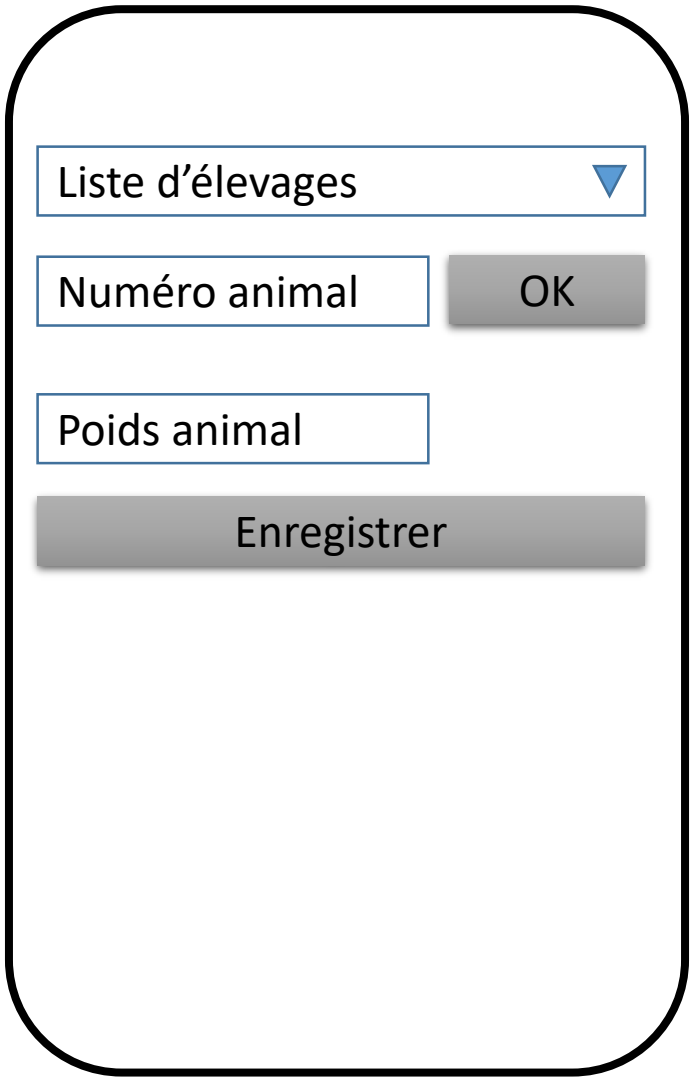
Bien démarrer avec Xamarin

En partant d'un exemple de programme à réaliser



Objectif du programme :

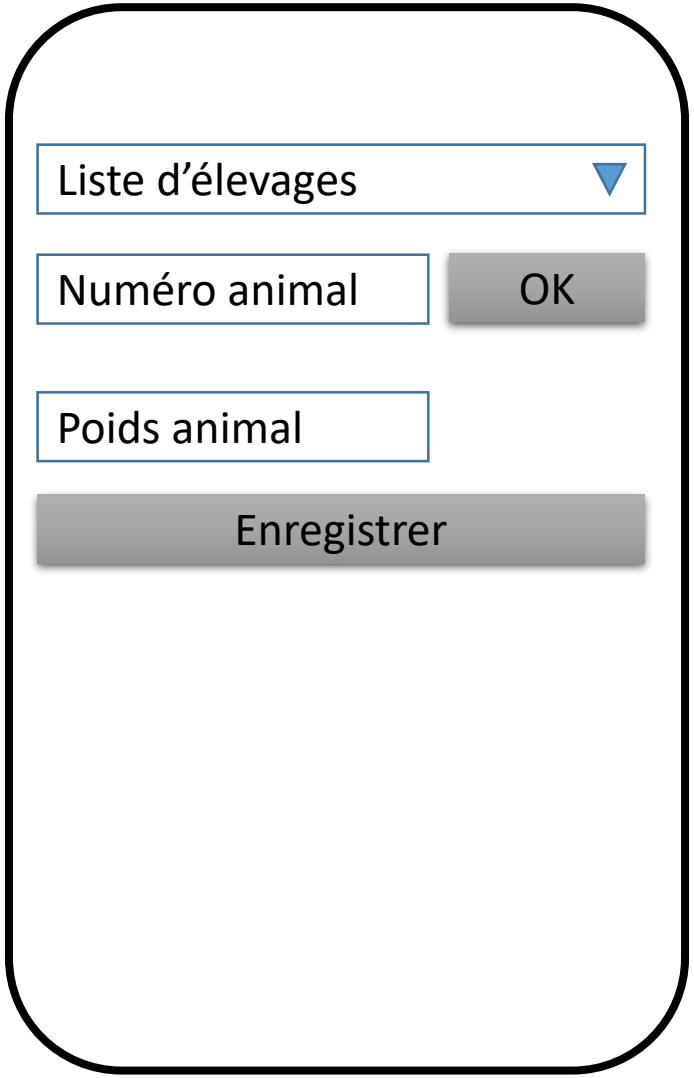
Enregistrer la pesée d'un animal

Etapes :

1/ Sélectionner un élevage
2/ Saisir un numéro animal
3/ Valider la saisie (= vérifier qu'il existe dans l'élevage sélectionné)
4/ Saisir le poids
5/ Enregistrer (simuler l'enregistrement en affichant les infos saisies)

