Travail de diplôme 2020 - Scrum'o'wall
Code Source

Gaël Mariot
T.IS-E2A
École d'informatique (CFPT-I)

8 Mai

Table des matières

1 Base

1.1 App.xaml

```
<Application x:Class="Scrum_o_wall.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Scrum_o_wall"
             StartupUri="Views/MainMenu.xaml">
    <Application.Resources>

    </Application.Resources>
</Application>
```

Listing 1 – ../../Scrum'o'wall/Scrum'o'wall/App.xaml

1.2 App.xaml.cs

```
using System.Windows;

namespace Scrum_o_wall
{
    /// <summary>
    /// Logique d'interaction pour App.xaml
    /// </summary>
    public partial class App : Application
    {
    }
}
```

Listing 2 – ../../Scrum'o'wall/Scrum'o'wall/App.xaml.cs

1.3 App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
</configuration>
```

Listing 3 – ../../Scrum'o'wall/Scrum'o'wall/App.config

2 Vues

2.1 ActivitiesMenu.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    ActivitiesMenu.xaml
 * Desc.     :    This file contains the basic template of the ActivitiesMenu view
    -->
<Window x:Class="Scrum_o_wall.Views.ActivitiesMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
       Title="Visualisation␣des␣activit s" ↩
           WindowStartupLocation="CenterScreen" Height="450" Width="447.5" ↩
           FontSize="16">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="70"></RowDefinition>
            <RowDefinition/>
            <RowDefinition Height="50"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" ↩
           Margin="10,16" TextAlignment="Center" FontSize="28" ↩
           Height="38">Visualisation des activit s</TextBlock>
        <Border Grid.Row="1" Grid.ColumnSpan="2" Margin="10" ↩
           BorderThickness="1" BorderBrush="{DynamicResource␣{x:Static␣↩
           SystemColors.GrayTextBrushKey}}">
            <ListView x:Name="lstActivities" ↩
               MouseDoubleClick="LstActivities_MouseDoubleClick" ↩
               TouchUp="LstActivities_MouseDoubleClick">
            </ListView>
        </Border>

        <Button x:Name="btnCancel" Grid.Row="2" Grid.ColumnSpan="2" Margin="10" ↩
           Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Retour</Button>
    </Grid>
</Window>
```

Listing 4 – ../../Scrum'o'wall/Scrum'o'wall/Views/ActivitiesMenu.xaml

2.2 ActivitiesMenu.xaml.cs

```
/*
 * Author    :    Ga l  Serge  Mariot
 * Project   :    Scrum'o'wall
 * File      :    ActivitiesMenu.xaml.cs
 * Desc.     :    This file contains the logic in the ActivitiesMenu view
 */
using Scrum_o_wall.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour ActivitiesMenu.xaml
    /// </summary>
    public partial class ActivitiesMenu : Window
    {
        private readonly List<Activity> activities;
        public ActivitiesMenu(List<Activity> someActivities)
        {
            activities = someActivities;
            InitializeComponent();
            foreach (Activity a in activities)
```

```
26              {
27                  lstActivities.Items.Add(a);
28              }
29          }
30
31          private void BtnCancel_Click(object sender, EventArgs e)
32          {
33
34              Close();
35          }
36          private void LstActivities_MouseDoubleClick(object sender, EventArgs e)
37          {
38              ListBox lbx = sender as ListBox;
39              if (lbx.SelectedItem != null)
40              {
41                  Activity activity = lbx.SelectedItem as Activity;
42                  MessageBox.Show(string.Format("Date : {0}\n{1}", ↩
                        activity.DateTime, activity.Description), "Activit ", ↩
                        MessageBoxButton.OK);
43              }
44          }
45      }
46  }
```

Listing 5 – ../../Scrum'o'wall/Scrum'o'wall/Views/ActivitiesMenu.xaml.cs

2.3 BurndownChart.xaml

```
1   <!--
2    * Author    :    Ga l  Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    BurndownChart.xaml
5    * Desc.     :    This file contains the basic template of the BurndownChart view
6        -->
7   <Window x:Class="Scrum_o_wall.Views.BurndownChart"
8           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
13          mc:Ignorable="d"
14          Title="Scrum'o'Wall" Height="450" Width="800" WindowState="Maximized" ↩
                WindowStyle="None" WindowStartupLocation="CenterScreen" ↩
                FontSize="16" ResizeMode="NoResize">
15      <Window.ContextMenu >
16          <ContextMenu >
17
18              <Separator/>
19              <MenuItem Header="Retour" Name="Quit" Click="Quit_Click"/>
20          </ContextMenu>
21      </Window.ContextMenu>
22      <Grid>
23          <Line x:Name="lnIdeal" X1="120" Y1="70" StrokeThickness="2" X2="730" ↩
                Y2="330" Stroke="Red"></Line>
24          <Polyline x:Name="lnCurrent" Points="120,70,300,300" ↩
                StrokeThickness="2" Stroke="Black">
25
26          </Polyline>
27          <Border x:Name="brdrGraphic" BorderBrush="Gray" ↩
                BorderThickness="2,0,0,2" Margin="120,70,70,120"/>
28          <Label Content="Complexit " VerticalAlignment="Center"></Label>
29          <Label Content="Jour" VerticalAlignment="Bottom" ↩
                HorizontalAlignment="Center" Margin="383,0,377,91" ↩
                RenderTransformOrigin="0.575,-1.226"></Label>
30          <Label VerticalAlignment="Bottom" Margin="120,0,0,100" ↩
                Name="lblDayBegin">DD.MM</Label>
31          <Label VerticalAlignment="Bottom" Margin="0,0,70,100" ↩
                HorizontalAlignment="Right" Name="lblDayEnd">DD.MM</Label>
32
33          <Label VerticalAlignment="Top"    Margin="90,60,0,0"  ↩
                HorizontalAlignment="Left" Name="lblComplexityMax">XX</Label>
34          <Label VerticalAlignment="Bottom" Margin="100,0,0,120"    ↩
                HorizontalAlignment="Left" Name="lblComplexityZero">0</Label>
35
36          <Button x:Name="btnCancel" Click="Quit_Click" TouchUp="Quit_Click" ↩
                Content="â  " VerticalContentAlignment="Top" ↩
```

```
            HorizontalAlignment="Center" Margin="10" Width="75" Height="75" ↩
            FontSize="45" VerticalAlignment="Bottom">
37              <Button.Resources>
38                  <Style TargetType="Border">
39                      <Setter Property="CornerRadius" Value="50"/>
40                  </Style>
41              </Button.Resources>
42          </Button>
43      </Grid>
44  </Window>
```

Listing 6 – ../../Scrum'o'wall/Scrum'o'wall/Views/BurndownChart.xaml

2.4 BurndownChart.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    BurndownChart.xaml.cs
5    * Desc.     :    This file contains the logic in the BurndownChart view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Collections.Generic;
10  using System.Windows;
11  using System.Windows.Media;
12
13  namespace Scrum_o_wall.Views
14  {
15      /// <summary>
16      /// Logique d'interaction pour BurndownChart.xaml
17      /// </summary>
18      public partial class BurndownChart : Window
19      {
20          private readonly Sprint sprint;
21          public BurndownChart(Sprint aSprint)
22          {
23              sprint = aSprint;
24              InitializeComponent();
25              Loaded += BurndownChart_Loaded;
26
27              lblDayBegin.Content = sprint.Begin.ToString("dd.MM");
28              lblDayEnd.Content = sprint.End.ToString("dd.MM");
29
30          }
31
32
33          private void BurndownChart_Loaded(object sender, RoutedEventArgs e)
34          {
35              double daysSinceBegin = (DateTime.Now - sprint.Begin).TotalDays;
36              double totalDays = (sprint.End - sprint.Begin).TotalDays;
37              double elapsedDays = Math.Min(totalDays, daysSinceBegin);
38
39              int totalComplexity = 0;
40              int completedComplexity = 0;
41
42              foreach (KeyValuePair<int, UserStory> keyValuePair in ↩
                    sprint.OrderedUserStories)
43              {
44                  totalComplexity += keyValuePair.Value.ComplexityEstimation;
45                  completedComplexity += keyValuePair.Value.CompletedComplexity;
46              }
47
48              lblComplexityMax.Content = totalComplexity.ToString();
49              double x2Line = elapsedDays / totalDays * (ActualWidth - ↩
                    brdrGraphic.Margin.Left - brdrGraphic.Margin.Right) + ↩
                    brdrGraphic.Margin.Left;
50              double y2Line = completedComplexity / (double)totalComplexity * ↩
                    (ActualHeight - brdrGraphic.Margin.Bottom - ↩
                    brdrGraphic.Margin.Top) + brdrGraphic.Margin.Top;
51
52              PointCollection linePoints = new PointCollection
53              {
54                  new Point(brdrGraphic.Margin.Left, brdrGraphic.Margin.Top),
55                  new Point(x2Line, y2Line)
56              };
57
```

```
58              lnIdeal.X2 = ActualWidth - brdrGraphic.Margin.Right;
59              lnIdeal.Y2 = ActualHeight - brdrGraphic.Margin.Bottom;
60
61              lnCurrent.Points = linePoints;
62
63          }
64          private void Quit_Click(object sender, EventArgs e)
65          {
66
67              Close();
68          }
69
70      }
71  }
```

Listing 7 – ../../Scrum'o'wall/Scrum'o'wall/Views/BurndownChart.xaml.cs

2.5 ChecklistCreate.xaml

