Doc TestXamConnection

Le projet TestXamConnections propose une couche d'abstraction pour se connecter à des appareils bluetooth depuis un appareil Android.
La librairie a été développée pour fonctionner avec les appareils suivants :

- Balance Teo de la marque Balea
- Appareil Android C-One² proposant le scan de puce RFID grâce à agrident Wedge
- Lecteur barcode HoneyWell
- Lecteur RFID externe Agrident
- *Barcode interne ?*

Pour chacun de ces appareils, il existe dans la librairie des classes dédiées qui proposent les fonctions pour les utiliser. Ces classes sont organisées par type d'appareils puis par fabricant. L'organisation des classes de la librairie est visible sur l'arbre ci-dessous.

```
.
├── Connection
│   ├── ConnectionConstants.cs
│   ├── IConnectionService.cs
│   └── IConnectionServiceProvider.cs
├── Device
│    ├── Balance
│    │   ├── BalanceProvider.cs
│    │   ├── Balea
│    │   │   ├── TeoBalance.cs
│    │   │   └── TeoConstants.cs
│    │   └── IBalance.cs
│    ├── Barcode
│    │    ├── BarcodeReaderProvider.cs
│    │    ├── Honeywell
│    │    │   ├── HoneywellBTReader.cs
│    │    │   ├── HoneyWellConstants.cs
│    │    │   └── HoneywellInternReader.cs
│    │    └── IBarcodeReader.cs
│    ├── DeviceInfo.cs
│    ├── IDevice.cs
│    └── RFID
│         ├── Agrident
│         │   ├── AgridentConstants.cs
│         │   └── AgridentInternReader.cs
│         ├── IRFIDReader.cs
│         └── RFIDReaderProvider.cs
├── Models
```

```
│     ├── ReponsePesee.cs
│     └── ReponseRFID.cs
└── Services
      └── IBluetoothService.cs
```

Les modèles pour chaque appareil sont classés selon leur type : lecteur RFID, lecteur de code-barre, et balance. Chaque type possède son interface de laquelle chaque modèle hérite. On les obtient grâce aux *Providers*. Un provider est une classe statique qui possède une fonction elle même statique. Cette fonction prend en argument le **modèle de l'appareil** (définit dans une énumération pour chaque type d'appareil : par exemple, RFID reader possède le lecteur agrident BT et le lecteur agrident interne au c-one dans l'énumération RFIDDevice - voir extrait de code ci-dessous), **le type de connexion** (voir code ci-dessous) ainsi que **les infos de l'appareil** (Adresse MAC et Nom).

```
public enum RFIDDevice
    {
        AgridentIntern,
        AgridentBT
    }

public enum ConnectionType
    {
        Bluetooth,
        Intern,
        Wifi
    }
```

Par exemple, si l'on souhaite récupérer le modèle qui gère la communication avec un lecteur Agrident en Bluetooth :

```
DeviceInfo myDeviceInfo = new DeviceInfo()
    {
        MACAddress = "00:25:96:FF:FE:12:34:56"
        Name = "Agrident Device"
    };


IRFIDReader MonLecteurAgrident =
RFIDReaderProvider.GetRFIDReader(RFIDDevice.AgridentIntern,
Connection.ConnectionType.Intern, myDeviceInfo);
```

Pour la balance Teo :
namespace: TestXamConnection.Device.Balance.Teo
Il s'agit de la classe TeoBalance qui hérite de l'inferce IBalance. Cette classe propose les fonctions suivantes (héritées de l'interface):

- **Task<bool> ConnectToTeoAsync()** : permet d'établir une connexion bluetooth entre l'appareil et la balance Teo. Important : le bluetooth doit être activé, la balance allumée et les appareils doivent déjà être appairés.
  Renvoie true si la connexion a réussi (i.e. un socket RFCOMM pour la balance Teo a été ouvert), false sinon.

- **Task<bool> SendCommandToTeoAsync(TeoCommandType)** : permet d'envoyer une commande à la balance pour le programme INRAE. L'ensemble des commandes est disponible dans l'enum TeoCommandType. Retourne true si la commande a été envoyée, false sinon.

- **Task<bool> ReadWeightAsync() :** permet d'envoyer une commande de pesée simple à la balance. Retourne true si la commande a été envoyée, false sinon.

La classe présente un évènement auquels il est possible de s'abonner :

- **TeoPeseeReceivedEvent<ReponsePesee>** : déclenché à la réception d'une pesée, disponible au format ReponsePesee.

L'objet ReponsePesee :

- *bool Err* : indique si la pesée a rencontré une erreur (vrai : erreur)
- *int NumPlateau* : numéro du plateau
- *int TypePlateau* : type du plateau
- *string PoidsMesure* : chaîne de caractère contenant la valeur de la pesée
- *bool IsNegative* : indique si le poids est faible ou non (true : poids négatif)

Pour le lecteur de RFID Agrident présent dans le C-One :
namespace: TestXamConnections.Device.RFID.Agrident

La classe AgridentInternReader hérite de l'interface IRFIDReader et propose les éléments suivants :
Évènement :

- **RFIDDataReceivedEvent<ReponseRFID>** : Événement déclenché à la réception de données par le lecteur RFID.

Fonctions :

- **Task<bool> EnableServiceAsync()** : Initialise la lecture de tag RFID.
  Retourne true la connexion a réussi, false sinon.
- **Task<bool> StartScanAsync()** : Lance un scan de tag RFID.
  Retourne true si le scan s'est lancé correctement, false sinon.

La classe pour le lecteur Agrident Bluetooth hérite de la même interface et propose donc les mêmes éléments.

Pour la lecture de code barre, le modèle bluetooth fonctionne sur la même base que le lecteur agrident bluetooth.
Le lecteur interne n'est pas encore traité, la classe est disponible mais le comportement obtenu n'est pas celui souhaité (travail futur…)

La structure de la librairie est visible sur le diagramme de classe suivant :

Voir SicpaDeviceConnectionsUML.drawio.svg (dans le rep. Git du projet, à ouvrir dans un navigateur Internet)

Il est aussi possible d'utiliser les composants de la librairie pour se connecter à un nouvel appareil.
Le fonctionnement repose sur le **ConnectionServiceProvider** qui fournit un **ConnectionService** en fonction du type de connexion demandé.
Ces classes proposent toutes les même fonctionnalité :

- La connexion avec la fonction Connect(params). Le passage des arguments se fait à l'aide d'un Dictionnary<string,string>:
  - En interne, il s'agit de l'enregistrement des intents qui vont être reçus en les passants en arguments. La fonction attend un champ "receivable_intents"

(ConnectionConstants.RECEIVABLE_INTENTS_KEY) avec une liste d'intent séparées par une virgule (csv)
    - En bluetooth il s'agit de l'ouverture d'un socket RFCOMM en fonction de l'adresse MAC passée en argument. La fonction attend un champ "mac_address" avec l'adresse MAC passé en string ("XX:XX:XX…").

Dans le cas où les paramètres sont erronés, la fonction Connect renvoie false.

- L'envoi d'une commande sous la forme d'un tableau d'octet à l'aide de la fonction SendCommand. En cas d'erreur lors de l'envoi de la commande, la fonction renvoie false.

- Un évènement DataReceivedEvent<byte[]>, déclenché à la réception de données. Présente les données sous forme d'un tableau d'octet.