

# LISTAS



# Structs e Arrays

Um **STRUCT** é um tipo de valor que encapsula os dados relacionados juntos, facilitando o gerenciamento e a manipulação

```
struct Estudante {  
    var nome: String  
    var idade: Int  
}
```

- Construindo OBJETOS (incluindo **Array** de objetos):

```
var aluno : Estudante = Estudante(nome: "João", idade: 25)  
  
var alunos = [ Estudante(nome: "JP", idade: 22),  
              Estudante(nome: "PH", idade: 23) ]
```

# List

Estrutura que permite iterar sobre uma coleção, como um array, e criar uma visualização para **cada elemento** dessa coleção já com um **estilo predefinido** pela linguagem.

```
NavigationStack {  
    VStack{  
        List(estudantes, id: \.self) { e in  
            Text(e.nome)  
  
        }.navigationTitle("Alunos")  
        .toolbar {  
            Button("Adicionar Aluno") {  
                estudantes.append(Estudante(nome: "Novo Aluno", idade: 27))  
            }  
        }  
    }  
}
```



Não permite muitas alterações de estilo e tem scrollview acoplado.

# ForEach

Uma estrutura que permite iterar sobre uma coleção, como um array, e criar uma visualização para cada elemento dessa coleção com um estilo personalizado.

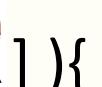
```
NavigationStack {  
    VStack {  
        ForEach(estudantes, id: \.self) { e in  
            Text(e.nome)  
                .padding(10)  
                .frame(minWidth: 100, maxWidth: 40)  
                .background(.gray)  
                .foregroundStyle(.white)  
        }  
        .navigationTitle("Alunos")  
        .toolbar {  
            Button("Adicionar Aluno") {  
                estudantes.append(Estudante(nome: "João", idade: 10))  
            }  
        }  
    }  
}
```



Permite alterações de estilo e não tem scrollview acoplado.

# Listas

```
[  
  Estudante(nome: "Hermione", idade: 11),  
  Estudante(nome: "Harry", idade: 10),  
  Estudante(nome: "Rony", idade: 12)  
]
```

```
ForEach([ , ,  ]){ aluno in  
  Text(aluno.nome)  
}
```



1ª Iteração - personagem =



2ª Iteração - personagem =



3ª Iteração - personagem =



# Elementos da Lista - Identificador Único

É importante que sejam identificados de forma única para que, quando sejam “clicados”, consigam informar o código de quem foi que recebeu o clique.

- A sintaxe `id: \.CAMPO` é usada para identificar exclusivamente cada elemento na lista a partir de um campo da própria Struct.
- A sintaxe `id: \.self` é usada para “burlar” a identificação única de cada elemento da lista no caso de não acessarmos outra tela a partir da lista.

Atenção!!

Hashable  
Identifiable  
Identifiable

# Hashable X Identifiable

```
struct Estudante : Hashable {  
    var nome : String  
    var idade : Int  
}
```

Protocolo Hashable: por meio de um cálculo matemático, gera cada filho da Struct de forma única, criando um ID.

```
List(array, id: \.self) { item in  
    Text(item.nome)  
}
```

```
ForEach(array, id: \.self) { item in  
    Text(item.nome)  
}
```

Uma vez com Hashable, tanto **List** quanto **ForEach** reconhecem o ID por meio do `\.self`.

```
struct Estudante : Identifiable {  
    var id : Int  
    var nome : String  
    var idade : Int  
}
```

Protocolo Identifiable: obriga que tenhamos um campo chamado `id` na Struct.

```
List(array) { item in  
    Text(item.nome)  
}
```

```
ForEach(array) { item in  
    Text(item.nome)  
}
```

Uma vez com Identifiable, tanto **List** quanto **ForEach** reconhecem de forma automática o ID.