```
1  <!--
2   * Author   :   Ga l Mariot
3   * Project  :   Scrum'o'wall
4   * File     :   ChecklistCreate.xaml
5   * Desc.    :   This file contains the basic template of the ChecklistCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.ChecklistCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr er une checklist" WindowStartupLocation="CenterScreen" ↩
               Height="450" Width="447.5" FontSize="16">
15      <Grid>
16          <Grid.RowDefinitions>
17              <RowDefinition Height="70*"></RowDefinition>
18              <RowDefinition Height="40*"></RowDefinition>
19              <RowDefinition Height="259*"></RowDefinition>
20              <RowDefinition Height="50*"></RowDefinition>
21          </Grid.RowDefinitions>
22          <Grid.ColumnDefinitions>
23              <ColumnDefinition Width="119*"></ColumnDefinition>
24              <ColumnDefinition Width="321*"></ColumnDefinition>
25          </Grid.ColumnDefinitions>
26          <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" ↩
               Margin="10,22" TextAlignment="Center" FontSize="20" Height="26" ↩
               >Cr er une checklist</TextBlock>
27          <TextBlock Grid.Row="1" VerticalAlignment="Center" Margin="10" ↩
               TextAlignment="Right" Height="20" >Nom :</TextBlock>
28          <TextBlock Grid.Row="2" VerticalAlignment="Top" Margin="10,10,10,0" ↩
               TextAlignment="Right" Height="21" >Objets :</TextBlock>
29
30          <TextBox Grid.Row="1" MaxLength="255" Grid.Column="1" Margin="10" ↩
               Name="tbxName" Stylus.IsPressAndHoldEnabled="False"></TextBox>
31          <Grid Grid.Row="2" Grid.Column="1">
32              <Grid.RowDefinitions>
33                  <RowDefinition Height="107*"></RowDefinition>
34                  <RowDefinition Height="25*"></RowDefinition>
35              </Grid.RowDefinitions>
36              <ListView x:Name="listItems" Margin="10" ↩
                   Stylus.IsPressAndHoldEnabled="False">
37              </ListView>
38              <Button x:Name="btnAddItem" Grid.Row="1" Margin="10" ↩
                   TouchUp="BtnAddItem_Click" Click="BtnAddItem_Click">Ajouter un ↩
                   objet</Button>
39          </Grid>
40
41          <Button x:Name="btnCancel" Grid.Row="3" Margin="10" ↩
               TouchUp="BtnCancel_Click" Click="BtnCancel_Click">Annuler</Button>
42          <Button x:Name="btnConfirm" Grid.Row="3" Grid.Column="1" Margin="10" ↩
               TouchUp="BtnConfirm_Click" Click="BtnConfirm_Click">Confirmer</Button>
43      </Grid>
44 </Window>
```

Listing 8 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistCreate.xaml

2.6 ChecklistCreate.xaml.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    ChecklistCreate.xaml.cs
 * Desc.     :    This file contains the logic in the ChecklistCreate view
 */
using Scrum_o_wall.Classes;
using System;
using System.Collections.Generic;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour ChecklistCreate.xaml
    /// </summary>
    public partial class ChecklistCreate : Window
    {
        public List<ChecklistItem> itemsToAdd;
        public ChecklistCreate()
        {
            InitializeComponent();

            itemsToAdd = new List<ChecklistItem>();

            Refresh();
        }

        private void Refresh()
        {
            listItems.Items.Clear();
            foreach (ChecklistItem item in itemsToAdd)
            {
                listItems.Items.Add(item);
            }
        }

        private void BtnAddItem_Click(object sender, EventArgs e)
        {
            ChecklistItemCreate checklistItemCreate = new ChecklistItemCreate();
            if (checklistItemCreate.ShowDialog() == true)
            {
                string name = checklistItemCreate.tbxObjet.Text.Trim();
                ChecklistItem checklistItem = new ChecklistItem(-1, name, ↩
                    false, -1);

                itemsToAdd.Add(checklistItem);
                Refresh();
            }
        }
        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            if (tbxName.Text.Trim().Length > 0 && listItems.Items.Count > 0)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                    "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

    }
}
```

Listing 9 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistCreate.xaml.cs

2.7 ChecklistEdit.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    ChecklistEdit.xaml
 * Desc.     :    This file contains the basic template of the ChecklistEdit view
    -->
<Window x:Class="Scrum_o_wall.Views.ChecklistEdit"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Modifier une checklist" WindowStartupLocation="CenterScreen" Height="535.5" Width="537.5" FontSize="16">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="70*"></RowDefinition>
            <RowDefinition Height="40*"></RowDefinition>
            <RowDefinition Height="269*"></RowDefinition>
            <RowDefinition Height="50*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="119*"></ColumnDefinition>
            <ColumnDefinition Width="321*"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" Margin="10,21" TextAlignment="Center" FontSize="20" >Modifier une checklist</TextBlock>
        <TextBlock Grid.Row="1" VerticalAlignment="Center" Margin="10,12" TextAlignment="Right" >Nom :</TextBlock>
        <TextBlock Grid.Row="2" VerticalAlignment="Top" Margin="10,10,10,0" TextAlignment="Right" >Objets :</TextBlock>
        <TextBox Grid.Row="1" Grid.Column="1" MaxLength="255" Margin="10" Name="tbxName" Stylus.IsPressAndHoldEnabled="False"></TextBox>
        <Grid Grid.Row="2" Grid.Column="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="229*"></RowDefinition>
                <RowDefinition Height="50*"></RowDefinition>
            </Grid.RowDefinitions>
            <ListView x:Name="listItems" Margin="10" Stylus.IsPressAndHoldEnabled="False">
            </ListView>
            <Button x:Name="btnAddItem" Grid.Row="1" Margin="10" TouchUp="BtnAddItem_Click" Click="BtnAddItem_Click">Ajouter un objet</Button>
        </Grid>

        <Grid Grid.Row="3" Grid.ColumnSpan="2">
            <Grid.ColumnDefinitions>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition Width="2*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Annuler</Button>
            <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" Click="BtnDelete_Click" TouchUp="BtnDelete_Click">Supprimer</Button>
            <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" Click="BtnConfirm_Click" TouchUp="BtnConfirm_Click">Confirmer</Button>
        </Grid>
    </Grid>
</Window>
```

Listing 10 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistEdit.xaml

2.8 ChecklistEdit.xaml.cs

```
/*
 * Author    :    Ga l  Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    ChecklistEdit.xaml.cs
 * Desc.     :    This file contains the logic in the ChecklistEdit view
 */
```

```
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Collections.Generic;
10 using System.Data;
11 using System.Windows;
12 using System.Windows.Controls;
13 using System.Windows.Input;
14
15 namespace Scrum_o_wall.Views
16 {
17     /// <summary>
18     /// Logique d'interaction pour ChecklistEdit.xaml
19     /// </summary>
20     public partial class ChecklistEdit : Window
21     {
22         private readonly Checklist checklist;
23         private readonly Controller controller;
24         private readonly UserStory userStory;
25         private readonly List<ChecklistItem> itemsToAdd;
26         public bool Deleted = false;
27         public ChecklistEdit(Checklist aChecklist, UserStory aUserStory, ↩
               Controller aController)
28         {
29             checklist = aChecklist;
30             controller = aController;
31             userStory = aUserStory;
32
33             InitializeComponent();
34
35             itemsToAdd = new List<ChecklistItem>();
36
37             tbxName.Text = checklist.Name;
38             listItems.MouseDoubleClick += ListItems_MouseDoubleClick;
39
40             Refresh();
41         }
42
43         private void Refresh()
44         {
45             listItems.Items.Clear();
46             foreach (ChecklistItem item in checklist.ChecklistItems)
47             {
48                 listItems.Items.Add(item);
49             }
50             foreach (ChecklistItem item in itemsToAdd)
51             {
52                 listItems.Items.Add(item);
53             }
54         }
55
56         private void ListItems_MouseDoubleClick(object sender, ↩
               MouseButtonEventArgs e)
57         {
58             ListView lstView = sender as ListView;
59             ChecklistItem checklistItem = lstView.SelectedItem as ChecklistItem;
60             if (checklist.ChecklistItems.Contains(checklistItem))
61             {
62                 ChecklistItemEdit checklistItemEdit = new ↩
                       ChecklistItemEdit(checklistItem, userStory, controller);
63                 if (checklistItemEdit.ShowDialog() == true)
64                 {
65                     if (checklistItemEdit.Deleted)
66                     {
67                         controller.Delete(checklistItem);
68                     }
69                     else
70                     {
71                         string objet = checklistItemEdit.tbxObjet.Text.Trim();
72                         bool done = checklistItemEdit.chkbxDone.IsChecked == true;
73                         controller.UpdateCheckListItem(objet, done, ↩
                               checklistItem);
74                     }
75                     Refresh();
76                 }
77             }
78         }
79         private void BtnAddItem_Click(object sender, EventArgs e)
80         {
81             ChecklistItemCreate checklistItemCreate = new ChecklistItemCreate();
```

```
82              if (checklistItemCreate.ShowDialog() == true)
83              {
84                  ChecklistItem checklistItem = new ChecklistItem(-1, ↩
                        checklistItemCreate.tbxObjet.Text.Trim(), false, -1);
85
86                  itemsToAdd.Add(checklistItem);
87                  Refresh();
88              }
89          }
90          private void BtnCancel_Click(object sender, EventArgs e)
91          {
92
93              Close();
94          }
95          private void BtnConfirm_Click(object sender, EventArgs e)
96          {
97              if (tbxName.Text.Trim().Length > 0 && listItems.Items.Count > 0)
98              {
99                  DialogResult = true;
100                 Close();
101             }
102             else
103             {
104                 MessageBox.Show("Le␣nom␣n'est␣pas␣rempli␣ou␣il␣n'y␣a␣aucun␣↩
                        objet␣dans␣la␣liste.", "Erreur", MessageBoxButton.OK, ↩
                        MessageBoxImage.Error);
105             }
106         }
107
108         private void BtnDelete_Click(object sender, EventArgs e)
109         {
110             if (MessageBox.Show("La␣liste␣sera␣supprim e.\n tes-vous␣↩
                    s r(e)?", "Attention", MessageBoxButton.YesNo, ↩
                    MessageBoxImage.Warning) == MessageBoxResult.Yes)
111             {
112                 Deleted = true;
113                 DialogResult = true;
114                 Close();
115             }
116         }
117     }
118 }
```

Listing 11 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistEdit.xaml.cs

2.9 ChecklistItemCreate.xaml

```
1  <!--
2   * Author    :    Ga l Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ChecklistItemCreate.xaml
5   * Desc.     :    This file contains the basic template of the ↩
        ChecklistItemCreate view
6       -->
7  <Window x:Class="Scrum_o_wall.Views.ChecklistItemCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation␣d'un␣objet␣de␣liste" Height="189.532" Width="558.533" ↩
               WindowStartupLocation="CenterScreen" FontSize="16">
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="161"></ColumnDefinition>
18             <ColumnDefinition></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition Height="47*"></RowDefinition>
22             <RowDefinition Height="50*"></RowDefinition>
23             <RowDefinition Height="62*"></RowDefinition>
24         </Grid.RowDefinitions>
25         <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
               VerticalAlignment="Center" Margin="10">Cr ation d'un objet de ↩
               liste</TextBlock>
```

```
26          <TextBlock Grid.Row="1" TextAlignment="Right" ↩
                VerticalAlignment="Center" Margin="10" >Objet :</TextBlock>
27          <TextBox Grid.Row="1" Grid.Column="1" MaxLength="255" Margin="10" ↩
                Name="tbxObjet" Stylus.IsPressAndHoldEnabled="False"></TextBox>
28          <Button x:Name="btnCancel" Grid.Row="2" Grid.Column="0" Margin="10" ↩
                Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Annuler</Button>
29          <Button x:Name="btnConfirm" Grid.Row="2" Grid.Column="1" Margin="10" ↩
                Click="BtnConfirm_Click" TouchUp="BtnConfirm_Click">Confirmer</Button>
30      </Grid>
31  </Window>
```

Listing 12 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistItemCreate.xaml

2.10 ChecklistItemCreate.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    ChecklistItemCreate.xaml.cs
5    * Desc.     :    This file contains the logic in the ChecklistItemCreate view
6    */
7   using System;
8   using System.Windows;
9
10  namespace Scrum_o_wall.Views
11  {
12      /// <summary>
13      /// Logique d'interaction pour ChecklistItemCreate.xaml
14      /// </summary>
15      public partial class ChecklistItemCreate : Window
16      {
17          public ChecklistItemCreate()
18          {
19              InitializeComponent();
20          }
21
22          private void BtnCancel_Click(object sender, EventArgs e)
23          {
24
25              Close();
26          }
27          private void BtnConfirm_Click(object sender, EventArgs e)
28          {
29              if (tbxObjet.Text.Trim().Length > 0)
30              {
31                  DialogResult = true;
32                  Close();
33              }
34              else
35              {
36                  MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
37              }
38          }
39      }
40  }
```

Listing 13 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistItemCreate.xaml.cs

