15 Persistance de données

15.1 SQLite

Plusieurs possibilités, ici on va créer une BD SQLite car il a l'avantage d'être extrêmement léger avec également peu d'adhérence au système d'exploitation utilisé puisqu'il se base sur un système de fichiers. Celle-ci est stocké dans le bac à sable de l'application. Seule l'application et le système d'exploitation a accès à cette zone. La Base de Données reste cependant locale et ne permet pas de créer une application partagée.

Pour pouvoir utiliser le code sur plusieurs plateformes, le mieux est d'installer une PCL (Portable Class Libraries). Pour ajouter la PCL **SQLite-Net**, il faut utiliser le gestionnaire de paquets NuGet.

Projet → Gérer les packages NuGet… et installer le package sqlite-net-pcl.

15.2 Classe de données

Les tables d'une base de données sont générées en se basant sur des classes de données.

Créer un nouveau répertoire appelé **Model**, dans lequel on va créer les différentes classes de données. Le code C# va être converti en requêtes SQL par le package SQLite.

```
// pour pouvoir utiliser les mots clés PrimaryKey et AutoIncrement
using SQLite.Net.Attributes;

// attributs Android, ici changement du nom de la table
[Table("People")]

// nom de la classe donc de la table
public class User {
   //identifiant, avec comme attributs clé primaire et auto-incremente
   [PrimaryKey, AutoIncrement]
   public int ID { get; set; }

   [MaxLength(100), Unique]
   public string useName { get; set; }

}
```

D'autres attributs sont disponibles tels que [Ignore] qui permet d'indiquer qu'une propriété déclarée dans la classe C# ne doit pas être interprétée par SQLite et donc pas ajoutée à la table associée ou [Indexed], [MaxLength] et [NotNull].

15.3 Service d'accès aux données

Il faut créer une interface entre la couche métier et la base de données. Cette interface se matérialise par un **Repository** déposé dans un nouveau répertoire **Service**. C'est ici que se trouveront toutes les méthodes de manipulation des données (ajout, suppression, modification etc…)

```
// repository pour la table User
public class UserRepository
{
   // message de statut
   public string _statusMessage { get; set; }

   // implementation SQLite specifique a la plateforme.
   SQLiteAsyncConnection _connection;

   public UserRepository(string dbPath)
   {
      // recuperation ou creation de la connexion a la DB
      _connection = new SQLiteAsyncConnection(dbPath);

      // creation de la table User
      _connection.CreateTableAsync<User>();

   }

   // ajout d'un utilisateur
   public async Task AddNewUserAsync(string name)
   {
      var result = 0;

      try
      {
         // ajout d'un utilisateur
         result = await _connection.InsertAsync(new User { useName = name });
         _statusMessage = $"{result} utilisateur ajouté : {name}";
      }
      catch (Exception ex)
      {
         // message en cas d'erreur
         _statusMessage = $"AdNewUser erreur {name}.\n Erreur : {ex.Message} ";
      }
   }

   // recuperation de tous les utilisateurs
   public async Task<List<User>> GetUsersAsync ()
   {
      try
      {
         // retourne la liste des tous les utilisateurs
         return await _connection.Table<User>().ToListAsync();
      }
      catch(Exception ex)
      {
         // en cas d'erreur, message et retour d'une liste vide
         _statusMessage = $"GetUsers Impossible.\n Erreur : {ex.Message} ";
         return new List<User>();
      }
   }
}
```

Il est ensuite possible d'utiliser ce repository dans son code.

```
// Instanciation du repository
_userRepository = new UserRepository(_dbPath);
```

Avec pour la lieu où est stocké la base de donnée :

```
// db
private string _dbPath = Path.Combine(FileSystem.AppDataDirectory,
"testSQLite.db3");
```

16 Utilisation d'un capteur (Sensor)

16.1 Détecter un capteur

Avant d'utiliser un capteur, il faut s'assurer de sa présence sur l'appareil qui exécute l'application.

Si le capteur n'est pas disponible et qu'il est indispensable au bon fonctionnement de l'application, cela pose un problème. C'est pourquoi il est préférable d'indiquer au Google Play Store que l'application ne doit être présentée qu'aux terminaux disposant de ce capteur.

L'utilisation des capteurs met en jeu des classes du package **android.hardware**.

La classe **SensorManager** permet d'utiliser les capteurs. Il est possible par exemple de vérifier la présence d'un capteur selon son type ou de lister tous les capteurs disponibles sur l'appareil.

```
// recuperation d'un objet SensorManager
SensorManager sensorMgr = (SensorManager)GetSystemService(SensorService);

// liste de tous les capteurs disponibles
IList<Sensor> sensors = sensorMgr.GetSensorList(SensorType.All);
foreach(Sensor crtSensor in sensors)
{
   textTmp.Text += "\n" + crtSensor.Name;
}
```

16.2 Utiliser un capteur

Pour pouvoir utiliser les événements liés aux capteurs, il est nécessaire de faire hériter la classe de **ISensorEventListener.**

```
public class MainActivity : AppCompatActivity, ISensorEventListener
```

Les capteurs sont de type **Sensor**. Pour choisir le capteur à utiliser, il faut l'enregistrer avec la méthode **RegisterListener**.

```
private SensorManager _sensorMgr;
private Sensor _accelerometer;
private float _xValue;


// enregistrement de l'accelerometre dans OnCreate ou OnResume
_accelerometer = _sensorMgr.GetDefaultSensor(SensorType.Accelerometer);
_sensorMgr.RegisterListener(this, _accelerometer, SensorDelay.Normal);
```

Il faut ensuite utiliser la méthode **OnSensorChanged** pour effectuer les tâches désirées.

```
public void OnSensorChanged(SensorEvent e)
{
   // il s'agit de l'accelrometre
   if (e.Sensor.Equals(_accelerometer))
   {
      // 3 valeurs qui correspondent ici au 3 accelerations en x, y et z
      IList<float> values = e.Values;

      // detecte un mouvement horizontal
      if (Math.Abs(values[0] - _xValue) > 5)
      {
         _textTmp.Text = "Effacer les éléments";
      }
```

```
            else
            {
               _xValue = values[0];
            }
         }
      }
```

Ne pas oublier de désenregistrer le capteur lorsque l’activité est en pause.

```
// liberer l'accelerometre dans OnPause
_sensorMgr.UnregisterListener(this);
```