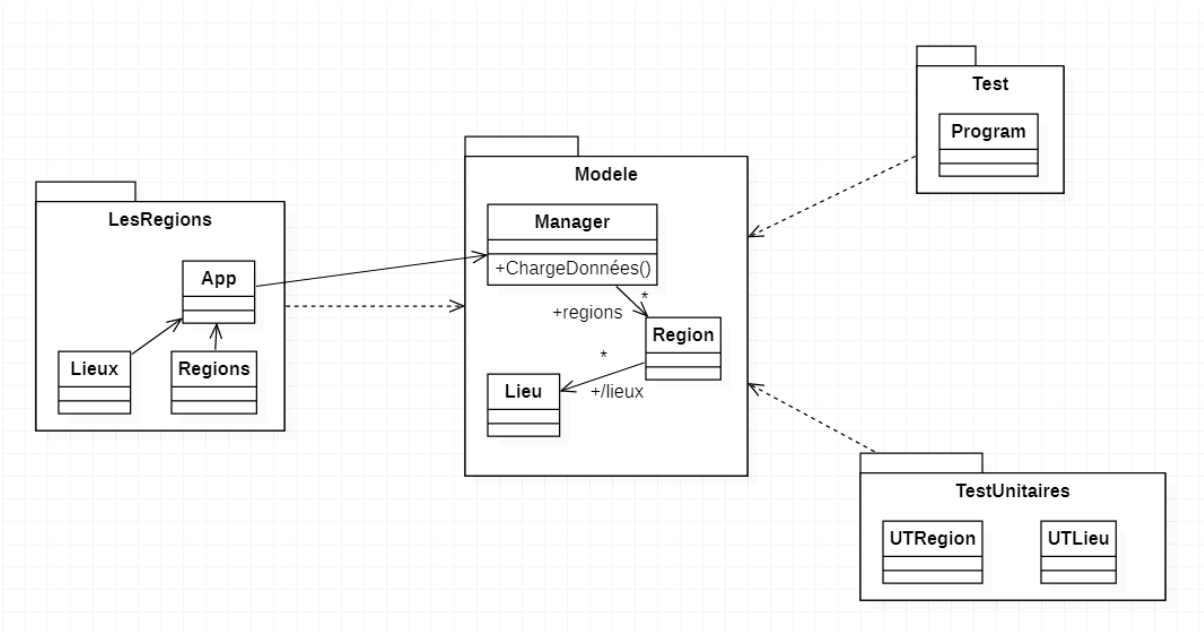
Conception et Programmation Orientées Objets (C#, .NET)

SOMMAIRE:

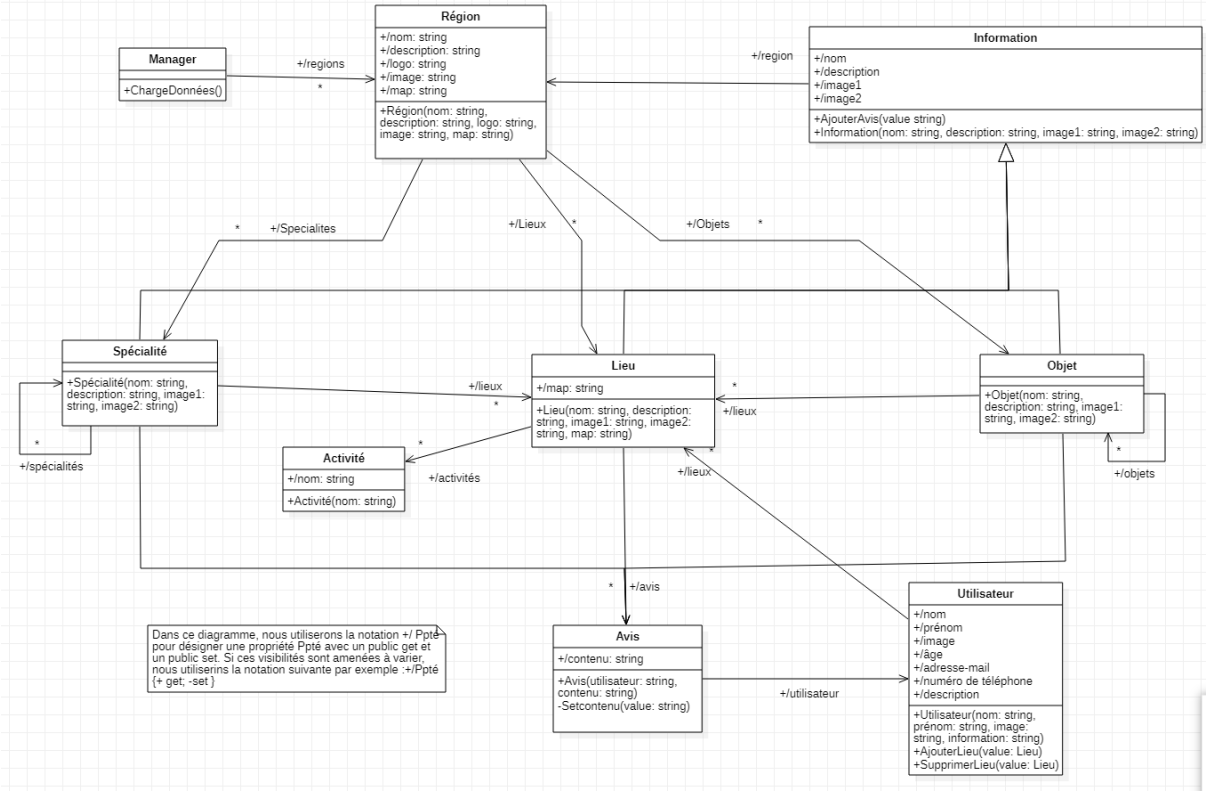
1) DIAGRAMME DE PAQUETAGE :



Description:

Le package LesRegions est l'ensemble de toutes nos vues et de l'application App avec leur code behind respectif. Il possède une dépendance avec Modele qui fait le lien entre le code XAML et les classes en C# pour permettre la gestion des actions de l'utilisateur (navigation, clics, remplissage, etc...) et la liaison de données (Data Binding). Pour avoir du contenu sur l'application on passe par le Manager qui est une classe qui va instancier une ReadOnlyCollection de région qui est une encapsulation d'une liste de région(regions) permettant de limiter les actions sur cette liste. Le manager possède une méthode ChargeDonnées, comme son nom l'indique elle va charger les données de l'application. Les données après sont disponibles sur les vues grâce à App qui instancie le manager. Pour finir, il reste plus qu'à faire le lien entre chaque fenêtre qui a besoin des données et du manager, pour cela on passe par App qui pointe déjà le manager. Ensuite il y a deux autres dépendances, une de Test à Modele et une autre de TestUnitaires à Modele. Ces deux package permettent de tester les affichages, les méthodes, les constructeurs, les propriétés… de chaque classe.

2) DIAGRAMME DE CLASSES :



Description:

Le but de notre application est de donner des informations sur les régions de la France, la classe principale est donc Région qui est composé de son nom, une description, son logo, une image, une map des département qui l'a compose et trois listes : une de spécialités régionales(Specialites), une de lieux(Lieux) et une d'objets atypiques(Objets). Les trois classes (Lieu, Specialite, Objet) héritent de la classe Information caractérisée par un nom, une description et deux images. L'héritage permet de factoriser les lignes de code de ces classes. La classe Objet a une liste d'objets et de lieux, de même spécialité a une liste de spécialités et de lieux. Lieu possède en plus des attribut d'information une map et une liste d'activités qui sont caractérisées par un nom. Chaque classe héritant d'informations a une liste d'avis. Un avis possède un utilisateur et un contenu. Pour finir le manager possède une liste de régions pour permettre de charger toutes les données.