2.11 ChecklistItemEdit.xaml

```
1   <!--
2    * Author    :    Ga l Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    ChecklistItemEdit.xaml
5    * Desc.     :    This file contains the basic template of the ChecklistItemEdit ↩
         view
6        -->
7   <Window x:Class="Scrum_o_wall.Views.ChecklistItemEdit"
8           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
```

```
13          mc:Ignorable="d"
14          Title="Modification d'un objet de liste" Height="270.407" Width="515.033" WindowStartupLocation="CenterScreen" FontSize="16">
15      <Grid>
16          <Grid.ColumnDefinitions>
17              <ColumnDefinition Width="161"></ColumnDefinition>
18              <ColumnDefinition></ColumnDefinition>
19          </Grid.ColumnDefinitions>
20          <Grid.RowDefinitions>
21              <RowDefinition/>
22              <RowDefinition/>
23              <RowDefinition/>
24              <RowDefinition/>
25              <RowDefinition/>
26          </Grid.RowDefinitions>
27          <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" VerticalAlignment="Center">Modification d'un objet de liste</TextBlock>
28          <TextBlock Grid.Row="1" TextAlignment="Right" VerticalAlignment="Center" Margin="10" Height="20" >Objet :</TextBlock>
29          <TextBox Grid.Row="1" Grid.Column="1" MaxLength="255" Margin="10" Name="tbxObjet" Stylus.IsPressAndHoldEnabled="False"></TextBox>
30
31          <CheckBox Grid.Row="2" Grid.ColumnSpan="2" HorizontalAlignment="Center" VerticalAlignment="Center" Name="chkbxDone" Stylus.IsPressAndHoldEnabled="False">Accompli</CheckBox>
32
33          <Button Click="BtnAssignedUsers_Click" TouchUp="BtnAssignedUsers_Click" Margin="10" Name="btnAssignedUsers" Grid.Row="3" Grid.ColumnSpan="2">Utilisateurs assign s</Button>
34
35          <Grid Grid.Row="100" Grid.ColumnSpan="2">
36              <Grid.ColumnDefinitions>
37                  <ColumnDefinition></ColumnDefinition>
38                  <ColumnDefinition></ColumnDefinition>
39                  <ColumnDefinition Width="2*"></ColumnDefinition>
40              </Grid.ColumnDefinitions>
41              <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Annuler</Button>
42              <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" Click="BtnDelete_Click" TouchUp="BtnDelete_Click">Supprimer</Button>
43              <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" Click="BtnConfirm_Click" TouchUp="BtnConfirm_Click">Confirmer</Button>
44          </Grid>
45      </Grid>
46  </Window>
```

Listing 14 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistItemEdit.xaml

2.12 ChecklistItemEdit.xaml.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ChecklistItemEdit.xaml.cs
5   * Desc.     :    This file contains the logic in the ChecklistItemEdit view
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Windows;
10
11 namespace Scrum_o_wall.Views
12 {
13     /// <summary>
14     /// Logique d'interaction pour ChecklistItemEdit.xaml
15     /// </summary>
16     public partial class ChecklistItemEdit : Window
17     {
18         private readonly ChecklistItem checklistItem;
19         private readonly Controller controller;
20         private readonly UserStory userStory;
21         public bool Deleted = false;
22
23         public ChecklistItemEdit(ChecklistItem aChecklistItem, UserStory aUserStory, Controller aController)
24         {
```

```
25              checklistItem = aChecklistItem;
26              controller = aController;
27              userStory = aUserStory;
28
29              InitializeComponent();
30
31              tbxObjet.Text = checklistItem.NameItem;
32              chkbxDone.IsChecked = checklistItem.Done;
33          }
34
35          private void BtnCancel_Click(object sender, EventArgs e)
36          {
37
38              Close();
39          }
40          private void BtnConfirm_Click(object sender, EventArgs e)
41          {
42              if (tbxObjet.Text.Trim().Length > 0)
43              {
44                  DialogResult = true;
45                  Close();
46              }
47              else
48              {
49                  MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
50              }
51          }
52          private void BtnDelete_Click(object sender, EventArgs e)
53          {
54              if (MessageBox.Show("L'objet de la liste sera ↩
                    supprim .\n tes -vous s r(e)?", "Attention", ↩
                    MessageBoxButton.YesNo, MessageBoxImage.Warning) == ↩
                    MessageBoxResult.Yes)
55              {
56                  DialogResult = true;
57                  Deleted = true;
58                  Close();
59              }
60          }
61          private void BtnAssignedUsers_Click(object sender, EventArgs e)
62          {
63              UserMenu userMenu = new UserMenu(checklistItem, ↩
                    userStory.GetUsers(), controller);
64              userMenu.ShowDialog();
65          }
66
67      }
68  }
```

Listing 15 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistItemEdit.xaml.cs

2.13 ChecklistMenu.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ChecklistMenu.xaml
5   * Desc.     :    This file contains the basic template of the ChecklistMenu view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.ChecklistMenu"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14        Title="Gestion des listes" Height="450" Width="447.5" ↩
             WindowStartupLocation="CenterScreen" FontSize="16">
15      <Grid>
16          <Grid.RowDefinitions>
17              <RowDefinition Height="70"></RowDefinition>
18              <RowDefinition/>
19              <RowDefinition Height="50"></RowDefinition>
20          </Grid.RowDefinitions>
21          <Grid.ColumnDefinitions>
22              <ColumnDefinition></ColumnDefinition>
```

```
23              <ColumnDefinition></ColumnDefinition>
24          </Grid.ColumnDefinitions>
25          <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" ↩
                Margin="10,16" TextAlignment="Center" FontSize="28" ↩
                Height="38">Gestion des listes</TextBlock>
26          <Border Grid.Row="1" Grid.ColumnSpan="2" Margin="10" ↩
                BorderThickness="1" BorderBrush="{DynamicResource␣{x:Static␣↩
                SystemColors.GrayTextBrushKey}}">
27              <ListView x:Name="lstLists" TouchUp="lstLists_TouchUp" ↩
                    MouseDoubleClick="lstLists_TouchUp">
28              </ListView>
29          </Border>
30
31          <Button x:Name="btnCancel" Grid.Row="2" Margin="10" ↩
                Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Retour</Button>
32          <Button x:Name="btnAddList" Grid.Row="2" Grid.Column="1" Margin="10" ↩
                Click="BtnAddList_Click" TouchUp="BtnCancel_Click">Ajouter</Button>
33      </Grid>
34  </Window>
```

Listing 16 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistMenu.xaml

2.14 ChecklistMenu.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    ChecklistMenu.xaml.cs
5    * Desc.     :    This file contains the logic in the ChecklistMenu view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Collections.Generic;
10  using System.Windows;
11  using System.Windows.Controls;
12  using System.Windows.Media;
13
14  namespace Scrum_o_wall.Views
15  {
16      /// <summary>
17      /// Logique d'interaction pour ChecklistMenu.xaml
18      /// </summary>
19      public partial class ChecklistMenu : Window
20      {
21          private readonly UserStory userStory;
22          private readonly Controller controller;
23          public ChecklistMenu(UserStory aUserStory, Controller aController)
24          {
25              controller = aController;
26              userStory = aUserStory;
27
28              InitializeComponent();
29
30              Refresh();
31          }
32
33          private void Refresh()
34          {
35              lstLists.Items.Clear();
36              foreach (Checklist chckLst in userStory.Checklists)
37              {
38                  ColumnDefinition col1 = new ColumnDefinition
39                  {
40                      Width = new GridLength(30, GridUnitType.Star)
41                  };
42                  ColumnDefinition col2 = new ColumnDefinition
43                  {
44                      Width = new GridLength(67, GridUnitType.Star)
45                  };
46
47                  //Create border
48                  Border border = new Border
49                  {
50                      BorderBrush = Brushes.Black,
51                      BorderThickness = new Thickness(1),
52                      Width = 390,
53                      Tag = chckLst
```

```
54                  };
55
56                  //Create grid
57                  Grid grd = new Grid
58                  {
59                      Name = "lst" + chckLst.Id.ToString()
60                  };
61                  grd.ColumnDefinitions.Add(col1);
62                  grd.ColumnDefinitions.Add(col2);
63                  grd.RowDefinitions.Add(new RowDefinition());
64                  grd.RowDefinitions.Add(new RowDefinition());
65
66                  //Create element for name
67                  TextBlock textBlock = new TextBlock
68                  {
69                      VerticalAlignment = VerticalAlignment.Top,
70                      TextWrapping = TextWrapping.Wrap,
71                      Margin = new Thickness(10),
72                      TextAlignment = TextAlignment.Right,
73                      Text = chckLst.Name
74                  };
75
76                  //Create element for list
77                  ListView lstView = new ListView
78                  {
79                      VerticalAlignment = VerticalAlignment.Top,
80                      Margin = new Thickness(10),
81                      Height = 26 * chckLst.ChecklistItems.Count
82                  };
83                  foreach (ChecklistItem item in chckLst.ChecklistItems)
84                  {
85                      CheckBox checkBox = new CheckBox
86                      {
87                          IsChecked = item.Done,
88                          Content = item.NameItem,
89                          Tag = item
90                      };
91                      checkBox.Checked += ChecklistItem_Checked;
92                      lstView.Items.Add(checkBox);
93                  }
94
95                  grd.Children.Add(textBlock);
96                  grd.Children.Add(lstView);
97                  Grid.SetRowSpan(lstView, 2);
98                  Grid.SetColumn(lstView, 1);
99                  border.Child = grd;
100
101                 lstLists.Items.Add(border);
102             }
103         }
104
105         private void ChecklistItem_Checked(object sender, RoutedEventArgs e)
106         {
107             ChecklistItem item = (sender as CheckBox).Tag as ChecklistItem;
108             item.Done = (sender as CheckBox).IsChecked == true;
109             controller.UpdateCheckListItem(item.NameItem, item.Done, item);
110         }
111         private void BtnCancel_Click(object sender, EventArgs e)
112         {
113
114             Close();
115         }
116         private void BtnAddList_Click(object sender, EventArgs e)
117         {
118             ChecklistCreate checklistCreate = new ChecklistCreate();
119             if (checklistCreate.ShowDialog() == true)
120             {
121                 string name = checklistCreate.tbxName.Text.Trim();
122                 Checklist checklist = controller.CreateCheckList(name, userStory);
123                 foreach (ChecklistItem item in checklistCreate.itemsToAdd)
124                 {
125                     controller.CreateCheckListItem(item.NameItem, checklist);
126                 }
127                 Refresh();
128             }
129         }
130
131         private void lstLists_TouchUp(object sender, EventArgs e)
132         {
```