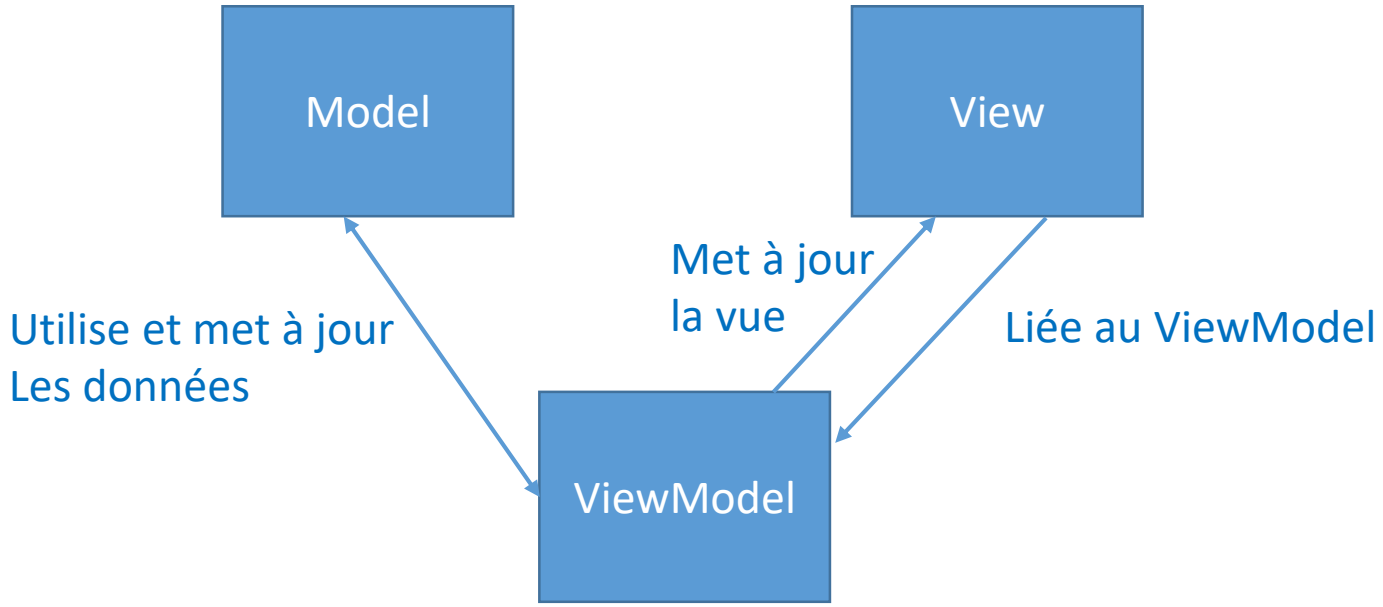
Bien démarrer avec Xamarin

En partant d'un exemple de programme à réaliser



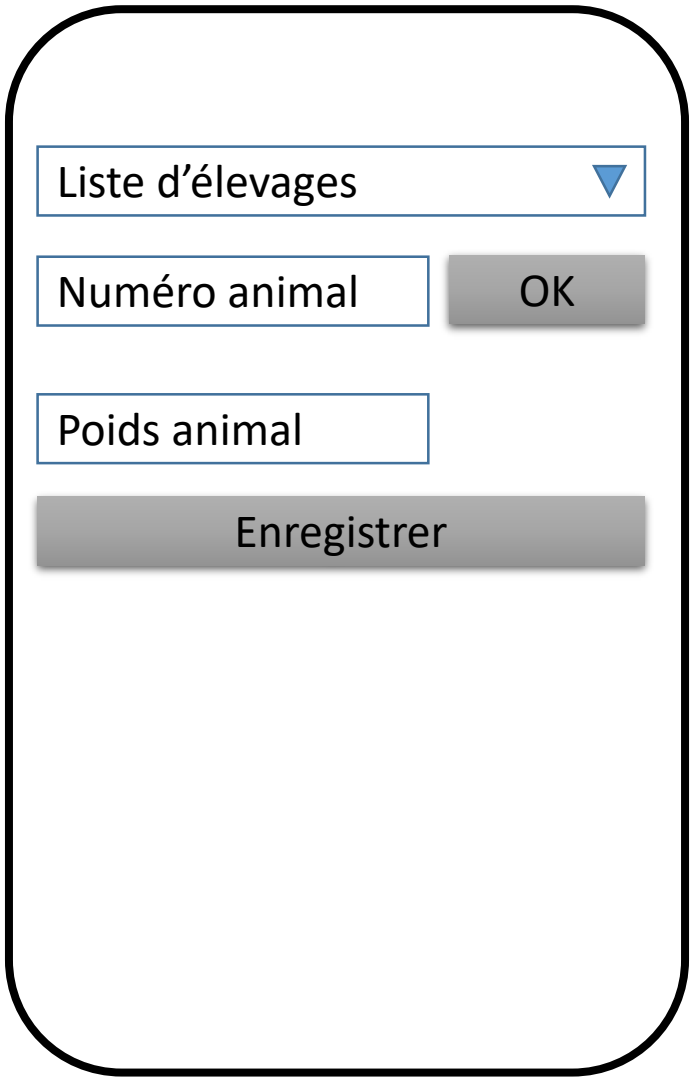
Objectif de la programmation :

