Criação das estruturas no banco de dados

Para podermos gravar os dados no banco de dados, necessitamos ter as estruturas de dados criadas com nossa API. Criaremos também algumas classes para representar as estruturas de dados e uma função que passará as informações para a API.

Confira o método na classe CommonUtils que fará o serviço de criação das tabelas:

```
  func criaTabela(_ tabela: Table) {
   let thread = DispatchGroup()
   thread.enter()
   guard let con = getConnection() else {
      return print("Sem conexão")
   }
   tabela.create(connection: con) { result in
      if !result.success {
         print("Falha ao criar a tabela
\(tabela.nameInQuery)")
      }
      thread.leave()
      }
   thread.wait()
  }
```

O método apresentado recebe o parâmetro do tipo Table, uma classe de estrutura semelhante à tabela a ser criada no banco de dados. Todo o processamento para a criação da tabela é assíncrono, ou seja, não ocorre em sincronia com a execução do método criaTabela que o chamou. Essa assincronia é necessária, pois sempre que há acesso à rede, o tempo de retorno da chamada não pode ser determinado.

Para tratar requisições assíncronas, a linguagem Swift dispõe de um mecanismo que chamamos **Thread**. A thread é criada com a chamada DispatchGroup() e é possível sinalizar:

- quando entramos num bloco de instruções que será executado assincronamente com thread.enter();
- quando esperamos no trecho de código síncrono pela finalização da thread com thread.wait(); e
- como deixamos a thread finalizando a execução dela com thread.leave().

Importante

Todos os métodos da API que acessarão a rede para alcançar o banco de dados MySQL farão uso de threads.

Como a definição das estruturas das tabelas deverão ser do tipo Table, construiremos as classes que representam nosso banco de dados. Portanto, crie um arquivo Swift com o nome de Database.swift e implemente as estruturas como a seguir:

```
  import Foundation
  import SwiftKuery

  class Bandas: Table {
   let tableName = "Banda"
   let idBanda = Column("idBanda", Int32.self,
              autoIncrement: true, primaryKey: true,
notNull: true)
   let nome = Column("nome", String.self, notNull:
true)
  }
```

Neste exemplo, criaremos cinco tabelas com relacionamentos, iniciando com a tabela Banda, o nome da classe está no plural para diferenciar de outra classe que criaremos posteriormente. Esta classe estende Table e utiliza a classe Column para definir as características para cada futuro atributo da tabela que será criado no MySQL. O atributo tableName representa o nome da tabela, os demais atributos representam as colunas na tabela. O Atributo idBanda representa a chave primária da tabela.

Toda a definição utilizada no DDL para a criação das estruturas com SQL é representada por parâmetros que informamos à instância da classe Column. Os tipos dos atributos utilizam tipos de dados Swift que serão convertidos para os tipos MySQL.

As outras tabelas seguem a mesma sequência de definições, analisemos as outras:

```
  class Estilos: Table {
   let tableName = "Estilo"
   let idEstilo = Column("idEstilo", Int32.self,
                autoIncrement: true, primaryKey: true, notNull:
true)
   let nome = Column("nome", String.self, notNull: true)
  }

  class Albuns: Table {
   let tableName = "Album"
   let idAlbum = Column("idAlbum", Int32.self,
                autoIncrement: true, primaryKey: true, notNull:
true)
   let nome = Column("nome", String.self, notNull: true)
   let ano = Column("ano", Int32.self, notNull: true)
   let idBanda = Column("idBanda", Int32.self, notNull: true)
   let idEstilo = Column("idEstilo", Int32.self, notNull: true)
  }

  class Musicas: Table {
   let tableName = "Musica"
   let idBanda = Column("idBanda", Int32.self, notNull: true)
   let idMusica = Column("idMusica", Int32.self,
                autoIncrement: true, primaryKey: true, notNull:
true)
   let nome = Column("nome", String.self, notNull: true)
   let idEstilo = Column("idEstilo", Int32.self, notNull: true)
  }

  class MusicasDosAlbuns: Table {
   let tableName = "Musicas_do_Album"
   let idBanda = Column("idBanda", Int32.self, notNull: true)
   let idMusica = Column("idMusica", Int32.self, notNull: true)
   let idAlbum = Column("idAlbum", Int32.self, notNull: true)
  }
```

A classe Albuns tem dois atributos, idBanda e idEstilo, que representam chaves estrangeiras. A classe Musicas tem um atributo (idEstilo) como chave estrangeira e a classe MusicasDosAlbuns representa uma tabela de junção para uma relação N para N, tendo os atributos idBanda como chave estrangeira para a tabela Banda e os atributos idMusica e idAlbum como chaves estrangeiras para a tabela Música. As relações entre tabelas serão implementadas na função que criaremos a seguir.