```
133             if(lstLists.SelectedItem != null)
134             {
135                 Checklist checklist = (lstLists.SelectedItem as Border).Tag as ↩
                        Checklist;
136                 ChecklistEdit checklistEdit = new ChecklistEdit(checklist, ↩
                        userStory, controller);
137                 if (checklistEdit.ShowDialog() == true)
138                 {
139                     if (checklistEdit.Deleted)
140                     {
141                         controller.Delete(checklist);
142                     }
143                     else
144                     {
145                         List<ChecklistItem> items = new List<ChecklistItem>();
146                         foreach (object item in checklistEdit.listItems.Items)
147                         {
148                             items.Add(item as ChecklistItem);
149                         }
150                         string name = checklistEdit.tbxName.Text.Trim();
151                         controller.UpdateCheckList(name, items, checklist);
152                     }
153                     Refresh();
154                 }
155             }
156         }
157 
158     }
159 }
```

Listing 17 – ../../Scrum'o'wall/Scrum'o'wall/Views/ChecklistMenu.xaml.cs

2.15 CommentCreate.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    CommentCreate.xaml
5   * Desc.     :    This file contains the basic template of the CommentCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.CommentCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation␣de␣commentaire" Height="310.865" Width="445.866" ↩
               WindowStartupLocation="CenterScreen" FontSize="16">
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="25*"></ColumnDefinition>
18             <ColumnDefinition Width="48*"></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition Height="63*"></RowDefinition>
22             <RowDefinition Height="117*"></RowDefinition>
23             <RowDefinition Height="50*"></RowDefinition>
24             <RowDefinition Height="50"></RowDefinition>
25         </Grid.RowDefinitions>
26         <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
               VerticalAlignment="Center" Margin="10">Poster un ↩
               Commentaire</TextBlock>
27         <TextBlock Grid.Row="1" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10,45,10,50" Height="22" >Contenu ↩
               :</TextBlock>
28         <TextBox x:Name="tbxContent" MaxLength="65535" Grid.Row="1" ↩
               Grid.Column="1" TextWrapping="WrapWithOverflow" Margin="10" ↩
               Stylus.IsPressAndHoldEnabled="False"></TextBox>
29         <TextBlock Grid.Row="2" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10,12,10,18" Height="20" >Auteur ↩
               :</TextBlock>
30         <ComboBox Name="cbxAuthor" Grid.Row="2" Grid.Column="1" Margin="10" ↩
               Stylus.IsPressAndHoldEnabled="False">
31         </ComboBox>
32         <Button x:Name="btnCancel" Grid.Row="3" Grid.Column="0" Margin="10" ↩
               Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Annuler</Button>
```

```
33          <Button x:Name="btnConfirm" Grid.Row="3" Grid.Column="1" Margin="10" ↩
                Click="BtnConfirm_Click" TouchUp="BtnConfirm_Click">Confirmer</Button>
34      </Grid>
35  </Window>
```

Listing 18 – ../../Scrum'o'wall/Scrum'o'wall/Views/CommentCreate.xaml

2.16 CommentCreate.xaml.cs

```
1  /*
2   * Author    :    Ga l  Serge  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    CommentCreate.xaml.cs
5   * Desc.     :    This file contains the logic in the CommentCreate view
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Collections.Generic;
10 using System.Windows;
11
12 namespace Scrum_o_wall.Views
13 {
14     /// <summary>
15     /// Logique d'interaction pour CommentCreate.xaml
16     /// </summary>
17     public partial class CommentCreate : Window
18     {
19         public CommentCreate(List<User> assignedUsers)
20         {
21             InitializeComponent();
22             foreach (User user in assignedUsers)
23             {
24                 cbxAuthor.Items.Add(user);
25             }
26         }
27
28         private void BtnCancel_Click(object sender, EventArgs e)
29         {
30
31             Close();
32         }
33         private void BtnConfirm_Click(object sender, EventArgs e)
34         {
35             if (cbxAuthor.SelectedItem != null && tbxContent.Text.Trim().Length ↩
                   > 0)
36             {
37                 DialogResult = true;
38                 Close();
39             }
40             else
41             {
42                 MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                       "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
43             }
44         }
45
46     }
47 }
```

Listing 19 – ../../Scrum'o'wall/Scrum'o'wall/Views/CommentCreate.xaml.cs

2.17 CommentMenu.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    CommentMenu.xaml
5   * Desc.     :    This file contains the basic template of the CommentMenu view
6       -->
7  <Window x:Class="Scrum_o_wall.Views.CommentMenu"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
13          mc:Ignorable="d"
14         Title="Gestion des commentaires" Height="450" Width="447.5" ↩
               WindowStartupLocation="CenterScreen" FontSize="16">
15      <Grid>
16          <Grid.RowDefinitions>
17              <RowDefinition Height="70"></RowDefinition>
18              <RowDefinition/>
19              <RowDefinition Height="50"></RowDefinition>
20          </Grid.RowDefinitions>
21          <Grid.ColumnDefinitions>
22              <ColumnDefinition></ColumnDefinition>
23              <ColumnDefinition></ColumnDefinition>
24          </Grid.ColumnDefinitions>
25          <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" ↩
                Margin="10,16" TextAlignment="Center" FontSize="28" ↩
                Height="38">Gestion des commentaires</TextBlock>
26          <Border Grid.Row="1" Grid.ColumnSpan="2" Margin="10" ↩
                BorderThickness="1" BorderBrush="{DynamicResource {x:Static ↩
                SystemColors.GrayTextBrushKey}}">
27              <ListView x:Name="lstComments" ↩
                    MouseDoubleClick="LstComments_MouseDoubleClick" ↩
                    TouchUp="LstComments_MouseDoubleClick">
28              </ListView>
29          </Border>
30
31          <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" Grid.Row="2" ↩
                Margin="10" Click="BtnCancel_Click">Retour</Button>
32          <Button TouchUp="BtnAddComment_Click" x:Name="btnAddComment" ↩
                Grid.Row="2" Grid.Column="1" Margin="10" ↩
                Click="BtnAddComment_Click">Ajouter</Button>
33      </Grid>
34  </Window>
```

Listing 20 – ../../Scrum'o'wall/Scrum'o'wall/Views/CommentMenu.xaml

2.18 CommentMenu.xaml.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    CommentMenu.xaml.cs
5   * Desc.     :    This file contains the logic in the CommentMenu view
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Windows;
10 using System.Windows.Controls;
11
12 namespace Scrum_o_wall.Views
13 {
14     /// <summary>
15     /// Logique d'interaction pour CommentMenu.xaml
16     /// </summary>
17     public partial class CommentMenu : Window
18     {
19         private readonly UserStory userStory;
20         private readonly Controller controller;
21         public CommentMenu(UserStory aUserStory, Controller aController)
22         {
23             userStory = aUserStory;
24             controller = aController;
25
26             InitializeComponent();
27
28             Refresh();
29         }
30
31         public void Refresh()
32         {
33             lstComments.Items.Clear();
34             foreach (Comment comment in userStory.Comments)
35             {
36                 lstComments.Items.Add(comment);
37             }
38         }
39
```

```
        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnAddComment_Click(object sender, EventArgs e)
        {
            CommentCreate commentCreate = new CommentCreate(userStory.GetUsers());
            if (userStory.GetUsers().Count == 0)
            {
                MessageBox.Show("Aucun utilisateur assign ", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
            else if (commentCreate.ShowDialog() == true)
            {
                string content = commentCreate.tbxContent.Text.Trim();
                User user = commentCreate.cbxAuthor.SelectedItem as User;
                controller.CreateComment(content, user, userStory);
                Refresh();
            }
        }

        private void LstComments_MouseDoubleClick(object sender, EventArgs e)
        {
            ListBox lbx = sender as ListBox;
            if (lbx.SelectedItem != null)
            {
                Comment comment = lbx.SelectedItem as Comment;
                MessageBox.Show(string.Format("Auteur : {0}\nDate : {1}\n{2}", comment.User, comment.DateTime, comment.Description), "Contenu du commentaire", MessageBoxButton.OK);
            }
        }
    }
}
```

Listing 21 – ../../Scrum'o'wall/Scrum'o'wall/Views/CommentMenu.xaml.cs

2.19 FileCreate.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    FileCreate.xaml
 * Desc.     :    This file contains the basic template of the FileCreate view
    -->
<Window x:Class="Scrum_o_wall.Views.FileCreate"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Ajouter un fichier" Height="500" Width="600" WindowStartupLocation="CenterScreen" FontSize="16">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="78*"></RowDefinition>
            <RowDefinition Height="56*"></RowDefinition>
            <RowDefinition Height="227*"></RowDefinition>
            <RowDefinition Height="56*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="140"></ColumnDefinition>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" Margin="10,26" TextAlignment="Center" FontSize="20" Height="26" >Ajouter un fichier</TextBlock>
        <TextBlock Grid.Row="1" VerticalAlignment="Center" Margin="10,14,10,22" TextAlignment="Right" Height="20" ><Run Text="Fichier :"/><LineBreak/><Run/><LineBreak/><Run/></TextBlock>
        <TextBlock Grid.Row="2" VerticalAlignment="Top" Margin="10,10,10,0" TextAlignment="Right" Height="21" >Description :</TextBlock>
        <TextBox x:Name="tbxFileName" MaxLength="65535" IsEnabled="False" Grid.Row="1" Grid.Column="1" Margin="10,10,100,10" ></TextBox>
```

```
30          <Button x:Name="btnFileSearch" TouchUp="BtnFileSearch_Click" ↩
                Grid.Row="1" Grid.Column="1" Margin="0,10,10,10" ↩
                Click="BtnFileSearch_Click" HorizontalAlignment="Right" ↩
                Width="87">+</Button>
31          <TextBox Margin="10" MaxLength="65535" TextWrapping="Wrap" ↩
                Name="tbxDescription" Grid.Row="2" Grid.Column="1" ↩
                Stylus.IsPressAndHoldEnabled="False">
32          </TextBox>
33
34          <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" Grid.Row="4" ↩
                Margin="10" Click="BtnCancel_Click">Annuler</Button>
35          <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" Grid.Row="4" ↩
                Grid.Column="1" Margin="10" Click="BtnConfirm_Click">Confirmer</Button>
36      </Grid>
37  </Window>
```

Listing 22 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileCreate.xaml

2.20 FileCreate.xaml.cs

```
1   /*
2    * Author   :    Ga l Serge Mariot
3    * Project  :    Scrum'o'wall
4    * File     :    FileCreate.xaml.cs
5    * Desc.    :    This file contains the logic in the FileCreate view
6    */
7   using Microsoft.Win32;
8   using System;
9   using System.Windows;
10
11  namespace Scrum_o_wall.Views
12  {
13      /// <summary>
14      /// Logique d'interaction pour FileCreate.xaml
15      /// </summary>
16      public partial class FileCreate : Window
17      {
18          public FileCreate()
19          {
20              InitializeComponent();
21          }
22
23          private void BtnFileSearch_Click(object sender, EventArgs e)
24          {
25              OpenFileDialog opf = new OpenFileDialog
26              {
27                  Multiselect = false
28              };
29              if (opf.ShowDialog() == true)
30              {
31                  tbxFileName.Text = opf.FileName;
32              }
33          }
34          private void BtnCancel_Click(object sender, EventArgs e)
35          {
36
37              Close();
38          }
39          private void BtnConfirm_Click(object sender, EventArgs e)
40          {
41              int descriptionLength = tbxDescription.Text.Trim().Length;
42              int fileNameLength = tbxFileName.Text.Trim().Length;
43              if (descriptionLength > 0 && fileNameLength > 0 && ↩
                    System.IO.File.Exists(tbxFileName.Text))
44              {
45                  DialogResult = true;
46                  Close();
47              }
48              else
49              {
50                  MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
51              }
52          }
53      }
54  }
```

Listing 23 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileCreate.xaml.cs

2.21 FileEdit.xaml