Utiliser les pattern MVVM et DAO

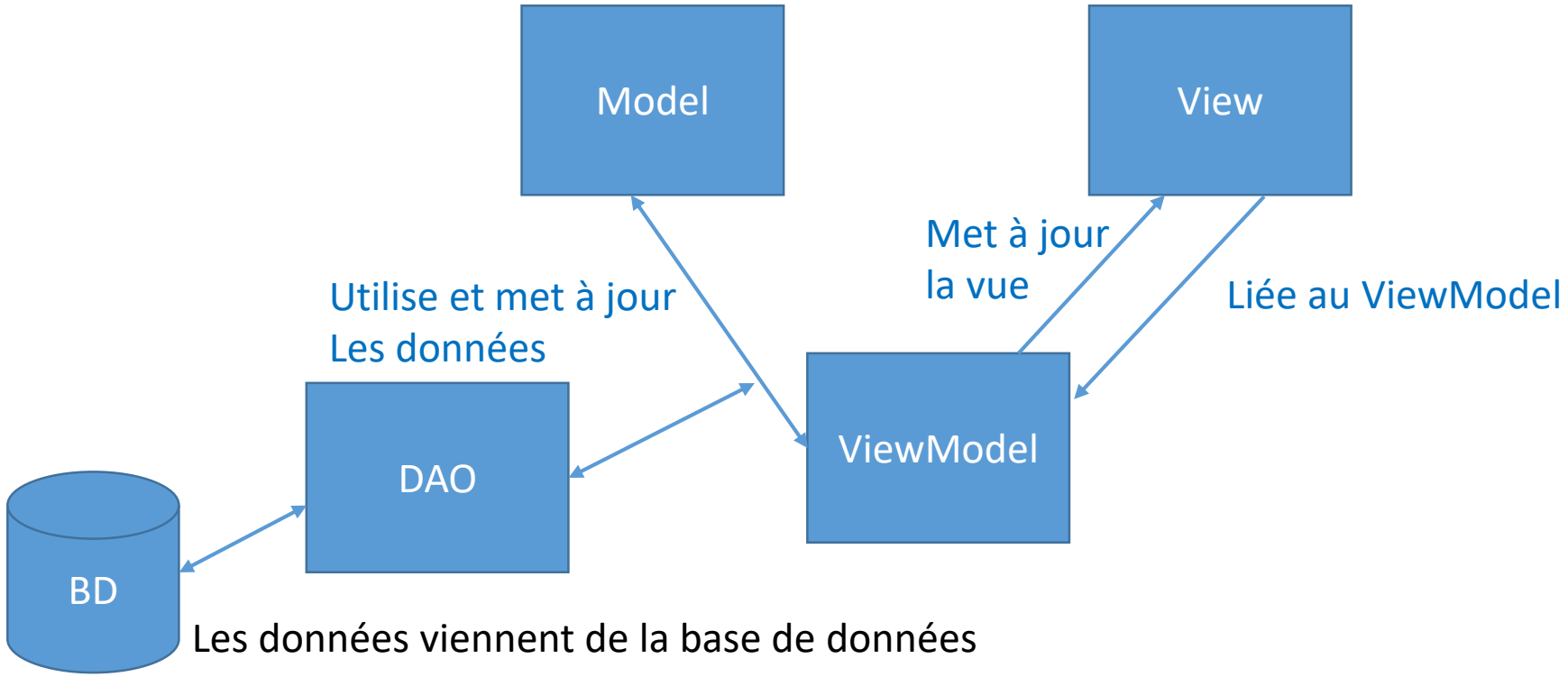


Bien démarrer avec Xamarin

En partant d'un exemple de programme à réaliser



Objectif de la programmation :
Utiliser les pattern MVVM et DAO



Bien démarrer avec Xamarin

Modèle de données

Model

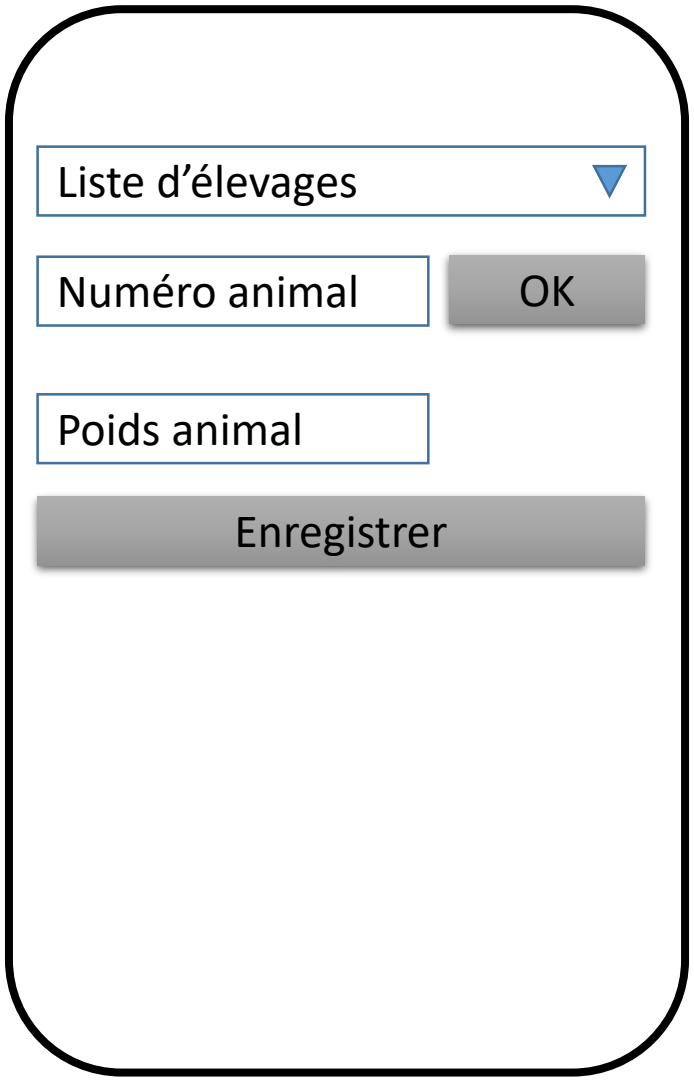
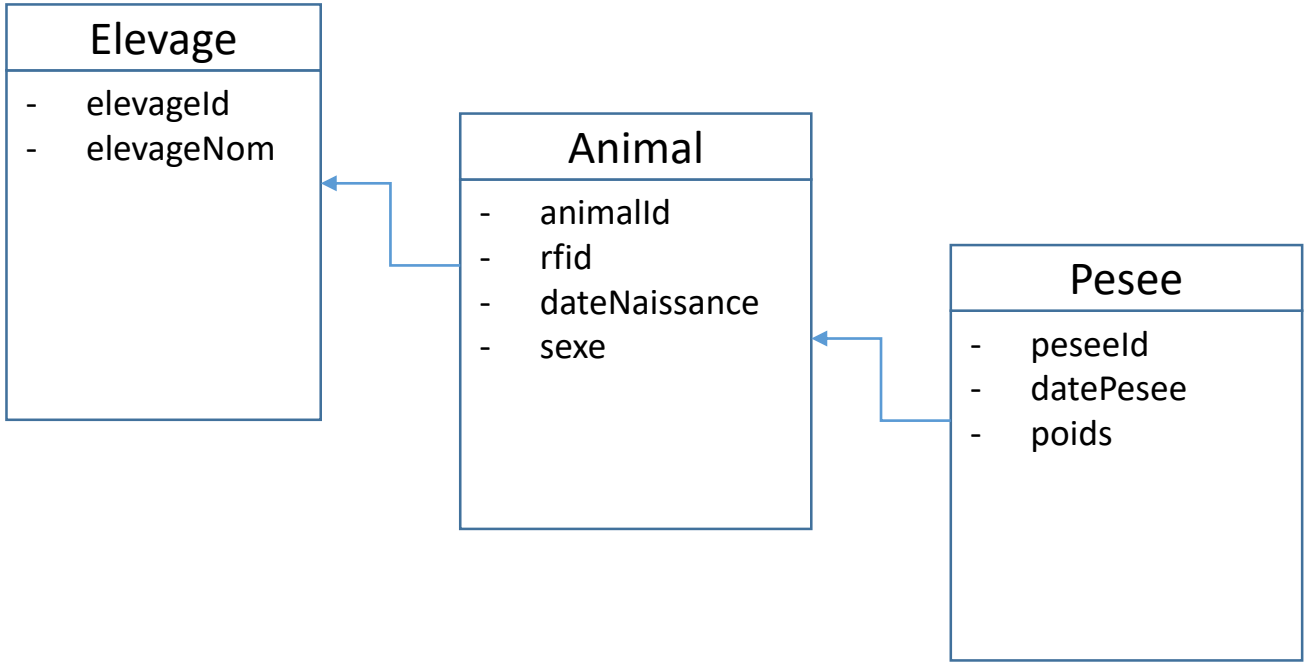


Diagramme de classes :

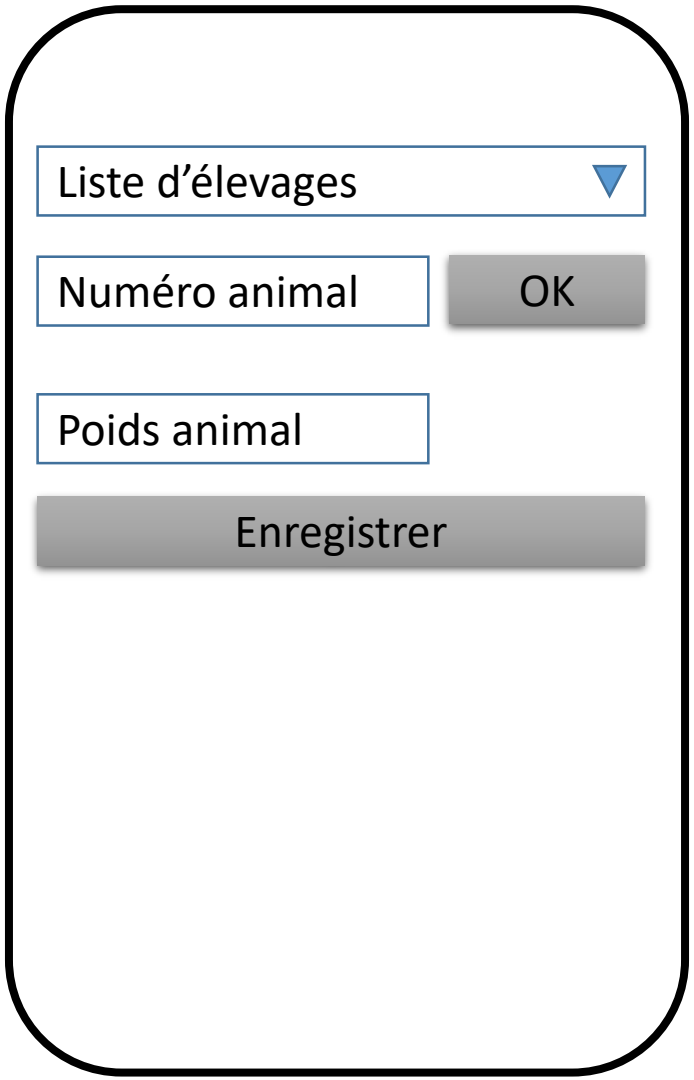


Donc 3 classes à écrire dans le dossier Model

Bien démarrer avec Xamarin

Accès aux données

DAO



Fonctions dont on a besoin :