# HORA DO DESAFIO!

Spoiler:

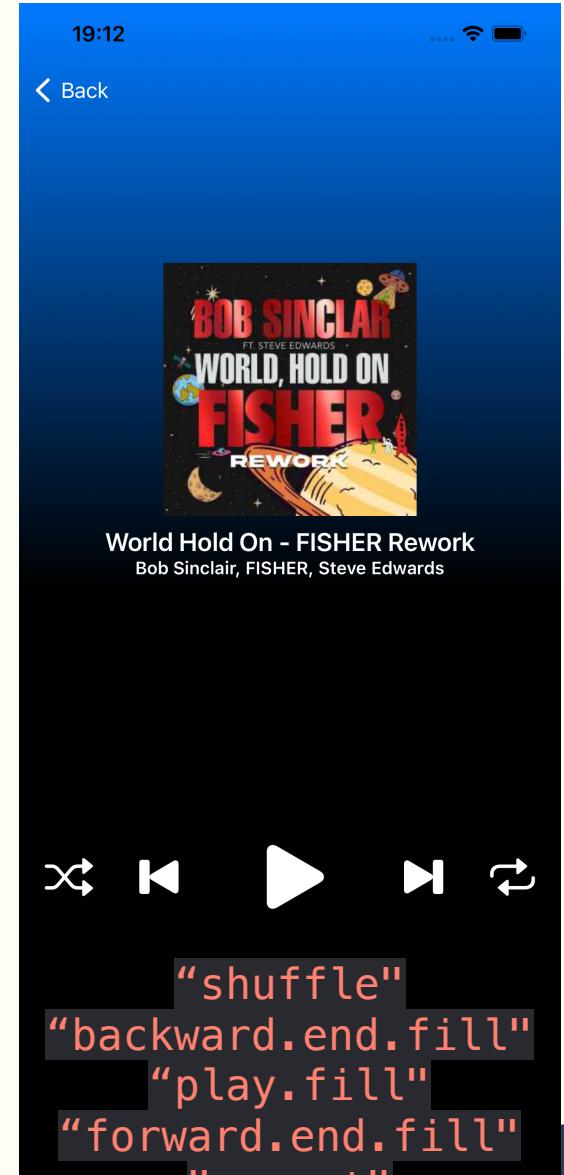
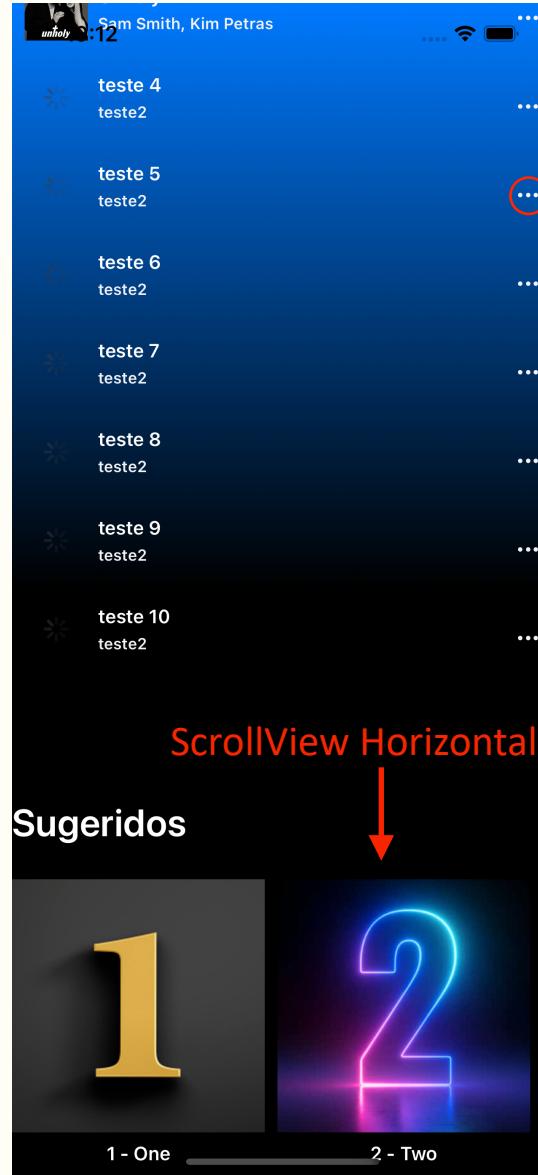
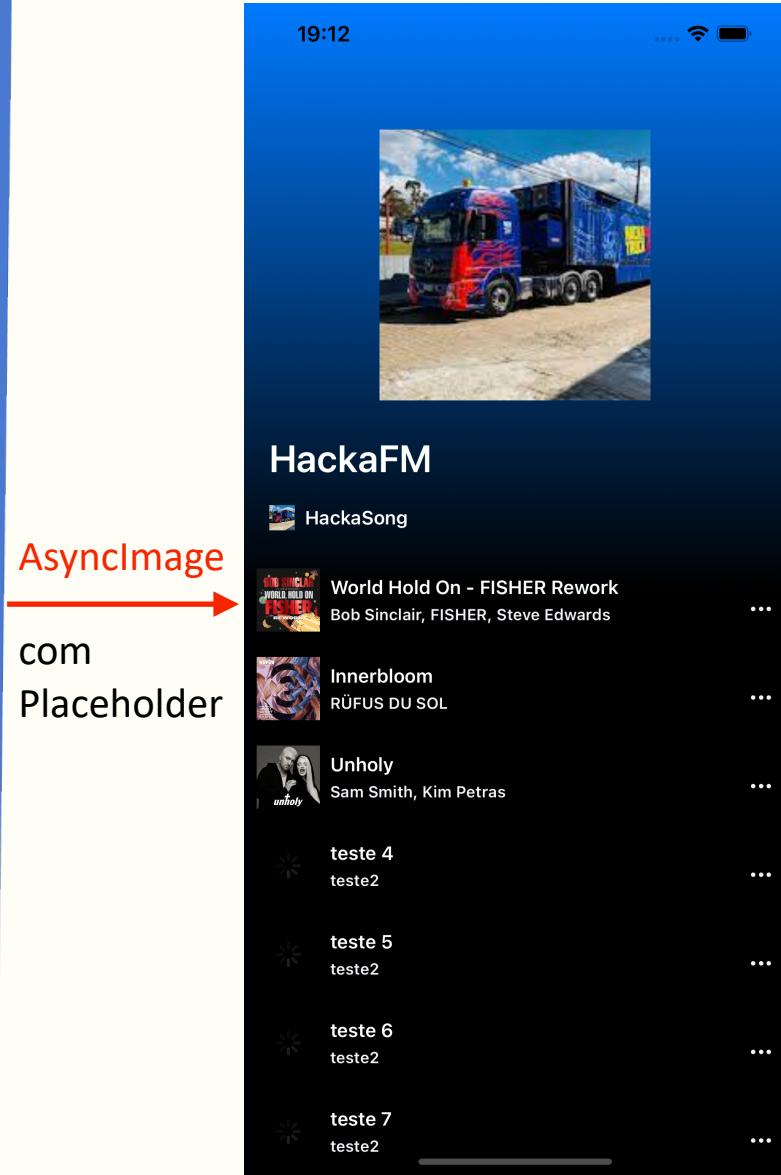


# Structs e Arrays

```
struct Song : Identifiable {  
    var id: Int  
    var name : String  
    var artist : String  
    var capa : String  
}
```

```
var arrayMusicas = [  
    Song(id: 4, name:"Numb Encore", artist:"Linkin Park", capa:"lpCollisionCourse"),  
    Song(id: 5, name:"Lost", artist:"Linkin Park", capa:"lpLOST")  
]
```

# Desafio:





HACKA  
TRUCK *Maker Space*

/hackatruck

@hackatruck\_makerspace

HACKATRUCK MAKERSPACE

[www.hackatruck.com.br](http://www.hackatruck.com.br)