```
<!--
 * Author   :    Ga l  Mariot
 * Project  :    Scrum'o'wall
 * File     :    FileEdit.xaml
 * Desc.    :    This file contains the basic template of the FileEdit view
   -->
<Window x:Class="Scrum_o_wall.Views.FileEdit"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Modifier un fichier" Height="500" Width="600" WindowStartupLocation="CenterScreen" FontSize="16">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="70*"></RowDefinition>
            <RowDefinition Height="50*"></RowDefinition>
            <RowDefinition Height="208*"></RowDefinition>
            <RowDefinition Height="50*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="140"></ColumnDefinition>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" Margin="10" TextAlignment="Center" FontSize="20" >Modifier un fichier</TextBlock>
        <TextBlock Grid.Row="1" VerticalAlignment="Center" Margin="10" TextAlignment="Right">Fichier :</TextBlock>
        <TextBlock Grid.Row="2" VerticalAlignment="Top" Margin="10" TextAlignment="Right" >Description :</TextBlock>
        <TextBox x:Name="tbxFileName" MaxLength="65535" IsEnabled="False" Grid.Row="1" Grid.Column="1" Margin="10" ></TextBox>
        <TextBox Margin="10" MaxLength="65535" TextWrapping="Wrap" Name="tbxDescription" Grid.Row="2" Grid.Column="1" Stylus.IsPressAndHoldEnabled="False"/>


        <Grid Grid.Row="4" Grid.ColumnSpan="3">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*"></ColumnDefinition>
                <ColumnDefinition Width="1*"></ColumnDefinition>
                <ColumnDefinition Width="3*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Button TouchUp="BtnCancel_Click" x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click">Annuler</Button>
            <Button TouchUp="BtnDelete_Click" x:Name="btnDelete" Grid.Column="1" Margin="10" Click="BtnDelete_Click">Supprimer</Button>
            <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" Grid.Column="2" Margin="10" Click="BtnConfirm_Click">Confirmer</Button>
        </Grid>
    </Grid>
</Window>
```

Listing 24 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileEdit.xaml

2.22 FileEdit.xaml.cs

```
/*
 * Author   :    Ga l  Serge Mariot
 * Project  :    Scrum'o'wall
 * File     :    FileEdit.xaml.cs
 * Desc.    :    This file contains the logic in the FileEdit view
 */
using Scrum_o_wall.Classes;
```

```
8  using System;
9  using System.Windows;
10
11 namespace Scrum_o_wall.Views
12 {
13     /// <summary>
14     /// Logique d'interaction pour FileEdit.xaml
15     /// </summary>
16     public partial class FileEdit : Window
17     {
18         private readonly File file;
19         public bool Deleted = false;
20
21
22         public FileEdit(File aFile)
23         {
24             file = aFile;
25
26             InitializeComponent();
27
28             tbxDescription.Text = file.Description;
29             tbxFileName.Text = file.Name;
30         }
31
32         private void BtnCancel_Click(object sender, EventArgs e)
33         {
34
35             Close();
36         }
37         private void BtnConfirm_Click(object sender, EventArgs e)
38         {
39             int descriptionLength = tbxDescription.Text.Trim().Length;
40             if (descriptionLength > 0)
41             {
42                 DialogResult = true;
43                 Close();
44             }
45             else
46             {
47                 MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", ↩
                       "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
48             }
49         }
50
51         private void BtnDelete_Click(object sender, EventArgs e)
52         {
53             if (MessageBox.Show("Le fichier sera supprim .\n tes -vous ↩
                   s r(e)?", "Attention", MessageBoxButton.YesNo, ↩
                   MessageBoxImage.Warning) == MessageBoxResult.Yes)
54             {
55                 DialogResult = true;
56                 Deleted = true;
57                 Close();
58             }
59         }
60     }
61 }
```

Listing 25 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileEdit.xaml.cs

2.23 FileMenu.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    FileMenu.xaml
5   * Desc.     :    This file contains the basic template of the FileMenu view
6       -->
7  <Window x:Class="Scrum_o_wall.Views.FileMenu"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14        Title="Gestion des fichiers" Height="450" Width="447.5" ↩
              WindowStartupLocation="CenterScreen" FontSize="16">
```

```
15      <Grid>
16          <Grid.RowDefinitions>
17              <RowDefinition Height="70"></RowDefinition>
18              <RowDefinition/>
19              <RowDefinition Height="50"></RowDefinition>
20          </Grid.RowDefinitions>
21          <Grid.ColumnDefinitions>
22              <ColumnDefinition></ColumnDefinition>
23              <ColumnDefinition></ColumnDefinition>
24          </Grid.ColumnDefinitions>
25          <TextBlock Grid.ColumnSpan="2" VerticalAlignment="Center" ↩
                Margin="10,16" TextAlignment="Center" FontSize="28" ↩
                Height="38">Gestion des fichiers</TextBlock>
26          <Border Grid.Row="1" Grid.ColumnSpan="2" Margin="10" ↩
                BorderThickness="1" BorderBrush="{DynamicResource␣{x:Static␣↩
                SystemColors.GrayTextBrushKey}}">
27              <ListView x:Name="lstFiles" TouchUp="LstFiles_MouseDoubleClick" ↩
                    MouseDoubleClick="LstFiles_MouseDoubleClick" ↩
                    Stylus.IsPressAndHoldEnabled="False">
28              </ListView>
29          </Border>
30
31          <Button TouchUp="Quit_Click" x:Name="btnCancel" Grid.Row="2" ↩
                Margin="10" Click="Quit_Click">Retour</Button>
32          <Button TouchUp="BtnAddFile_Click" x:Name="btnAddFile" Grid.Row="2" ↩
                Grid.Column="1" Margin="10" Click="BtnAddFile_Click">Ajouter</Button>
33      </Grid>
34  </Window>
```

Listing 26 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileMenu.xaml

2.24 FileMenu.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    FileMenu.xaml.cs
5    * Desc.     :    This file contains the logic in the FileMenu view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Windows;
10  using System.Windows.Controls;
11  using System.Windows.Media;
12
13  namespace Scrum_o_wall.Views
14  {
15      /// <summary>
16      /// Logique d'interaction pour FileMenu.xaml
17      /// </summary>
18      public partial class FileMenu : Window
19      {
20          private readonly UserStory userStory;
21          private readonly Controller controller;
22          public FileMenu(UserStory aUserStory, Controller aController)
23          {
24              userStory = aUserStory;
25              controller = aController;
26
27
28              InitializeComponent();
29
30              Refresh();
31          }
32
33          public void Refresh()
34          {
35              lstFiles.Items.Clear();
36              foreach (File file in userStory.Files)
37              {
38                  //Create border
39                  Border border = new Border
40                  {
41                      BorderBrush = Brushes.Black,
42                      BorderThickness = new Thickness(1),
43                      Width = 408,
44                      Tag = file
```

```
45                  };
46                  border.TouchDown += FileInList_Click;
47                  border.MouseLeftButtonDown += FileInList_Click;
48
49                  //Create grid
50                  Grid grd = new Grid
51                  {
52                      Name = "lst" + file.Id.ToString()
53                  };
54                  grd.RowDefinitions.Add(new RowDefinition());
55                  grd.RowDefinitions.Add(new RowDefinition());
56
57                  //Create element for name
58                  TextBlock textBlock = new TextBlock
59                  {
60                      VerticalAlignment = VerticalAlignment.Top,
61                      TextWrapping = TextWrapping.Wrap,
62                      Margin = new Thickness(10),
63                      TextAlignment = TextAlignment.Right,
64                      Text = file.Name
65                  };
66
67                  //Create element for name
68                  TextBlock descBlock = new TextBlock
69                  {
70                      VerticalAlignment = VerticalAlignment.Top,
71                      TextWrapping = TextWrapping.Wrap,
72                      Margin = new Thickness(10),
73                      TextAlignment = TextAlignment.Right,
74                      Text = file.Name
75                  };
76
77                  grd.Children.Add(textBlock);
78                  grd.Children.Add(descBlock);
79                  Grid.SetColumn(descBlock, 1);
80                  border.Child = grd;
81
82                  lstFiles.Items.Add(file);
83              }
84          }
85
86          private void FileInList_Click(object sender, EventArgs e)
87          {
88              File file = (sender as Border).Tag as File;
89              FileEdit fileEdit = new FileEdit(file);
90              if (fileEdit.ShowDialog() == true)
91              {
92                  if (fileEdit.Deleted)
93                  {
94                      controller.Delete(file);
95                  }
96                  else
97                  {
98                      controller.UpdateFile(fileEdit.tbxDescription.Text.Trim(), ↩
                            file);
99                  }
100                 Refresh();
101             }
102             if (fileEdit.ShowDialog() == true)
103             {
104                 controller.UpdateFile(fileEdit.tbxDescription.Text.Trim(), file);
105             }
106             Refresh();
107         }
108         private void Quit_Click(object sender, EventArgs e)
109         {
110
111             Close();
112         }
113         private void BtnAddFile_Click(object sender, EventArgs e)
114         {
115             FileCreate fileCreate = new FileCreate();
116             if (fileCreate.ShowDialog() == true)
117             {
118                 string fileName = fileCreate.tbxFileName.Text.Trim();
119                 string description = fileCreate.tbxDescription.Text.Trim();
120                 controller.CreateFile(fileName, description, userStory);
121                 Refresh();
122             }
```

```
123
124         }
125
126         private void LstFiles_MouseDoubleClick(object sender, EventArgs e)
127         {
128             ListBox lbx = sender as ListBox;
129             if (lbx.SelectedItem != null)
130             {
131                 File file = lbx.SelectedItem as File;
132                 FileEdit fileEdit = new FileEdit(file);
133                 if (fileEdit.ShowDialog() == true)
134                 {
135                     if (fileEdit.Deleted)
136                     {
137                         controller.Delete(file);
138                     }
139                     else
140                     {
141                         controller.UpdateFile(fileEdit.tbxDescription.Text, file);
142                     }
143                     Refresh();
144                 }
145             }
146         }
147     }
148 }
```

Listing 27 – ../../Scrum'o'wall/Scrum'o'wall/Views/FileMenu.xaml.cs

2.25 MainMenu.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ActivitiesMenu.xaml
5   * Desc.     :    This file contains the basic template of the ActivitiesMenu view
6      -->
7  <Window x:Class="Scrum_o_wall.MainMenu"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall"
13         mc:Ignorable="d"
14         Title="Scrum'o'Wall" ResizeMode="NoResize" WindowState="Maximized" ↩
               WindowStartupLocation="CenterScreen" Width="1000" Height="1000" ↩
               WindowStyle="None" FontSize="16">
15     <Window.ContextMenu >
16         <ContextMenu >
17             <MenuItem Header="Ajouter␣un␣projet" Name="AddProject" ↩
                   Click="AddProject_Click"/>
18             <Separator/>
19             <MenuItem Header="Quitter" Name="Quit" Click="Quit_Click"/>
20         </ContextMenu>
21     </Window.ContextMenu>
22     <Grid>
23         <Grid.RowDefinitions >
24             <RowDefinition />
25             <RowDefinition Height="100"/>
26         </Grid.RowDefinitions>
27         <Grid.ColumnDefinitions>
28             <ColumnDefinition></ColumnDefinition>
29             <ColumnDefinition></ColumnDefinition>
30         </Grid.ColumnDefinitions>
31         <ScrollViewer x:Name="scrllVwr" VerticalScrollBarVisibility="Auto" ↩
               Grid.ColumnSpan="2">
32         <Canvas x:Name="cnvsProject" Width="1000" Margin="0,0,0,0" ↩
               HorizontalAlignment="Left" Height="796">
33         </Canvas>
34     </ScrollViewer>
35         <Button Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" ↩
               x:Name="btnAddProject" TouchUp="AddProject_Click" ↩
               Click="AddProject_Click" Content="+" VerticalContentAlignment="Top"  ↩
               Width="75" Height="75" BorderBrush="Black" FontSize="45" Margin="10">
36         <Button.Resources>
37             <Style TargetType="Border">
38                 <Setter Property="CornerRadius" Value="50"/>
```

```
            </Style>
        </Button.Resources>
        </Button>
        <Button Grid.Row="1"  HorizontalAlignment="Right" x:Name="btnReturn" ↩
            TouchUp="Quit_Click" Click="Quit_Click" Content="â  " ↩
            VerticalContentAlignment="Top" BorderBrush="Black" FontSize="45"  ↩
            Width="75" Height="75" Margin="10">
            <Button.Resources>
                <Style TargetType="{x:Type␣Border}">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
    </Grid>
</Window>
```