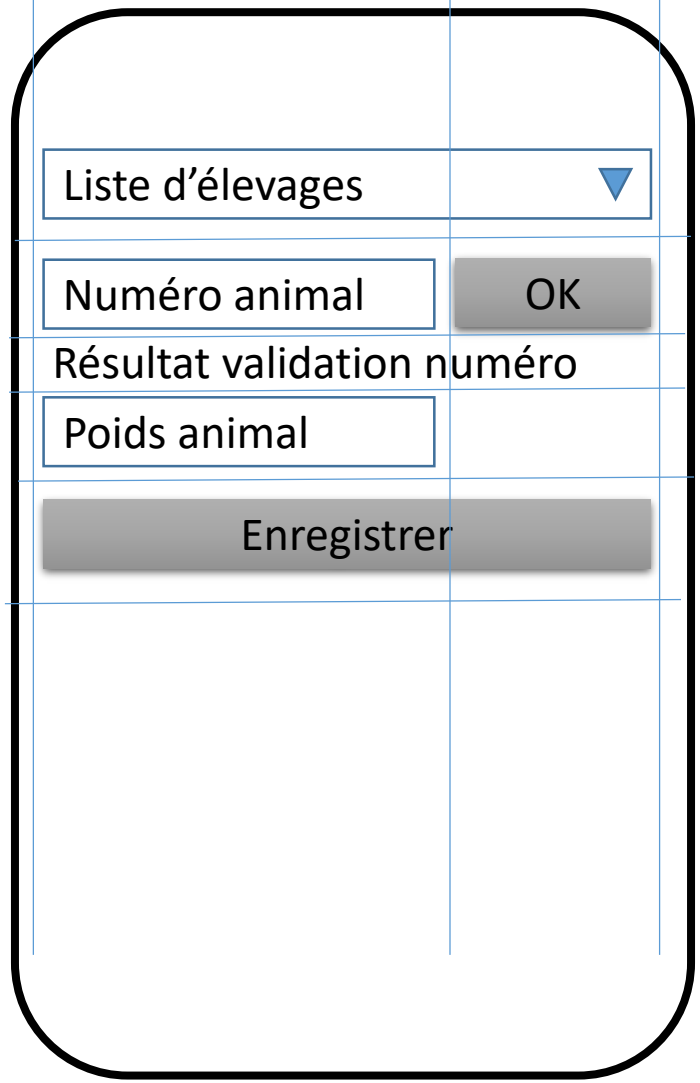
- Lister tous les élevages de la base de données
- Lister tous les animaux d'un élevage
- Enregistrer la pesée
- => 3 méthodes à écrire :
  - `List<Elevage> getElevages()`
  - `List<Animal> getAnimauxByElevage(Elevage el)`
  - `void save(Pesee pe)`

- Comme ces 3 méthodes concernent 3 objets différents, il est recommandé de **créer 3 classes dans le dossier DAO** :
  - `ElevageDAO`
  - `AnimalDAO`
  - `PeseeDAO`

Bien démarrer avec Xamarin

Construction de l'IHM

View



Composition de l'écran :

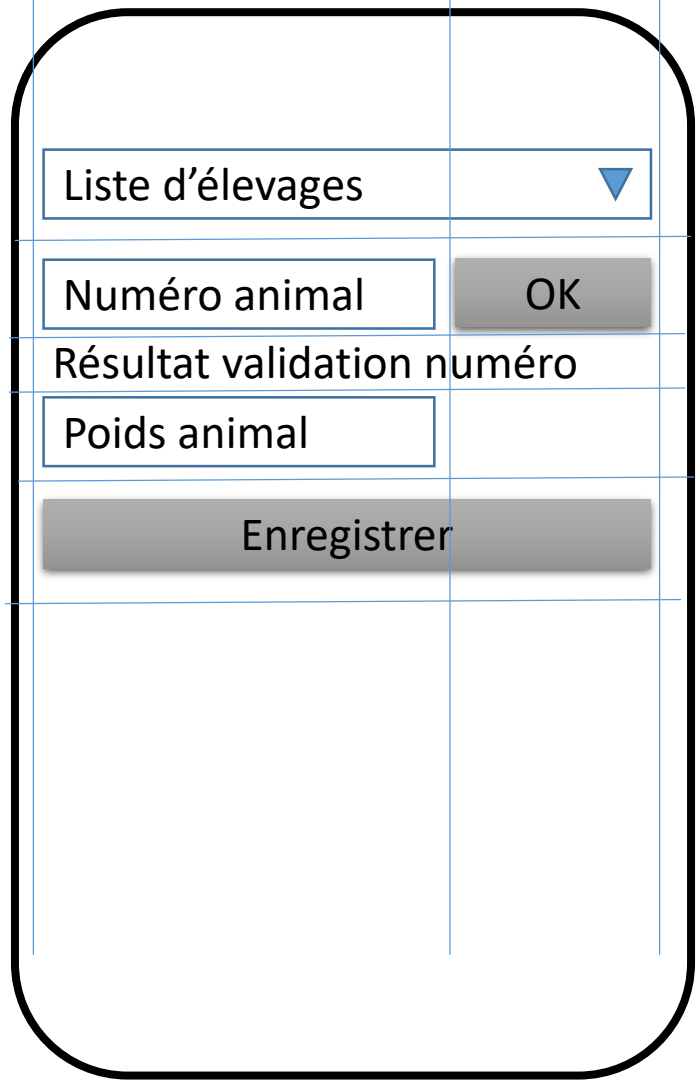
- Une grille (2 colonnes, 5 lignes) : `<Grid>`
- Une liste déroulante : `<Picker>`
- Deux champs de saisie : `<Entry>`
- Deux boutons : `<Button>`
- Un texte d'affichage : `<Label>`

Donc 1 vue à écrire dans le dossier View : `PeseeAnimalView`

Bien démarrer avec Xamarin

Construction de l'IHM

View



La grille :

- `<Grid ColumnDefinitions="75*,25*" RowDefinitions="auto,auto,auto,auto,auto">`

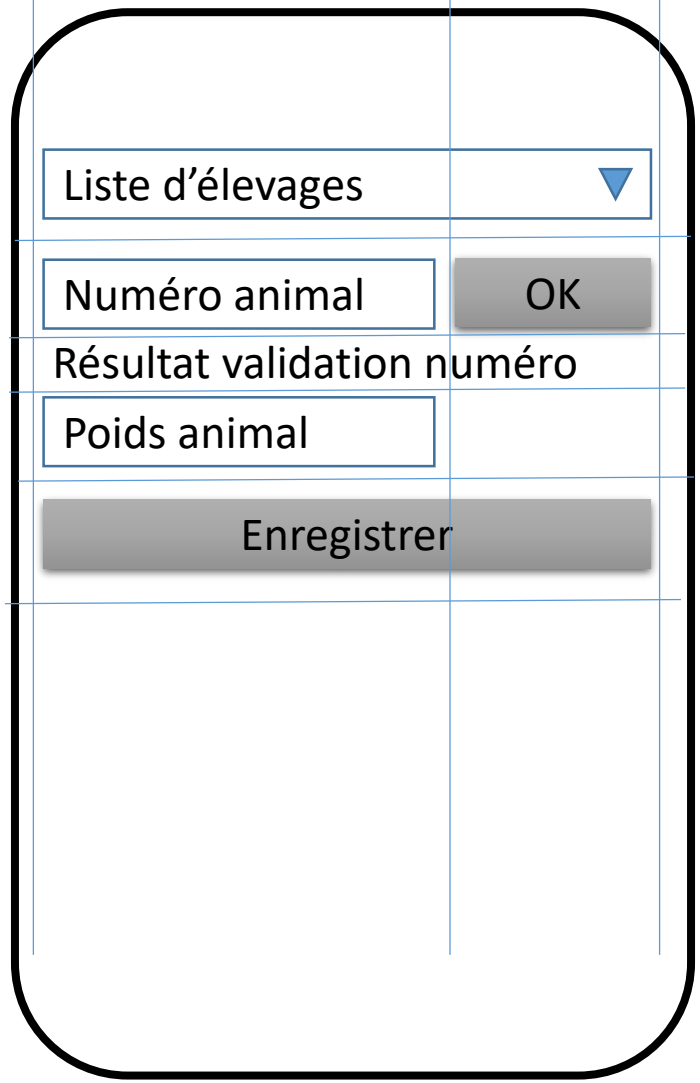
La liste déroulante :

- `<Picker Grid.Row="0" Grid.ColumnSpan="2" Title="Liste des élevages" ItemsSource="{Binding Elevages}" ItemDisplayBinding="{Binding ElevageNom}" SelectedItem="{Binding ElevageSelectionne}"/>`

