemon.types.first ?? .normal))

                    Text("\(stat.value)")
                        .frame(width: 30, alignment: .trailing)
                }
            }
        }
        .padding()
        .background(Color(.secondarySystemBackground))
        .cornerRadius(12)
    }
}

struct Stat: Identifiable {
    let id = UUID()
    let name: String
    let value: Int
}

#Preview {
    StatsView(pokemon: pokemons[0])

}
```

4. Helpers

`Helpers/TypeIconHelper.swift`

```swift
import SwiftUI

struct TypeIconView: View {
    let type: ElementType
    var large: Bool = false

    var body: some View {
        Text(type.rawValue.capitalized)
            .font(large ? .subheadline.bold() : .caption.bold())
            .padding(.horizontal, large ? 12 : 8)
            .padding(.vertical, large ? 6 : 4)
            .background(typeColor(for: type))
            .foregroundColor(.white)
            .cornerRadius(20)
    }

    func typeColor(for type: ElementType) -> Color {
        switch type {
        case .grass: return Color.green
        case .poison: return Color.purple
        case .fire: return Color.orange
        case .flying: return Color.blue.opacity(0.5)
        case .water: return Color.blue
        case .bug: return Color.green.opacity(0.7)
        case .normal: return Color.gray
        case .electric: return Color.yellow
        case .ground: return Color.brown
        case .fairy: return Color.pink
        case .psychic: return Color.pink.opacity(0.7)
        case .rock: return Color.gray.opacity(0.7)
        case .ice: return Color.cyan
        case .dragon: return Color.indigo
        case .dark: return Color.black
        case .steel: return Color.gray.opacity(0.5)
        case .fighting: return Color.red
        case .ghost: return Color.purple.opacity(0.5)
        }
    }
}

#Preview {
    HStack {
```

```
        TypeIconView(type: .fire)
        TypeIconView(type: .water, large: true)
    }
}
```

5. Arquivo Principal

`PokedexApp.swift`

```
import SwiftUI

@main
struct PokedexApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```