Listing 28 – ../../Scrum'o'wall/Scrum'o'wall/Views/MainMenu.xaml

2.26 MainMenu.xaml.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   MainMenu.xaml.cs
 * Desc.    :   This file contains the logic in the MainMenu view
 */
using Scrum_o_wall.Classes;
using Scrum_o_wall.Views;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace Scrum_o_wall
{
    /// <summary>
    /// Logique d'interaction pour MainMenu.xaml
    /// </summary>
    public partial class MainMenu : Window
    {
        private readonly Controller controller;
        private readonly List<UserControl> projectControls = new ↩
            List<UserControl>();
        public MainMenu()
        {
            InitializeComponent();
            controller = new Controller();
            Loaded += MainMenu_Loaded;
        }

        private void Refresh()
        {
            //Create controls for the projects
            int maxProj = controller.Projects.Count;
            foreach (UserControl project in projectControls)
            {
                cnvsProject.Children.Remove(project);
            }
            projectControls.Clear();
            for (int i = 0; i < maxProj; i++)
            {
                Project p = controller.Projects[i];

                //create a control for title
                Label title = new Label
                {
                    Content = p.Name,
                    HorizontalContentAlignment = HorizontalAlignment.Center,
                    FontSize = 24
                };
                Grid.SetRow(title, 0);

                //Create a control for description
                Label desc = new Label();
                TextBlock txtBlck = new TextBlock
```

```
57                  {
58                      TextWrapping = TextWrapping.WrapWithOverflow,
59                      Text = p.Description
60                  };
61                  desc.Content = txtBlck;
62                  desc.HorizontalAlignment = HorizontalAlignment.Stretch;
63                  desc.FontSize = 18;
64                  Grid.SetRow(desc, 1);
65
66                  //Create a control of date
67                  Label date = new Label
68                  {
69                      Content = "Date␣de␣d but␣:␣" + p.Begin.ToShortDateString(),
70                      HorizontalContentAlignment = HorizontalAlignment.Left,
71                      FontSize = 18
72                  };
73                  Grid.SetRow(date, 2);
74
75                  //Place controls in a grid
76                  Grid grd = new Grid();
77                  grd.RowDefinitions.Add(new RowDefinition() { Height = new ↩
                        GridLength(60) });
78                  grd.RowDefinitions.Add(new RowDefinition());
79                  grd.RowDefinitions.Add(new RowDefinition() { Height = new ↩
                        GridLength(40) });
80                  grd.Children.Add(title);
81                  grd.Children.Add(desc);
82                  grd.Children.Add(date);
83
84                  //Create project frame
85                  UserControl usrCntrl = new UserControl
86                  {
87                      Width = cnvsProject.Width / 4,
88                      Height = cnvsProject.Height / 5,
89                      Content = grd,
90                      BorderBrush = Brushes.Black,
91                      Background = Brushes.LightGray,
92                      Cursor = Cursors.Hand,
93                      Tag = p
94                  };
95                  usrCntrl.TouchDown += UsrCntrl_Click;
96                  usrCntrl.MouseLeftButtonDown += UsrCntrl_Click;
97
98                  //Positioning of control
99                  Canvas.SetLeft(usrCntrl, (cnvsProject.Width - usrCntrl.Width) / ↩
                        2.0 + ((usrCntrl.Width + usrCntrl.Width / 4) * (i % 3 - 1)));
100                 Canvas.SetTop(usrCntrl, 20 + (usrCntrl.Height + usrCntrl.Height ↩
                        / 5) * (i / 3));
101
102                 if (Canvas.GetTop(usrCntrl) + usrCntrl.Height > ↩
                        cnvsProject.Height)
103                 {
104                     cnvsProject.Height = Canvas.GetTop(usrCntrl) + ↩
                            usrCntrl.Height;
105                 }
106
107                 //Add project frame to canvas
108                 cnvsProject.Children.Add(usrCntrl);
109                 projectControls.Add(usrCntrl);
110             }
111         }
112
113         private void MainMenu_Loaded(object sender, RoutedEventArgs e)
114         {
115             //ActualWidth and ActualHeight measured when window is loaded minus ↩
                    border sizes
116             cnvsProject.Width = ActualWidth;
117             cnvsProject.Height = ActualHeight;
118             scrllVwr.Width = cnvsProject.Width;
119             scrllVwr.Height = cnvsProject.Height;
120
121             //Refresh the view
122             Refresh();
123         }
124         private void UsrCntrl_Click(object sender, EventArgs e)
125         {
126             Project p = (sender as UserControl).Tag as Project;
127             ProjectMenu projectMenu = new ProjectMenu(p, controller);
128             projectMenu.ShowDialog();
```

```
129             Refresh();
130         }
131         private void Quit_Click(object sender, EventArgs e)
132         {
133             if (MessageBox.Show("Vous allez quitter l'application.\n tes-vous↩
                    s r de vouloir continuer ?", "Attention", ↩
                    MessageBoxButton.OKCancel, MessageBoxImage.Warning) == ↩
                    MessageBoxResult.OK)
134             {
135                 Application.Current.Shutdown();
136             }
137         }
138         private void AddProject_Click(object sender, EventArgs e)
139         {
140             ProjectCreate projectCreate = new ProjectCreate();
141             if (projectCreate.ShowDialog() == true)
142             {
143                 string name = projectCreate.tbxName.Text;
144                 string desc = projectCreate.tbxDesc.Text;
145                 DateTime date = (DateTime)projectCreate.tbxDate.SelectedDate;
146                 controller.CreateProject(name, desc, date);
147 
148                 Refresh();
149             }
150         }
151     }
152 }
```

Listing 29 – ../../Scrum'o'wall/Scrum'o'wall/Views/MainMenu.xaml.cs

2.27 MindmapCreate.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    MindmapCreate.xaml
5   * Desc.     :    This file contains the basic template of the MindmapCreate view
6       -->
7  <Window x:Class="Scrum_o_wall.Views.MindmapCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation d'un mindmap" Height="196.027" Width="538.14" ↩
               WindowStartupLocation="CenterScreen" FontSize="16">
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="161"></ColumnDefinition>
18             <ColumnDefinition></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition Height="47*"></RowDefinition>
22             <RowDefinition Height="50*"></RowDefinition>
23             <RowDefinition Height="62*"></RowDefinition>
24         </Grid.RowDefinitions>
25         <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
               VerticalAlignment="Center" Height="28" Margin="0,10,0,9">Cr ation ↩
               d'un mindmap</TextBlock>
26         <TextBlock Grid.Row="1" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10">Nom :</TextBlock>
27         <TextBox x:Name="tbxName" Grid.Row="1" MaxLength="100" Grid.Column="1" ↩
               Margin="10" Stylus.IsPressAndHoldEnabled="False"></TextBox>
28         <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" Grid.Row="2" ↩
               Grid.Column="0" Margin="10" Click="BtnCancel_Click">Annuler</Button>
29         <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" Grid.Row="2" ↩
               Grid.Column="1" Margin="10" Click="BtnConfirm_Click">Confirmer</Button>
30     </Grid>
31 </Window>
```

Listing 30 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapCreate.xaml

2.28 MindmapCreate.xaml.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   MindmapCreate.xaml.cs
 * Desc.    :   This file contains the logic in the MindmapCreate view
 */
using System;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour MindmapCreate.xaml
    /// </summary>
    public partial class MindmapCreate : Window
    {
        public MindmapCreate()
        {
            InitializeComponent();
        }

        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            if (tbxName.Text.Trim().Length > 0)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Listing 31 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapCreate.xaml.cs

2.29 MindmapEdit.xaml

```
<!--
 * Author   :   Ga l Mariot
 * Project  :   Scrum'o'wall
 * File     :   MindmapEdit.xaml
 * Desc.    :   This file contains the basic template of the MindmapEdit view
    -->
<Window x:Class="Scrum_o_wall.Views.MindmapEdit"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Modification d'un mindmap" Height="196.027" Width="538.14" WindowStartupLocation="CenterScreen" FontSize="16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="161"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="47*"></RowDefinition>
            <RowDefinition Height="50*"></RowDefinition>
            <RowDefinition Height="62*"></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" VerticalAlignment="Center" Height="28" Margin="0,10,0,9">Modification d'un mindmap</TextBlock>
        <TextBlock Grid.Row="1" TextAlignment="Right" VerticalAlignment="Center" Margin="10">Nom :</TextBlock>
```

```
27          <TextBox x:Name="tbxName" Grid.Row="1" MaxLength="100" Grid.Column="1" ↩
                Margin="10" Stylus.IsPressAndHoldEnabled="False"></TextBox>
28          <Grid Grid.Row="2" Grid.ColumnSpan="2">
29              <Grid.ColumnDefinitions>
30                  <ColumnDefinition></ColumnDefinition>
31                  <ColumnDefinition></ColumnDefinition>
32                  <ColumnDefinition Width="2*"></ColumnDefinition>
33              </Grid.ColumnDefinitions>
34              <Button TouchUp="BtnCancel_Click" x:Name="btnCancel"  ↩
                    Grid.Column="0" Margin="10" ↩
                    Click="BtnCancel_Click">Annuler</Button>
35              <Button TouchUp="BtnDelete_Click" x:Name="btnDelete"  ↩
                    Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click">Supprimer</Button>
36              <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm"  ↩
                    Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click">Confirmer</Button>
37          </Grid>
38      </Grid>
39  </Window>
```