Bien démarrer avec Xamarin

Construction de l'IHM

View



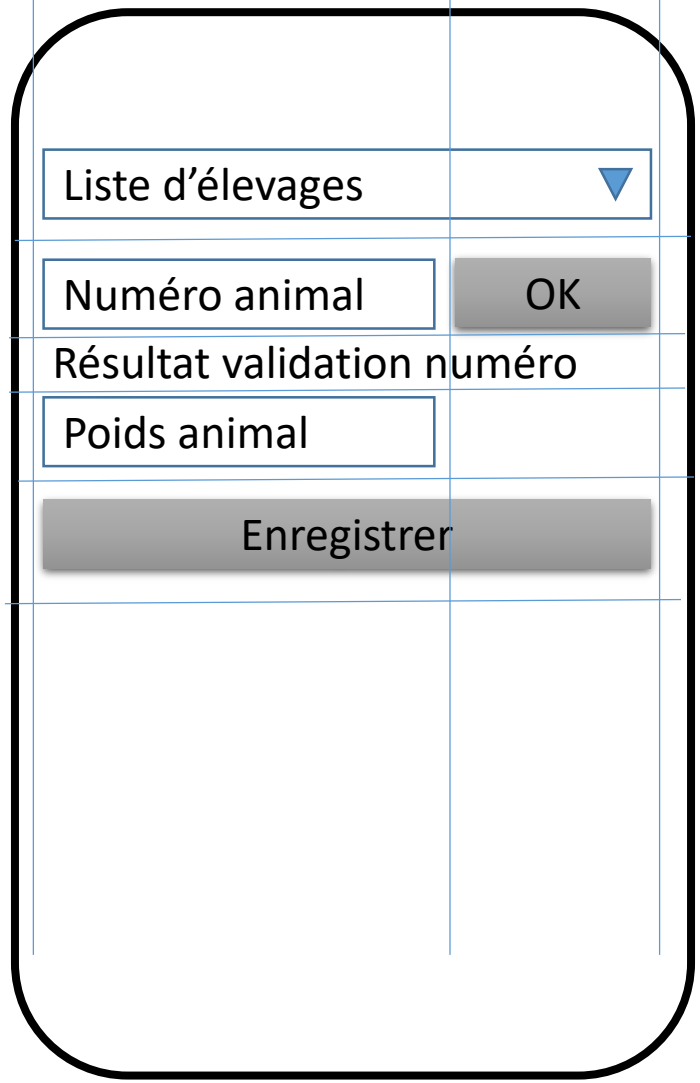
Les 2 champs de saisie :

```
<Entry Grid.Row="1"  Grid.Column="0" Placeholder="Numéro animal" Text="{Binding AnimalNumero}" Keyboard="Numeric"/>

<Entry Grid.Row="3" Grid.Column="0" Placeholder="Poids" Text="{Binding AnimalPoids}" Keyboard="Numeric"/>
```

Bien démarrer avec Xamarin

Construction de l'IHM

View



Les 2 boutons :

```
<Button Grid.Row="1" Grid.Column="1" Text="OK"
Command="{Binding ValiderNumeroAnimalCommand}"/>

<Button Grid.Row="4" Grid.ColumnSpan="2" Text="Enregistrer"
Command="{Binding EnregistrerCommand}"/>
```

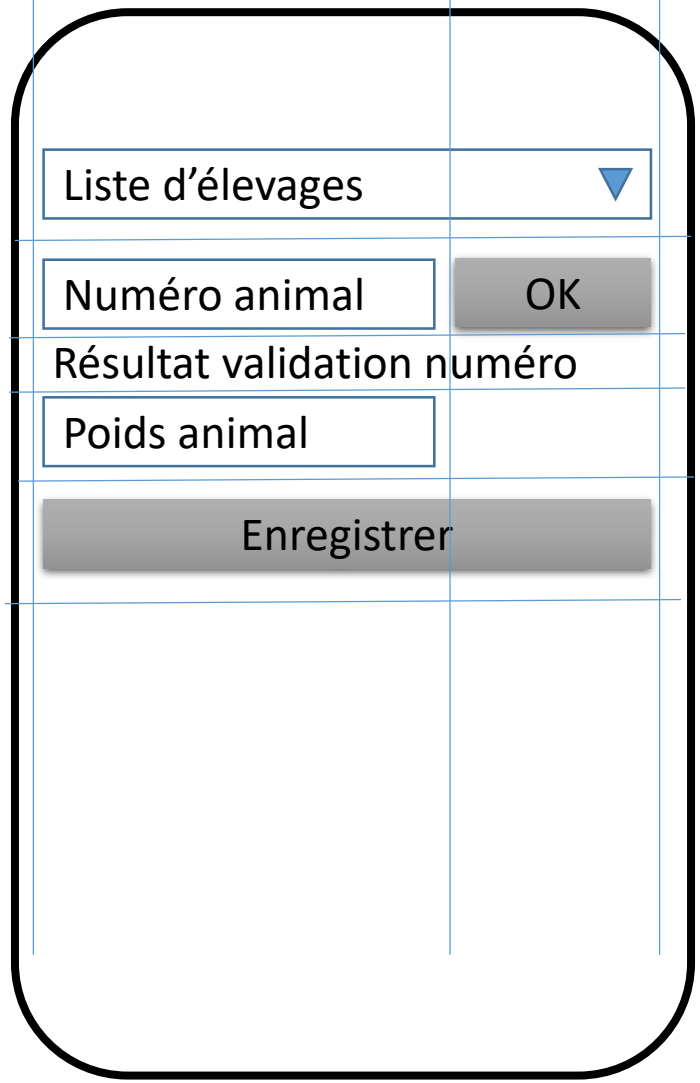
Le texte :

```
<Label Grid.Row="2" Grid.ColumnSpan="2"
Text="{Binding RetourValidation}"/>
```

Bien démarrer avec Xamarin

Construction de l'IHM

View



Pour que la vue soit appelée au démarrage de l'application :

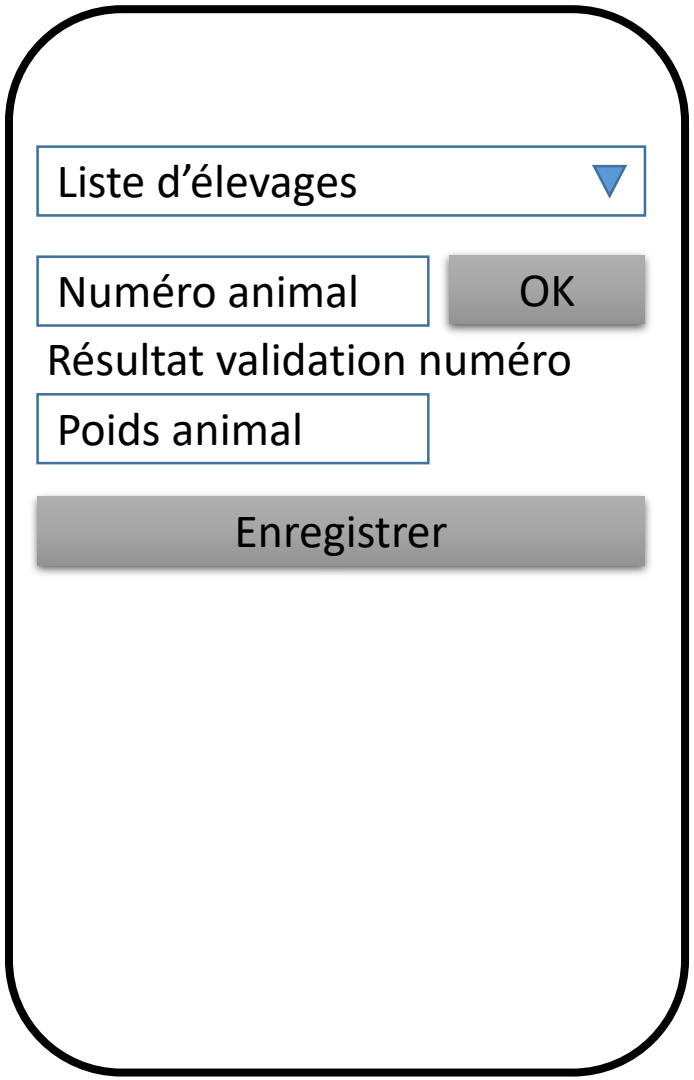
Dans le constructeur de la classe App, ajouter l'appel de la vue :

```
public App()
{
    InitializeComponent();

    MainPage = new PeseeAnimalView();
}
```

