```swift
//
//  ViewController.swift
//  lemercier_18_tp10_test
//
//  Created by marc on 28/11/2018.
//  Copyright © 2018 if26. All rights reserved.
//

import UIKit
import SQLite

class ViewController: UIViewController {

    @IBOutlet weak var boutonTest: UIButton!
    @IBOutlet weak var texteCount: UITextField!

    var database: Connection!

    let users_table = Table("users")
    let users_id = Expression<Int>("id")
    let users_name = Expression<String>("name")
    let users_email = Expression<String>("email")

    var pk = 1000;           // valeur de départ pour la primary key
    var tableExist = false   // false la table n'est encore pas créée

    // ====================================================

    override func viewDidLoad() {
        super.viewDidLoad()
        print ("--> viewDidLoad debut")

        // Il est possible de créer des fichiers dans le répertoire
         "Documents" de votre application.
        // Ici, création d'un fichier users.sqlite3

        do {
            let documentDirectory = try
             FileManager.default.url(for: .documentDirectory,
             in: .userDomainMask, appropriateFor: nil, create: true)
            let fileUrl =
             documentDirectory.appendingPathComponent("users").appendingPath
             Extension("sqlite3")
            let base = try Connection(fileUrl.path)
            self.database = base;
        }
        catch {
            print (error)
        }
        print ("--> viewDidLoad fin")
    }

    // ====================================================

    func createTableUsers() {
```

```
    print ("--> createTableUsers debut")
    if !self.tableExist {
        self.tableExist = true

        // Instruction pour faire un drop de la table USERS
        let dropTable = self.users_table.drop()

        // Instruction pour faire un create de la table USERS
        let createTable = self.users_table.create { table in
            table.column(self.users_id, primaryKey: true)
            table.column(self.users_name)
            table.column(self.users_email)
        }

        do {
            // Exécution du drop et du create
            try self.database.run(dropTable)
            try self.database.run(createTable)
            print ("Table users est créée")
        }
        catch {
            print (error)
        }
    }
    print ("--> createTableUsers fin")
}

// ================================================
// Création d'un générateur de clé primaire

func getPK() -> Int {
    self.pk += 1
    return self.pk
}

// ================================================

func insertTableUsers() {
    print ("--> insertTableUsers debut")

    // Insertion de 2 tuples exemples (sera réalisé à chaque click sur
     le bouton)
    let insert1 = self.users_table.insert(self.users_id <- getPK(),
     self.users_name <- "marc", self.users_email <- "marc@if26.fr")
    let insert2 = self.users_table.insert(self.users_id <- getPK(),
     self.users_name <- "sophie", self.users_email <- "sophie@if26.fr")

    do {
        try self.database.run(insert1)
        print ("Insert1 ok")
        try self.database.run(insert2)
        print ("Insert2 ok")
    }
    catch {
        print (error)
```

```
    }
    print ("--> insertTableUsers fin")
}

// ====================================================
// Solution 1 : Combien de tuples dans la table USERS ?

func CountTableUsers1() -> Int {
    print ("--> CountTableUsers1 debut")
    var resultat = 0
    do {
        resultat = try self.database.scalar(users_table.count)
        print ("count1 = ", resultat)
    }
    catch {
        print (error)
        resultat = -1
    }
    print ("--> CountTableUsers1 fin")
    return resultat
}

// ====================================================
// Solution 2 : Combien de tuples dans la table USERS ?

func CountTableUsers2() -> Int64 {
    print ("--> CountTableUsers2 debut")
    do {
        let stmt = try self.database.prepare("SELECT COUNT(*) FROM
         USERS")
        for row in stmt{
            if let resultat = row[0] {
                print ("count2 = ", resultat)
                return (resultat as! Int64)
            }
        }
    } catch{
        print("CountTableUsers2 est en erreur")
    }
    print ("--> CountTableUsers2 fin")
    return -1;
}

// ====================================================
// Solution 1 pour récupérer les data à partir d'une requête SQL

func selectUSersSophie() {
    print ("--> selectUSersSophie debut")
    do {
        let users = try
         self.database.prepare(self.users_table.filter(users_name ==
         "sophie") )
        for user in users {
            print ("id =", user[self.users_id], ", name =",
             user[self.users_name], ", email= ", user[self.users_email])
```

```
        }
    } catch{
        print("--> selectUSersSophie est en erreur")
    }
    print ("--> selectUSersSophie fin")
}
// ====================================================
// Solution 2 pour récupérer les data à partir d'une requête SQL

func selectUsersMarc() {
    print ("--> selectUsersMarc debut")
    do {
        let stmt = try self.database.prepare("SELECT * FROM USERS where
         name = 'marc'")
        for row in stmt {
            print ("id =", row[0]!, " name =", row[1]!)
        }
    } catch{
        print("--> selectUsersMarc est en erreur")
    }
    print ("--> selectUsersMarc fin")
}

// ====================================================
// Select *

func selectUSers() {
    print ("--> selectUSers debut")
    do {
        let users = try self.database.prepare(self.users_table)
        for user in users {
            print ("id =", user[self.users_id], ", name =",
             user[self.users_name], ", email= ", user[self.users_email])
        }
    } catch{
        print("--> selectUSers est en erreur")
    }
    print ("--> selectUSers fin")
}

// ====================================================
// Update ?

func updateUserName() {
    print ("--> updateUserName debut")

    let user = self.users_table.filter(self.users_id == 1001)
    let updateUser = user.update (self.users_name <- "Ines")
    do {
        try self.database.run(updateUser)
        print ("User 1 modifié")
    }
    catch{
        print("--> updateUserName est en erreur")
    }
```

```
        print ("--> updateUserName fin")
    }

    // ================================================
    // delete ?

    func deleteUser(numero : Int ) {
        print ("--> deleteUser debut")

        let user = self.users_table.filter(self.users_id == numero)
        let deleteUser = user.delete()
        do {
            try self.database.run(deleteUser)
            print ("Suppression user ", numero, " effectuée même s'il
             n'existe pas !")
        }
        catch{
            print("--> deleteUser est en erreur")
        }
        print ("--> deleteUser fin")
    }

    // ================================================
    // Moteur de l'application ! gestion du click sur le bouton

    @IBAction func boutonAction() {
        print ("-")
        print ("=====================================")
        print ("--> Début des traces ")
        createTableUsers()
        insertTableUsers()

        let resultat1 = CountTableUsers1()
        texteCount.text = String (resultat1)

        let resultat2 = CountTableUsers2()
        texteCount.text = String (resultat2)

        selectUSersSophie()
        selectUsersMarc()
        selectUSers()

        updateUserName()
        deleteUser(numero: 1004)
        print ("--> boutonAction fin")
    }
}
```