3) DIAGRAMME DE SÉQUENCE :

4) DESCRIPTION ÉCRITE DE L'ARCHITECTURE

EXEMPLE DE TD:

```
namespace appli
{
    10 références
    public class Navigator : INotifyPropertyChanged
    {
        13 références
        public enum Etat
        {
            VIDE,
            DETAIL,
            AJOUT
        }

        public static Dictionary<Etat, Func<UserControl>> FactoryUC = new Dictionary<Etat,
          Func<UserControl>>
        {
            [Etat.VIDE] = () => null,
            [Etat.DETAIL] = () => new UCDetail(),
            [Etat.AJOUT] = () => new UCAjout()
        };
```

```
public event PropertyChangedEventHandler PropertyChanged;
1 référence
public void OnPropertyChanged([CallerMemberName] string propertyName = null) =>
  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

4 références
public Etat EtatCourant
{
    get => etatCourant;
    set
    {
        if(value != etatCourant)
        {
            etatCourant = value;
            OnPropertyChanged();
        }
    }
}
private Etat etatCourant;


public Navigator()
{
    EtatCourant = Etat.VIDE;
}
```

```
        Title="En Cuisine" Height="450" Width="800">
    <DockPanel>
        <uc:UCMaster DockPanel.Dock="Left"/>
        <ContentControl Content="{Binding EtatCourant, Converter={StaticResource convEtat2UC}}">
            <!--<uc:UCDetail/>-->
            <!--<uc:UCAjout/>-->
        </ContentControl>
    </DockPanel>
</Window>
```

```
namespace appli.converters
{
    0 références
    class ConvEtat2UC : IValueConverter
    {
        1 référence
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            Etat e = (Etat)value;
            return Navigator.FactoryUC[e]();
        }

        1 référence
        public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

Converteur image:

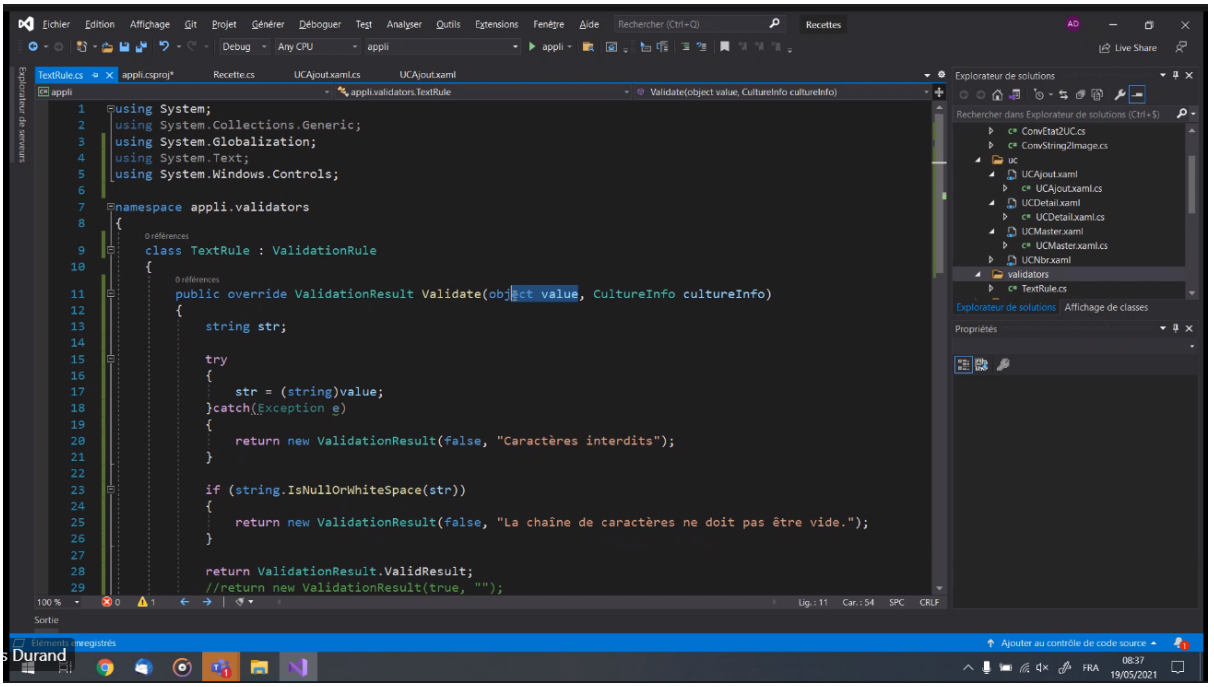
```
public class ConvString2Image : IValueConverter
{
    1 référence
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string nomImage = (value as string);

        if (string.IsNullOrWhiteSpace(nomImage))
        {
            nomImage = "no-image.png";
        }

        string dossierImage = Path.Combine(Directory.GetCurrentDirectory(), "..\\img\\");
        string cheminImage = Path.Combine(dossierImage, nomImage);

        return new Uri(cheminImage, UriKind.RelativeOrAbsolute);
    }
```

Validator:




```
<TextBox Grid.Column="3" Grid.Row="0">
    <TextBox.Text>
        <Binding Path="Nom" UpdateSourceTrigger="PropertyChanged">
            <Binding.ValidationRules>
                <valid:TextRule/>
            </Binding.ValidationRules>
        </Binding>
    </TextBox.Text>
</TextBox>
```

Changer la couleur quand ne verifie pas

```
<Style TargetType="TextBox">
    <Setter Property="Margin" Value="0,2,5,2"/>
    <Setter Property="MaxHeight" Value="25"/>
    <Style.Triggers>
        <Trigger Property="Validation.HasError" Value="true">
            <Setter Property="BorderBrush" Value="Yellow"/>
            <Setter Property="Background" Value="Pink"/>
            <Setter Property="ToolTip" Value="{Binding RelativeSource={x:Static
              RelativeSource.Self}, Path=(Validation.Errors)[0].ErrorContent}"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

multi binding

```
<Button Style="{StaticResource bigButton}" Background="YellowGreen" DockPanel.Dock="Right"
 ToolTip="Sauvegarder" Click="SauverButton_Click">
    <Button.Content>
        <icon:PackIconUnicons Kind="Save" />
    </Button.Content>
    <Button.IsEnabled>
        <MultiBinding Converter="{StaticResource convValid2Bool}">
            <Binding ElementName="TBNom" Path="(Validation.HasError)"/>
            <Binding ElementName="TBNbParts" Path="(Validation.HasError)"/>
        </MultiBinding>
    </Button.IsEnabled>
</Button>
```

diagramme de séquence sur la persistance

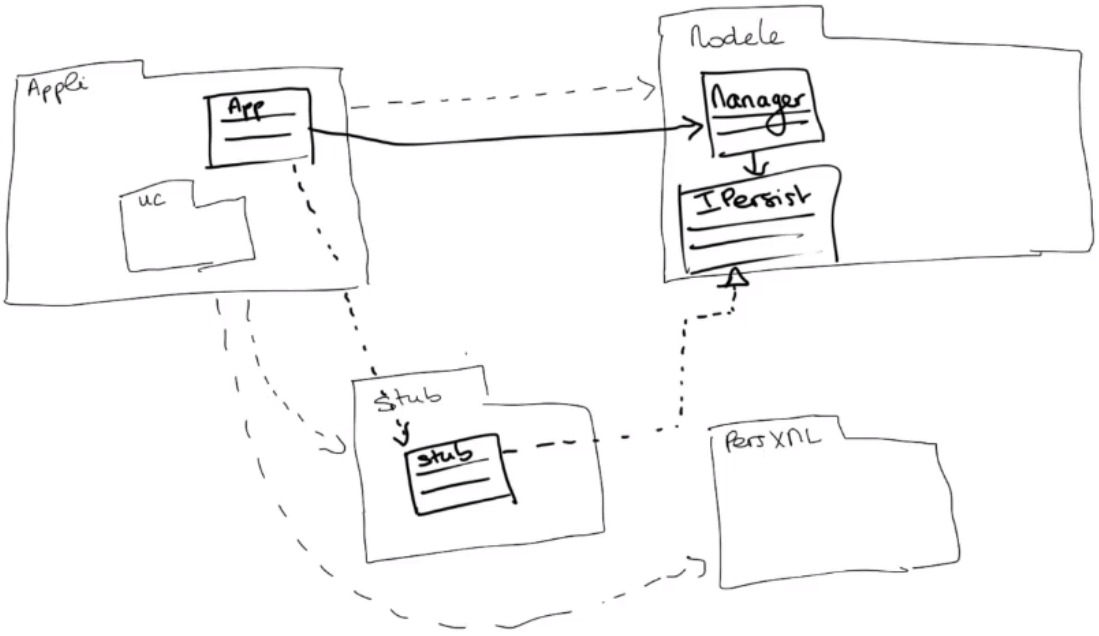


Diagramme de paquetages