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Afin que le binding fonctionne correctement :

Créer la classe `BaseViewModel` dans le dossier ViewModel

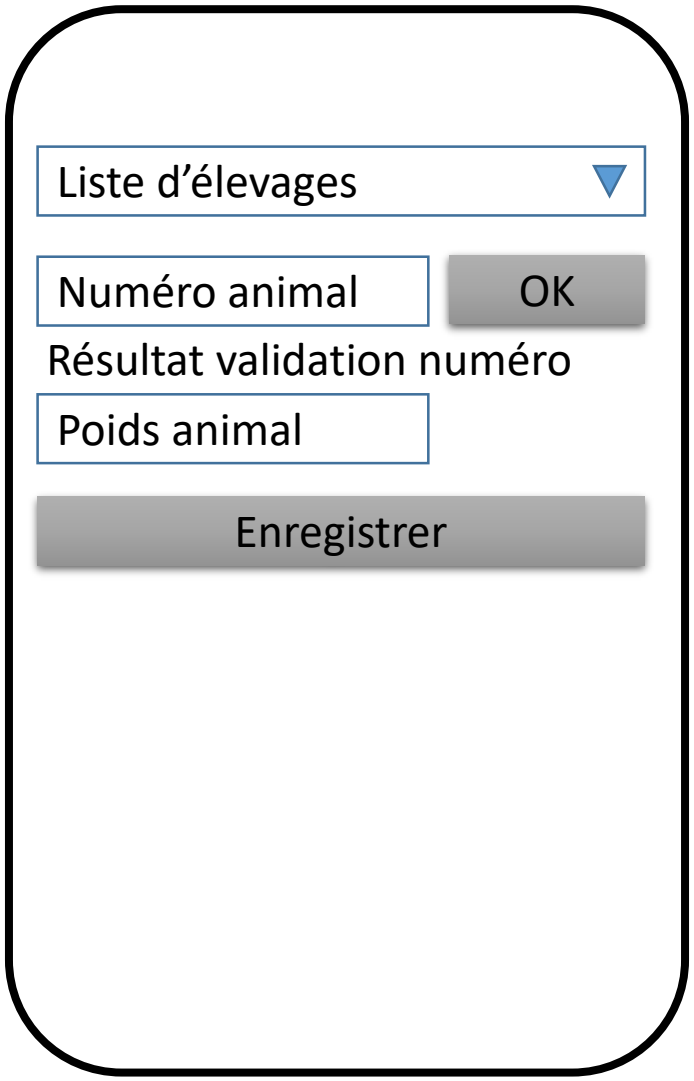
```
public class BaseViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;
    protected void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }

}
```

Cette classe sera héritée par toutes les autres classes ViewModel afin de bénéficier de la méthode `OnPropertyChanged`

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Afin d'interagir avec la vue `PeseeAnimalView` :

- Créer la classe `PeseeAnimalViewModel` dans le dossier ViewModel

```
public class PeseeAnimalViewModel : BaseViewModel
```

<u>Règle</u> : 1 ViewModel pour 1 View

- Instancier cette classe dans le constructeur de la vue :

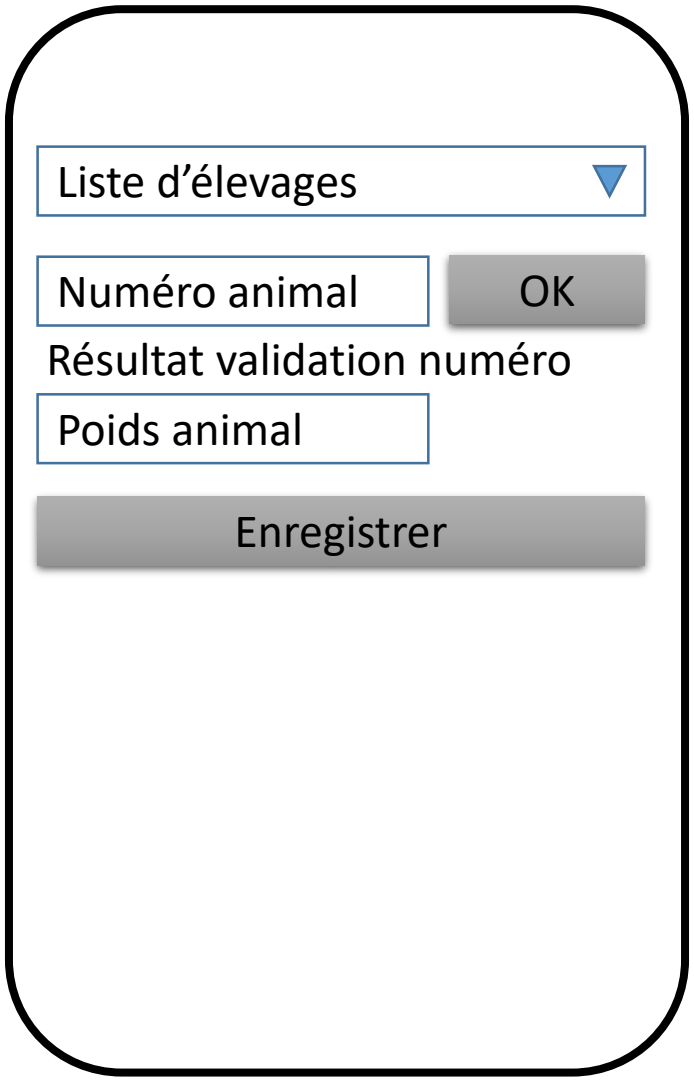
(dans le code Behind)

```
public PeseeAnimalView()
{
    InitializeComponent();
    peseeAnimalView.BindingContext = new PeseeAnimalViewModel();
}
```

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Afficher la liste des élevages dans la liste déroulante :

L'objet Picker de la vue a 3 propriétés en Binding

- `ItemsSource="{Binding Elevages}"`

`Elevages` est une `List<Elevage>` qu'il faut alimenter dans le constructeur par l'appel de la méthode `getElevages()` de `ElevageDAO`

- `ItemDisplayBinding="{Binding ElevageNom}"`

`ElevageNom` est la propriété de la classe `Elevage` qui sera affichée dans la liste déroulante