Crie outro arquivo Swift com o nome CriaTabelas e codifique da seguinte maneira:

```
import Foundation
import SwiftKuery

func criaTabelas() {
 let utils = CommonUtils.sharedInstance

}
```

A função criaTabelas conterá toda a lógica necessária à criação das tabelas e à definição das regras de relação entre elas. Foi declarada uma constante utils que contém a referência à instância da classe que implementa a nossa API para o acesso ao MySQL. Vamos criar algumas tabelas a seguir:

```
func criaTabelas() {
 let utils = CommonUtils.sharedInstance

 let bandas = Bandas()
 utils.criaTabela(bandas)
 print("Tabela Banda criada")

 let estilos = Estilos()
 utils.criaTabela(estilos)
 print("Tabela Estilo criada")
}
```

No código destacado é criada a instância de Bandas, que é passada para o método criaTabela; esta chamada à API criará a tabela no banco de dados seguindo os critérios definidos como propriedades da classe Bandas. Em caso de falha, a API emitirá uma mensagem; em caso de sucesso, apresentaremos a mensagem Tabela criada. O mesmo procedimento é executado para a classe Estilos.

Vejamos a criação das regras de relacionamento entre tabelas:

```
func criaTabelas() {
 let utils = CommonUtils.sharedInstance

 let bandas = Bandas()
 utils.criaTabela(bandas)
 print("Tabela Banda criada")

 let estilos = Estilos()
 utils.criaTabela(estilos)
 print("Tabela Estilo criada")
```

```
let albuns = Albuns()
   _ = albuns.foreignKey(albuns.idBanda, references:
bandas.idBanda)
   _ = albuns.foreignKey(albuns.idEstilo,
                         references:
estilos.idEstilo)

   utils.criaTabela(albuns)
   print("Tabela Álbum criada")

   let musicas = Musicas()
   _ = musicas.foreignKey(musicas.idBanda,references:
bandas.idBanda)
   _ = musicas.foreignKey(musicas.idEstilo,
                         references:
estilos.idEstilo)

   utils.criaTabela(musicas)
   print("Tabela Música criada")
  }
```

Após a criação da instância da classe Albuns, são criadas as associações entre as tabelas utilizando as chaves associativas das respectivas tabelas.

Para finalizar a criação da nossa estrutura de banco de dados, falta a tabela que representa a associação N para N. Vejamos:

```
  func criaTabelas() {
    let utils = CommonUtils.sharedInstance

    let bandas = Bandas()
    utils.criaTabela(bandas)
    print("Tabela Banda criada")

    let estilos = Estilos()
    utils.criaTabela(estilos)
    print("Tabela Estilo criada")

    let albuns = Albuns()
    _ = albuns.foreignKey(albuns.idBanda, references:
bandas.idBanda)
    _ = albuns.foreignKey(albuns.idEstilo,
                          references: estilos.idEstilo)
    utils.criaTabela(albuns)
    print("Tabela Álbum criada")

    let musicas = Musicas()
    _ = musicas.foreignKey(musicas.idBanda,references:
bandas.idBanda)
    _ = musicas.foreignKey(musicas.idEstilo,
                          references: estilos.idEstilo)
    utils.criaTabela(musicas)
    print("Tabela Música criada")
```

```
    let musicasDoAlbum = MusicasDosAlbuns()
    _ =
musicasDoAlbum.primaryKey(musicasDoAlbum.idBanda,
                    musicasDoAlbum.idMusica,
musicasDoAlbum.idAlbum)

    _ = musicasDoAlbum.foreignKey(
                [musicasDoAlbum.idBanda,
musicasDoAlbum.idMusica],
                references: [musicas.idBanda,
musicas.idMusica])
    _ =
musicasDoAlbum.foreignKey(musicasDoAlbum.idAlbum,
                    references: albuns.idAlbum)

    utils.criaTabela(musicasDoAlbum)
    print("Tabela Músicas do Álbum criada")
  }
```

A chave primária, que para a tabela Musicas_do_Album é uma chave formada pelos idBanda , idMusica e idAlbum e como depende da descrição dos campos das outas tabelas, não pode ser definida em sua própria classe. Neste exemplo também foram declaradas as chaves estrangeiras para as tabelas Música e Álbum. Vale notar que a relação para a tabela Música contém duas chaves associativas e por essa razão no método foreignKey foram informados os arrays contendoas respectivas chaves de cada tabela.

Para efetuar a criação das tabelas, adicionaremos a chamada a esta função que acabamos de codificar no programa main:

```
import Foundation

criaTabelas()
```

Basta simplesmente efetuar a chamada à função criaTabelas que o resultado a seguir será apresentado:

```
Tabela Banda criada
Tabela Estilo criada
Tabela Álbum criada
Tabela Música criada
Tabela Musicas do Álbum criada
```