Listing 32 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapEdit.xaml

2.30 MindmapEdit.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    MindmapEdit.xaml.cs
5    * Desc.     :    This file contains the logic in the MindmapEdit view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Windows;
10
11  namespace Scrum_o_wall.Views
12  {
13      /// <summary>
14      /// Logique d'interaction pour MindmapCreate.xaml
15      /// </summary>
16      public partial class MindmapEdit : Window
17      {
18          public bool Deleted = false;
19          private MindMap mindMap;
20          public MindmapEdit(MindMap aMindMap)
21          {
22              mindMap = aMindMap;
23              InitializeComponent();
24              tbxName.Text = mindMap.Name;
25          }
26
27          private void BtnCancel_Click(object sender, EventArgs e)
28          {
29
30              Close();
31          }
32          private void BtnDelete_Click(object sender, EventArgs e)
33          {
34              if (MessageBox.Show("Le␣mindmap␣sera␣supprim .\n tes-vous␣↩
                    s r(e)?", "Attention", MessageBoxButton.YesNo, ↩
                    MessageBoxImage.Warning) == MessageBoxResult.Yes)
35              {
36                  Deleted = true;
37                  DialogResult = true;
38                  Close();
39              }
40          }
41          private void BtnConfirm_Click(object sender, EventArgs e)
42          {
43              if (tbxName.Text.Trim().Length > 0)
44              {
45                  DialogResult = true;
46                  Close();
47              }
48              else
49              {
```

```
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", ↩
                    "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Listing 33 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapEdit.xaml.cs

2.31 MindmapMenu.xaml

```
<!--
 * Author   :   Ga l Mariot
 * Project  :   Scrum'o'wall
 * File     :   MindmapMenu.xaml
 * Desc.    :   This file contains the basic template of the MindmapMenu view
    -->
<Window x:Class="Scrum_o_wall.Views.MindmapMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Scrum'o'Wall" Height="450" Width="800" WindowState="Maximized" ↩
            WindowStyle="None" WindowStartupLocation="CenterScreen" ↩
            ResizeMode="NoResize" FontSize="16">
    <Window.ContextMenu >
        <ContextMenu >
            <MenuItem Header="Cr er un noeud" Name="CreateNode" ↩
                Click="CreateNode_Click"/>
            <MenuItem Header="Modifier le mindmap" Name="Modify" ↩
                Click="Modify_Click"/>
            <Separator/>
            <MenuItem Header="Retour" Name="Quit" Click="Quit_Click"/>
        </ContextMenu>
    </Window.ContextMenu>
    <Grid>
        <Grid.RowDefinitions >
            <RowDefinition />
            <RowDefinition Height="100"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <ScrollViewer x:Name="scrllVwr" VerticalScrollBarVisibility="Auto" ↩
            HorizontalScrollBarVisibility="Auto" Grid.ColumnSpan="2">
            <Grid x:Name="grdMindMap" >
            </Grid>
        </ScrollViewer>
        <Button Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" ↩
            x:Name="btnAddNode" TouchUp="CreateNode_Click" ↩
            Click="CreateNode_Click" Content="+" VerticalContentAlignment="Top"  ↩
            Width="75" Height="75" BorderBrush="Black" FontSize="45" Margin="10">
            <Button.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
        <Button Grid.Row="1"  HorizontalAlignment="Right" x:Name="btnReturn" ↩
            TouchUp="Quit_Click" Click="Quit_Click" Content="â  " ↩
            VerticalContentAlignment="Top" BorderBrush="Black" FontSize="45"  ↩
            Width="75" Height="75" Margin="10">
            <Button.Resources>
                <Style TargetType="{x:Type Border}">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
    </Grid>
</Window>
```

Listing 34 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapMenu.xaml

2.32 MindmapMenu.xaml.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    MindmapMenu.xaml.cs
 * Desc.     :    This file contains the logic in the MindmapMenu view
 */
using Scrum_o_wall.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour MindmapMenu.xaml
    /// </summary>
    public partial class MindmapMenu : Window
    {
        private readonly MindMap mindMap;
        private readonly Controller controller;
        private readonly List<UserControl> nodeControls = new List<UserControl>();
        public MindmapMenu(MindMap aMindMap, Controller aController)
        {
            mindMap = aMindMap;
            controller = aController;

            InitializeComponent();

            Loaded += MindmapMenu_Loaded;

        }
        private void MindmapMenu_Loaded(object sender, RoutedEventArgs e)
        {
            Refresh();
        }

        public void Refresh()
        {
            foreach (UserControl nodeControl in nodeControls)
            {
                grdMindMap.Children.Remove(nodeControl);
            }
            grdMindMap.ColumnDefinitions.Clear();
            grdMindMap.RowDefinitions.Clear();
            nodeControls.Clear();
            DrawNode(mindMap.Root);

        }
        private int DrawNode(Node node, int level = 0)
        {
            TextBlock content = new TextBlock
            {
                Text = node.ToString(),
                TextWrapping = TextWrapping.Wrap
            };
            UserControl nodeControl = new UserControl
            {
                Content = content,
                Tag = node,
                BorderBrush = Brushes.Black,
                BorderThickness = new Thickness(1),
                Background = Brushes.LightGray,
                Margin = new Thickness(3),
                Cursor = Cursors.Hand,
                MinHeight = 50
            };
            nodeControl.MouseDoubleClick += NodeControl_Click;
            nodeControl.TouchUp += NodeControl_Click;

            grdMindMap.Children.Add(nodeControl);
            nodeControls.Add(nodeControl);

            grdMindMap.RowDefinitions.Add(new RowDefinition());

```

```
78              Grid.SetRow(nodeControl, grdMindMap.RowDefinitions.Count - 1);
79
80              int maxlvl = (level == 0 ? 1 : level);
81              foreach (Node n in node.Childrens)
82              {
83                  int nodeMaxLvl = DrawNode(n, level + 1);
84                  maxlvl = (maxlvl < nodeMaxLvl ? nodeMaxLvl : maxlvl);
85              }
86
87              while (grdMindMap.ColumnDefinitions.Count <= level)
88              {
89                  grdMindMap.ColumnDefinitions.Add(new ColumnDefinition());
90              }
91              Grid.SetColumn(nodeControl, level);
92
93              return maxlvl + 1;
94          }
95
96          private void NodeControl_Click(object sender, EventArgs e)
97          {
98              Node node = (sender as UserControl).Tag as Node;
99              if (node.Previous == null)
100             {
101                 MessageBox.Show("Pour␣changer␣l'intitul ␣de␣ce␣noeud,␣modifiez␣↩
                        le␣mindmap", "Information", MessageBoxButton.OK);
102             }
103             else
104             {
105                 NodeEdit nodeEdit = new NodeEdit(node);
106                 if (nodeEdit.ShowDialog() == true)
107                 {
108                     if (nodeEdit.Deleted)
109                     {
110                         controller.Delete(node);
111                     }
112                     else
113                     {
114                         string name = nodeEdit.tbxName.Text.Trim();
115                         Node previous = nodeEdit.cbxPrevious.SelectedItem as Node;
116                         controller.UpdateNode(name,previous, node);
117                     }
118                     Refresh();
119                 }
120             }
121         }
122
123         private void Quit_Click(object sender, RoutedEventArgs e)
124         {
125
126             Close();
127         }
128
129         private void Modify_Click(object sender, RoutedEventArgs e)
130         {
131             MindmapEdit mindMapEdit = new MindmapEdit(mindMap);
132             if (mindMapEdit.ShowDialog() == true)
133             {
134                 if (mindMapEdit.Deleted)
135                 {
136                     controller.Delete(mindMap);
137                     DialogResult = false;
138                     Close();
139                 }
140                 else
141                 {
142                     string name = mindMapEdit.tbxName.Text.Trim();
143                     controller.UpdateMindMap(name, mindMap);
144                 }
145                 Refresh();
146             }
147         }
148
149         private void CreateNode_Click(object sender, EventArgs e)
150         {
151             NodeCreate nodeCreate = new NodeCreate(mindMap);
152             if (nodeCreate.ShowDialog() == true)
153             {
154                 string name = nodeCreate.tbxName.Text.Trim();
155                 Node previous = nodeCreate.cbxPrevious.SelectedItem as Node;
```

```
156                 controller.CreateNode(name, previous, mindMap);
157                 Refresh();
158             }
159         }
160 
161     }
162 }
```

Listing 35 – ../../Scrum'o'wall/Scrum'o'wall/Views/MindmapMenu.xaml.cs

2.33 NodeCreate.xaml

```
1  <!--
2   * Author   :   Ga l Mariot
3   * Project  :   Scrum'o'wall
4   * File     :   NodeCreate.xaml
5   * Desc.    :   This file contains the basic template of the NodeCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.NodeCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation␣d'un␣noeud" Height="240.027" Width="490.14" ↩
               WindowStartupLocation="CenterScreen" FontSize="16">
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="161"></ColumnDefinition>
18             <ColumnDefinition></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition Height="47*"></RowDefinition>
22             <RowDefinition Height="50*"></RowDefinition>
23             <RowDefinition Height="62*"></RowDefinition>
24             <RowDefinition Height="62*"></RowDefinition>
25         </Grid.RowDefinitions>
26         <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
               VerticalAlignment="Center" Height="28" Margin="0,10,0,9">Cr ation ↩
               d'un noeud</TextBlock>
27         <TextBlock Grid.Row="1" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10">Nom :</TextBlock>
28         <TextBox x:Name="tbxName" Grid.Row="1" MaxLength="100" Grid.Column="1" ↩
               Margin="10" Stylus.IsPressAndHoldEnabled="False"></TextBox>
29 
30         <TextBlock Grid.Row="2" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10">Pr c dent :</TextBlock>
31         <ComboBox x:Name="cbxPrevious" Grid.Row="2" Grid.Column="1" Margin="10" ↩
               Stylus.IsPressAndHoldEnabled="False"></ComboBox>
32 
33         <Grid Grid.Row="3" Grid.ColumnSpan="2">
34             <Grid.ColumnDefinitions>
35                 <ColumnDefinition></ColumnDefinition>
36                 <ColumnDefinition Width="2*"></ColumnDefinition>
37             </Grid.ColumnDefinitions>
38             <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" ↩
                   Grid.Column="0" Margin="10" ↩
                   Click="BtnCancel_Click">Annuler</Button>
39             <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" ↩
                   Grid.Column="2" Margin="10" ↩
                   Click="BtnConfirm_Click">Confirmer</Button>
40         </Grid>
41     </Grid>
42 </Window>
```

Listing 36 – ../../Scrum'o'wall/Scrum'o'wall/Views/NodeCreate.xaml

2.34 NodeCreate.xaml.cs

```
1  /*
2   * Author   :   Ga l Serge Mariot
3   * Project  :   Scrum'o'wall
4   * File     :   NodeCreate.xaml.cs
```

```
 * Desc.    :   This file contains the logic in the NodeCreate view
 */
using Scrum_o_wall.Classes;
using System;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour NodeCreate.xaml
    /// </summary>
    public partial class NodeCreate : Window
    {
        public NodeCreate(MindMap mindMap)
        {
            InitializeComponent();
            foreach (Node node in mindMap.GetAllNodes())
            {
                cbxPrevious.Items.Add(node);
            }
        }
        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            if (tbxName.Text.Trim().Length > 0 && cbxPrevious.SelectedItem != null)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Listing 37 – ../../Scrum'o'wall/Scrum'o'wall/Views/NodeCreate.xaml.cs

2.35 NodeEdit.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    NodeEdit.xaml
 * Desc.     :    This file contains the basic template of the NodeEdit view
    -->
<Window x:Class="Scrum_o_wall.Views.NodeEdit"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Modification d'un noeud" Height="240.027" Width="490.14" WindowStartupLocation="CenterScreen" FontSize="16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="161"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="47*"></RowDefinition>
            <RowDefinition Height="50*"></RowDefinition>
            <RowDefinition Height="62*"></RowDefinition>
            <RowDefinition Height="62*"></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" VerticalAlignment="Center" Height="28" Margin="0,10,0,9">Modification d'un noeud</TextBlock>
```