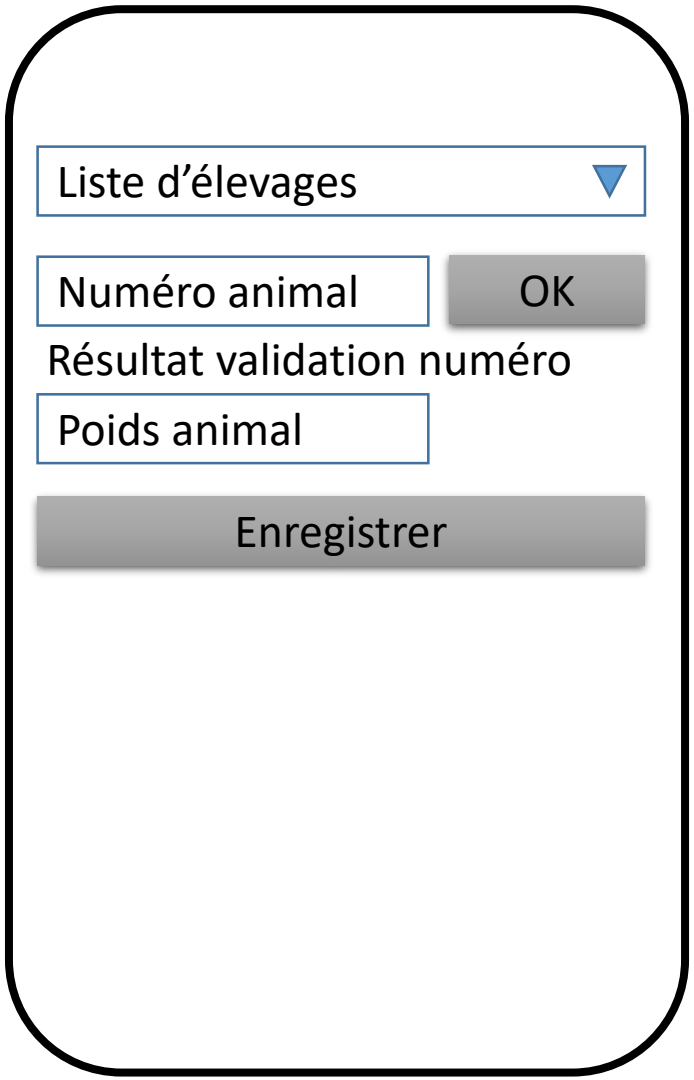
- `SelectedItem="{Binding ElevageSelectionne}"`

`ElevageSelectionne` est un objet de type `Elevage` qui correspondra à l'élevage sélectionné

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Lier les propriétés `Text` des 2 champs de saisie :

- `Text="{Binding AnimalNumero}"`

`AnimalNumero` est un `string` qui sera alimenté par la saisie de l'utilisateur

- `Text="{Binding AnimalPoids}"`

`AnimalPoids` est un `int` qui sera alimenté par la saisie de l'utilisateur

Lier la propriété `Text` du label :

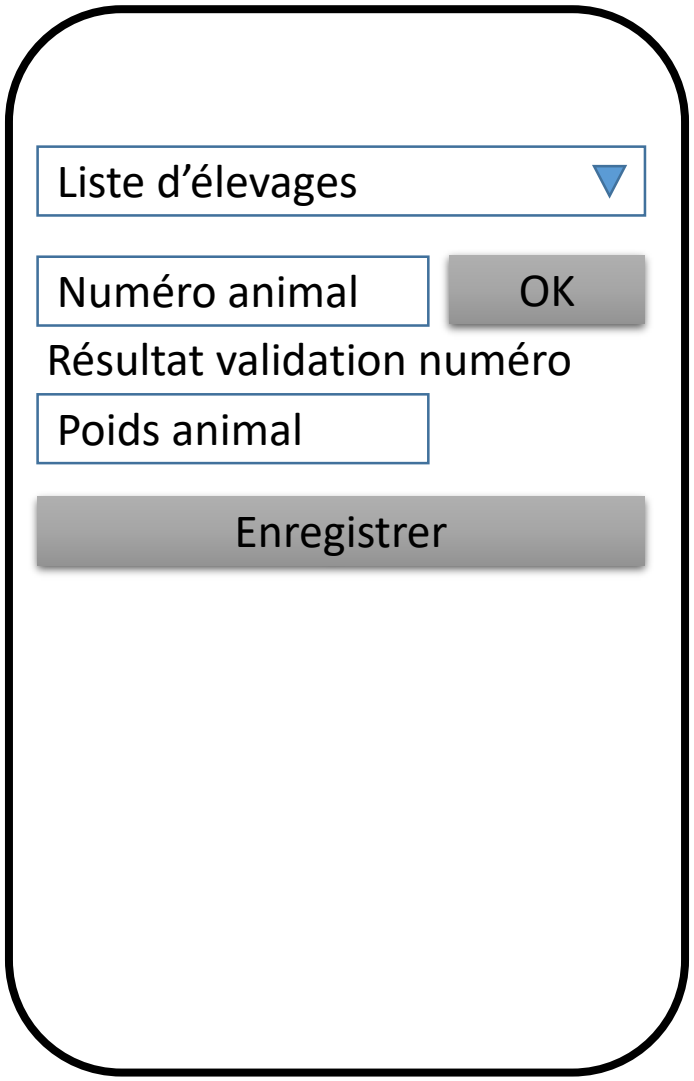
- `Text="{Binding RetourValidation}"`

`RetourValidation` est un `string` qui sera alimenté par la validation du bouton « OK »

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Faire fonctionner le bouton « OK » :

- `Command="{Binding ValiderNumeroAnimalCommand}"`

`ValiderNumeroAnimalCommand` est un objet de type `ICommand` qui permet de lier le bouton à une action et à une condition d'exécution

- Déclaration :

```
public ICommand ValiderNumeroAnimalCommand { get; private set; }
```

- Instanciation dans le construteur du ViewModel :

```
ValiderNumeroAnimalCommand = new Command(ValiderNumeroAnimal, CanValiderNumeroAnimal);
```

- 2 méthodes à écrire :

```
private void ValiderNumeroAnimal()
```

Permet de vérifier (et afficher) si le numéro saisi correspond bien à un animal de l'élevage sélectionné

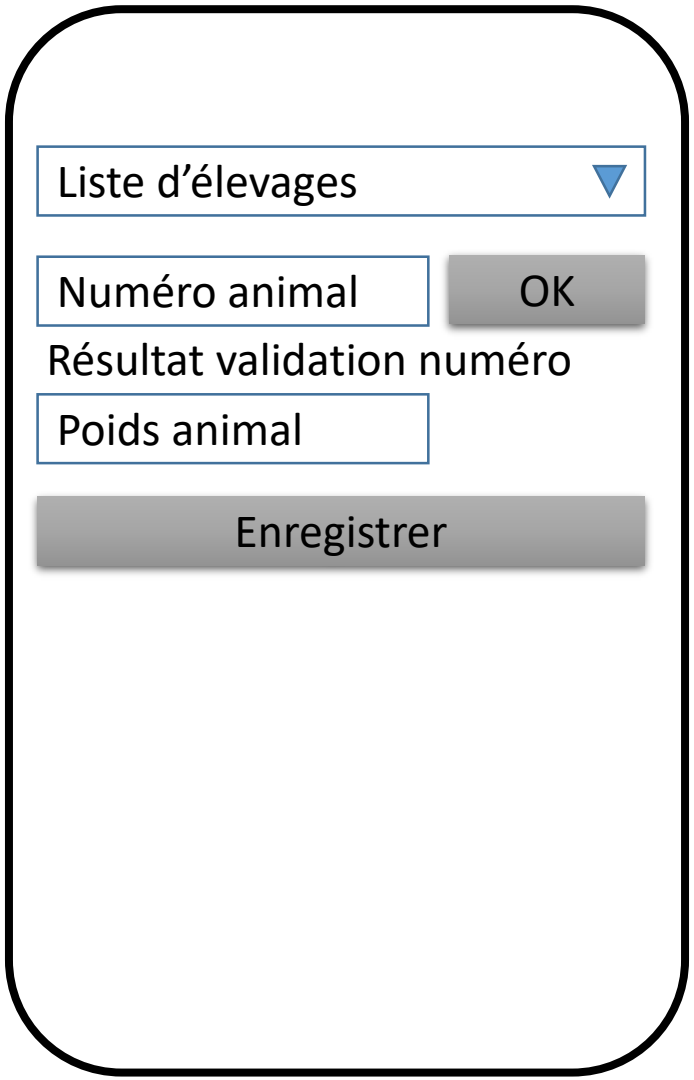
```
private bool CanValiderNumeroAnimal()
```

