Efetuando consultas

Para efetuar consultas, necessitamos de um novo método na nossa API. Esse método deverá receber as informações referentes à consulta e retornar zero ou mais registros obtidos do banco de dados. Crie um método na classe CommonUtils com o nome executaConsulta e insira o código a seguir:

```
  func executaConsulta(_ query: Select,
                       _ aoFinal: @escaping ([[Any?]]?)-
>()) {
   let thread = DispatchGroup()
   thread.enter()
   var registros = [[Any?]]()
   if let connection = getConnection() {
      connection.execute(query: query) { result in
        guard let dados = result.asResultSet else {
              print("Não houve resultado para esta
consulta")
              return thread.leave()
        }
```

```
            dados.forEach { linha, error in
                if let _linha = linha {
                    var colunas: [Any?] = [Any?]()
                    _linha.forEach({ atributo in
                        colunas.append(atributo)
                    })
                    registros.append(colunas)
                } else {
                    thread.leave()
                }
            }
        }
    } else {
        print("Sem conexão")
        thread.leave()
    }
    thread.wait()
    aoFinal(registros)
}
```

Este método recebe dois parâmetros, o primeiro define a consulta a ser executada e o segundo é o bloco de instruções que tratará a lista de registros retornados da consulta. O bloco de instruções será codificado nas funções que criaremos a seguir. Para a execução da consulta, uma thread é iniciada, que a partir da conexão efetuará a consulta submetendo uma instância da estrutura Select. Os dados retornados dessa consulta são tratados a seguir com a sua transformação em um array, para que, caso tudo esteja correto ao final, repasse ao bloco de instruções fornecido no segundo parâmetro a chamada: aoFinal(registros).

Para elaborarmos as consultas, crie um arquivo do tipo Swift com o nome Consultas e, em seguida, acrescente o código a seguir:

```
import Foundation
import SwiftKuery

public extension String {
 func fill(to: Int = 20) -> String {
    var saida = self
    if self.count < to {
       for _ in 0..<(to - self.count) {
            saida += " "
       }
    }
    return saida
 }
}

func consulta(_ select: Select) {
 let utils = CommonUtils.sharedInstance
 utils.executaConsulta(select) { registros in
    guard let registros = registros else {
       return print("Sem registros")
    }
    registros.forEach { linha in
       linha.forEach { item in
            print("\(item ?? "")".fill(), terminator:
" ")
       }
       print()
    }
 }
}
```

Iniciaremos criando uma **extension** em String para propiciar a melhor formatação do texto na saída do programa. Desse modo, poderemos visualizar melhor os dados retornados da consulta SQL. Já a função consulta vai permitir simplificar imensamente a apresentação dos resultados de qualquer consulta que façamos, pois ela obtém a instância da API para executar o Select passado como argumento além de verificar se há conteúdo ao receber os registros. Em caso negativo, emite uma mensagem avisando, e se houver conteúdo, este é apresentado, tendo cada informação separada por até vinte espaços.

Vejamos a primeira consulta:

```
func consulta1() {
 let albuns = Albuns()
 let musicas = Musicas()
 let musicasDoAlbum = MusicasDosAlbuns()

}
```

Nossa primeira consulta retornará o nome do álbum e o nome da música. Para construir essa resposta, será necessária a utilização das classes Albuns, Musicas e MusicasDosAlbuns em função da associação existente entre as tabelas no banco de dados.

A consulta a ser utilizada é semelhante à query SQL a seguir:

```
  SELECT Album.nome, Musica.nome FROM Album
         LEFT JOIN Musicas_do_Album
                ON Album.idAlbum =
Musicas_do_Album.idAlbum
         LEFT JOIN Musica
                ON Musicas_do_Album.idBanda =
Musica.idBanda
               AND Musicas_do_Album.idMusica =
Musica.idMusica;
  Vamos implementar esta consulta SQL utilizando a
linguagem Swift:
  func consulta1() {
   let albuns = Albuns()
   let musicas = Musicas()
   let musicasDoAlbum = MusicasDosAlbuns()

   consulta(Select(albuns.nome, musicas.nome, from:
albuns)
      .leftJoin(musicasDoAlbum)
        .on(albuns.idAlbum == musicasDoAlbum.idAlbum)
      .leftJoin(musicas)
        .on(musicasDoAlbum.idBanda == musicas.idBanda
        && musicasDoAlbum.idMusica ==
musicas.idMusica))

  }
```

Como é possível notar no código em destaque, a sintaxe oferecida pela plataforma de desenvolvimento do SwiftKuery nos permite construir a consulta de modo similar ao utilizado na query SQL. A chamada à função consulta, recebendo como parâmetro a instância de Select, permite executar e apresentar o resultado para a consulta.

Agora, alteraremos mais uma vez o programa main para acrescentar a consulta dos dados:

```
import Foundation

//criaTabelas()
//insereDados()
consulta1()
```

Comentamos a função anterior e acrescentamos a chamada à função consulta1, o que produzirá a seguinte saída:

```
Empty Words          Burning Babylon
Empty Words          Empty Words
Simplicity           Devotion
Rise from the Ashe   Avalanche
Rise from the Ashe   Between the Lines
Rise from the Ashe   Dear Youth
Rise from the Ashe   Greater Distance
Stay Like This
Walk to you          Everlasting
Above it All         Above it All
```

Vamos elaborar outra consulta utilizando diferentes cláusulas SQL, veja o que implementaremos a seguir:

```
SELECT Banda.nome, COUNT(Musica.idBanda)
 FROM Musica
 JOIN Banda ON Musica.idBanda = Banda.idBanda
 GROUP BY Musica.idBanda
 HAVING COUNT(Musica.idBanda) >= 2
 ORDER BY COUNT(Musica.idBanda) DESC;
```

Vamos iniciar a construção de nossa nova consulta em Swift com o código a seguir:

```
func consulta2() {
 let musicas = Musicas()
 let bandas = Bandas()

 consulta(Select(bandas.nome, count(musicas.idBanda),
    from: musicas)
    .join(bandas).on(musicas.idBanda ==
bandas.idBanda)
    .group(by: musicas.idBanda)
    .having(count(musicas.idBanda) >= 2)
    .order(by: .DESC(count(musicas.idBanda))))
}
```

Com a implementação da função consulta, podemos notar o quanto as consultas são simplificadas. Para isso, basta obter as instâncias das classes que desejamos atuar, construir o Select e passar para ela consultar e imprimir o conteúdo retornado.

O código em destaque é mais uma vez semelhante à construção SQL que apresentamos. Vamos alterar o programa main para acrescentar a consulta2:

```
import Foundation

//criaTabelas()
//insereDados()
//consulta1()
consulta2()
```

O resultado dessa execução será:

```
The Ghost Inside     4
Alix Perez           2
```