Retourne vrai si le numéro animal est saisi

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Faire fonctionner le bouton « Enregistrer » :

- `Command="{Binding EnregistrerCommand}"`

`EnregistrerCommand` est un objet de type `ICommand` qui permet de lier le bouton à une action et à une condition d'exécution

- Déclaration :

```
public ICommand EnregistrerCommand { get; private set; }
```

- Instanciation dans le construteur du ViewModel :

```
EnregistrerCommand = new Command(EnregistrerPesee, CanValiderNumeroAnimal);
```

- 2 méthodes à écrire :

```
private void EnregistrerPesee()
```

Permet d'enregistrer la pesée par l'appel de la méthode save() de `PeseeDAO`

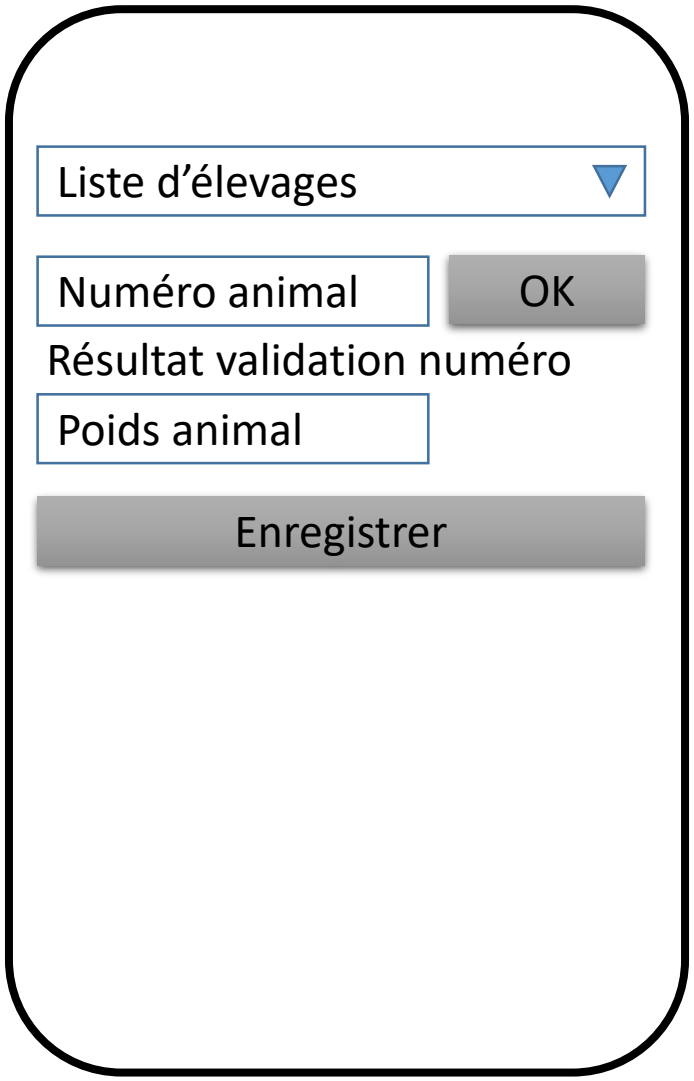
```
private bool CanValiderNumeroAnimal()
```

Retourne vrai si le numéro animal est saisi

Bien démarrer avec Xamarin

Interagir avec la vue

ViewModel



Et, enfin, pour que tout fonctionne bien :

Comme la condition sur les commandes dépend de la saisie ou non du numéro animal, lancer la méthode `ChangeCanExecute()` dans le set de AnimalNumero :

```
((Command)ValiderNumeroAnimalCommand).ChangeCanExecute();
((Command)EnregistrerCommand).ChangeCanExecute();
```

Bien démarrer avec Xamarin

Pour aller plus loin…

Changer la couleur d'un label dynamiquement (1/2) :

- Créer la classe `StringToColorConverter` (dans le dossier Converter)

```
public class StringToColorConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string valueAsString = value as String;

        if (valueAsString == null) return Color.Default;

        if (valueAsString.StartsWith("#")) return Color.FromHex(valueAsString);

        if (!valueAsString.Equals("")) return System.Drawing.Color.FromName(valueAsString);

        return Color.Default;
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        return null;
    }
}
```

Bien démarrer avec Xamarin

Pour aller plus loin…

Changer la couleur d'un label dynamiquement (2/2) :

- Dans la vue, mettre en place le binding sur `TextColor`

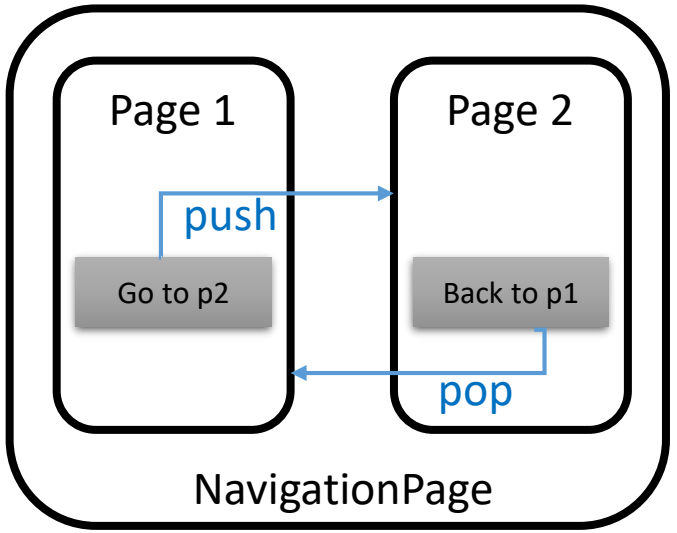
```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             ...
             xmlns:labelTextColorSample="clr-namespace:PetitProgramme.Converter"
             x:Name="peseeAnimalView">

    <ContentPage.Resources>
        <ResourceDictionary>
            ...
            <labelTextColorSample:StringToColorConverter x:Key="StringToColorConverter"/>
            ...
        </ResourceDictionary>
    </ContentPage.Resources>

        ...
<Label Grid.Row="3" Grid.ColumnSpan="2" Text="{Binding RetourValidation}"
                    TextColor="{Binding CouleurMessage, Converter={StaticResource StringToColorConverter}}"/>
        ...
```

- Dans le ViewModel, ajouter la propriété `CouleurMessage` et coder ses mises à jour

Bien démarrer avec Xamarin

Pour aller plus loin…

Ouvrir une page à partir d'une autre page :



- Dans le constructeur de la classe App, appeler la 1ère page en paramètre de `NavigationPage()` :

```
MainPage = new NavigationPage( new Page1());
```

- Dans le ViewModel coder le changement de page :

```
App.Current.MainPage.Navigation.PushAsync(new Page2());

App.Current.MainPage.Navigation.PopAsync();
```

Bien démarrer avec Xamarin

Pour aller plus loin…

Créer un style pour un contrôle :

```
<ContentPage.Resources>
    <ResourceDictionary>
        ...
        <x:Int16 x:Key="border">3</x:Int16>
        <Style x:Name="myButton" x:Key="myButton" TargetType="Button">
            <Setter Property="BorderWidth" Value="{DynamicResource border}"/>
            <Setter Property="CornerRadius" Value="10"/>
            <Setter Property="BackgroundColor" Value="#00A3A6"/>
            <Setter Property="BorderColor" Value="#0077A6"/>
            <Setter Property="TextColor" Value="White"/>
            <Setter Property="Margin" Value="5,0,5,0"/>
        </Style>
        ...
    </ResourceDictionary>
</ContentPage.Resources>

...

<Button Grid.Row="2" Grid.Column="1" Text="OK" Command="{Binding ValiderNumeroAnimalCommand}«
                    Style="{StaticResource myButton}"/>
```