```
27          <TextBlock Grid.Row="1" TextAlignment="Right" ↩
                VerticalAlignment="Center" Margin="10">Nom :</TextBlock>
28          <TextBox x:Name="tbxName" Grid.Row="1" MaxLength="100" Grid.Column="1" ↩
                Margin="10" Stylus.IsPressAndHoldEnabled="False"></TextBox>
29
30          <TextBlock Grid.Row="2" TextAlignment="Right" ↩
                VerticalAlignment="Center" Margin="10">Pr c dent :</TextBlock>
31          <ComboBox x:Name="cbxPrevious" Grid.Row="2" Grid.Column="1" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"></ComboBox>
32
33          <Grid Grid.Row="3" Grid.ColumnSpan="2">
34              <Grid.ColumnDefinitions>
35                  <ColumnDefinition></ColumnDefinition>
36                  <ColumnDefinition></ColumnDefinition>
37                  <ColumnDefinition Width="2*"></ColumnDefinition>
38              </Grid.ColumnDefinitions>
39              <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" ↩
                    Grid.Column="0" Margin="10" ↩
                    Click="BtnCancel_Click">Annuler</Button>
40              <Button TouchUp="BtnDelete_Click" x:Name="btnDelete" ↩
                    Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click">Supprimer</Button>
41              <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" ↩
                    Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click">Confirmer</Button>
42          </Grid>
43      </Grid>
44  </Window>
```

Listing 38 – ../../Scrum'o'wall/Scrum'o'wall/Views/NodeEdit.xaml

2.36 NodeEdit.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    NodeEdit.xaml.cs
5    * Desc.     :    This file contains the logic in the NodeEdit view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Windows;
10
11  namespace Scrum_o_wall.Views
12  {
13      /// <summary>
14      /// Logique d'interaction pour NodeCreate.xaml
15      /// </summary>
16      public partial class NodeEdit : Window
17      {
18          public bool Deleted = false;
19          private readonly Node node;
20          public NodeEdit(Node aNode)
21          {
22              node = aNode;
23              InitializeComponent();
24              tbxName.Text = node.Name;
25              if (aNode.Previous != null)
26              {
27                  foreach (Node n in node.MindMap.GetAllNodes())
28                  {
29                      cbxPrevious.Items.Add(n);
30                  }
31                  cbxPrevious.SelectedItem = aNode.Previous;
32              }
33          }
34          private void BtnCancel_Click(object sender, EventArgs e)
35          {
36
37              Close();
38          }
39          private void BtnConfirm_Click(object sender, EventArgs e)
40          {
41              if (tbxName.Text.Trim().Length > 0 && cbxPrevious.SelectedItem != ↩
                    null)
42              {
43                  DialogResult = true;
```

```
                Close();
            }
            else
            {
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Le noeud sera supprim .\n tes-vous s r(e)?", "Attention", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                Deleted = true;
                DialogResult = true;
                Close();
            }
        }
    }
}
```

Listing 39 – ../../Scrum'o'wall/Scrum'o'wall/Views/NodeEdit.xaml.cs

2.37 ProjectCreate.xaml

```
<!--
 * Author   :   Ga l Mariot
 * Project  :   Scrum'o'wall
 * File     :   ProjectCreate.xaml
 * Desc.    :   This file contains the basic template of the ProjectCreate view
    -->
<Window x:Class="Scrum_o_wall.Views.ProjectCreate"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Cr ation d'un projet" Height="500" Width="600" WindowStartupLocation="CenterScreen"  FontSize="16" >
    <Grid Margin="0,0,2,-1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="199"/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="94*"/>
            <RowDefinition Height="54*"/>
            <RowDefinition Height="200*"/>
            <RowDefinition Height="59*"/>
            <RowDefinition Height="63*"/>
        </Grid.RowDefinitions>
        <Label HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="36" Height="58" Margin="172,18,168,18" Width="250" Grid.ColumnSpan="2">Cr er un projet</Label>

        <Label Grid.Row="1" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" >Nom du projet :</Label>
        <TextBox Grid.Row="1" Name="tbxName" VerticalAlignment="Center" Grid.Column="1" Margin="10" MaxLength="50" Stylus.IsPressAndHoldEnabled="False"></TextBox>

        <Label Grid.Row="2" HorizontalAlignment="Right" Margin="10">Description du projet :</Label>
        <TextBox Grid.Row="2" Name="tbxDesc" Grid.Column="1" TextWrapping="WrapWithOverflow" MaxLength="65535" Margin="10" Stylus.IsPressAndHoldEnabled="False"/>

        <Label Grid.Row="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10">Date de d but :</Label>
        <DatePicker Grid.Row="3" VerticalAlignment="Center" Name="tbxDate" Grid.Column="1" Margin="10" Stylus.IsPressAndHoldEnabled="False"/>


        <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" Click="BtnCancel_Click" Grid.Row="4" Margin="10">Annuler</Button>
```

```
40          <Button TouchUp="BtnAddProject_Click" x:Name="btnAddProject" ↩
                Click="BtnAddProject_Click" Grid.Row="4" Grid.Column="1" ↩
                Margin="10">Confirmer</Button>
41      </Grid>
42  </Window>
```

Listing 40 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectCreate.xaml

2.38 ProjectCreate.xaml.cs

```
1  /*
2   * Author    :    Ga l  Serge  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ProjectCreate.xaml.cs
5   * Desc.     :    This file contains the logic in the ProjectCreate view
6   */
7  using System.Windows;
8
9  namespace Scrum_o_wall.Views
10 {
11     /// <summary>
12     /// Logique d'interaction pour Window1.xaml
13     /// </summary>
14     public partial class ProjectCreate : Window
15     {
16         public ProjectCreate()
17         {
18             InitializeComponent();
19         }
20
21         private void BtnCancel_Click(object sender, RoutedEventArgs e)
22         {
23
24             Close();
25         }
26         private void BtnAddProject_Click(object sender, RoutedEventArgs e)
27         {
28             int nameLength = tbxName.Text.Trim().Length;
29             int descLength = tbxDesc.Text.Trim().Length;
30             if (nameLength > 0 && descLength > 0 && tbxDate.SelectedDate != null)
31             {
32                 DialogResult = true;
33                 Close();
34             }
35             else
36             {
37                 MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
38             }
39         }
40     }
41 }
```

Listing 41 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectCreate.xaml.cs

2.39 ProjectEdit.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ProjectEdit.xaml
5   * Desc.     :    This file contains the basic template of the ProjectEdit view
6       -->
7  <Window x:Class="Scrum_o_wall.Views.ProjectEdit"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Modification␣d'un␣projet" Height="500" Width="600" ↩
               WindowStartupLocation="CenterScreen" FontSize="16"  >
15     <Grid Margin="0,0,2,-1">
16         <Grid.ColumnDefinitions>
```

```
17              <ColumnDefinition Width="140*"/>
18              <ColumnDefinition Width="249*"/>
19          </Grid.ColumnDefinitions>
20          <Grid.RowDefinitions>
21              <RowDefinition Height="84*"/>
22              <RowDefinition Height="43*"/>
23              <RowDefinition Height="184*"/>
24              <RowDefinition Height="53*"/>
25              <RowDefinition Height="56*"/>
26              <RowDefinition Height="56*"/>
27          </Grid.RowDefinitions>
28          <Label Grid.ColumnSpan="2" HorizontalAlignment="Center" ↩
                VerticalAlignment="Center" FontSize="36"  >Modification un ↩
                projet</Label>
29
30          <Label Grid.Row="1" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center">Nom du projet :</Label>
31          <TextBox Grid.Row="1" Name="tbxName" Grid.Column="1" MaxLength="50" ↩
                Margin="10"/>
32
33          <Label Grid.Row="2" HorizontalAlignment="Right" ↩
                VerticalAlignment="Top">Description du projet :</Label>
34          <TextBox Grid.Row="2" Name="tbxDesc" Grid.Column="1" MaxLength="65535" ↩
                TextWrapping="WrapWithOverflow" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
35
36          <Label Grid.Row="3" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center">Date de d but :</Label>
37          <DatePicker Grid.Row="3" Name="dtpckrDateBegin" Grid.Column="1" ↩
                Margin="10" VerticalAlignment="Center" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
38
39          <Grid Grid.Row="4" Grid.ColumnSpan="2">
40              <Grid.ColumnDefinitions>
41                  <ColumnDefinition></ColumnDefinition>
42                  <ColumnDefinition></ColumnDefinition>
43              </Grid.ColumnDefinitions>
44              <Button TouchUp="BtnStates_Click" Margin="10" Name="btnStates" ↩
                    Click="BtnStates_Click">Colonnes</Button>
45              <Button TouchUp="BtnUsers_Click" Grid.Column="1" Margin="10" ↩
                    Name="btnUsers" Click="BtnUsers_Click">Utilisateurs ↩
                    assign s </Button>
46          </Grid>
47          <Grid Grid.Row="5" Grid.ColumnSpan="2">
48              <Grid.ColumnDefinitions>
49                  <ColumnDefinition></ColumnDefinition>
50                  <ColumnDefinition></ColumnDefinition>
51                  <ColumnDefinition Width="2*"></ColumnDefinition>
52              </Grid.ColumnDefinitions>
53              <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" ↩
                    TouchUp="BtnCancel_Click">Annuler</Button>
54              <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click" ↩
                    TouchUp="BtnDelete_Click">Supprimer</Button>
55              <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click" ↩
                    TouchUp="BtnConfirm_Click">Confirmer</Button>
56          </Grid>
57      </Grid>
58  </Window>
```

Listing 42 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectEdit.xaml

2.40 ProjectEdit.xaml.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ProjectEdit.xaml.cs
5   * Desc.     :    This file contains the logic in the ProjectEdit view
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Windows;
10
11 namespace Scrum_o_wall.Views
12 {
```

```
13      /// <summary>
14      /// Logique d'interaction pour ProjectEdit.xaml
15      /// </summary>
16      public partial class ProjectEdit : Window
17      {
18          private readonly Project project;
19          public bool Deleted = false;
20          private readonly Controller controller;
21          public ProjectEdit(Project aProject, Controller aController)
22          {
23              project = aProject;
24              controller = aController;
25
26              InitializeComponent();
27
28              tbxDesc.Text = project.Description;
29              tbxName.Text = project.Name;
30              dtpckrDateBegin.SelectedDate = project.Begin;
31          }
32
33          private void BtnConfirm_Click(object sender, EventArgs e)
34          {
35              int nameLength = tbxName.Text.Trim().Length;
36              int descLength = tbxDesc.Text.Trim().Length;
37              if (dtpckrDateBegin.SelectedDate != null && descLength > 0 && ↩
                    nameLength > 0)
38              {
39                  DialogResult = true;
40                  Close();
41              }
42              else
43              {
44                  MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
45              }
46          }
47          private void BtnStates_Click(object sender, EventArgs e)
48          {
49              StateMenu stateMenu = new StateMenu(project, controller);
50              stateMenu.ShowDialog();
51          }
52          private void BtnUsers_Click(object sender, EventArgs e)
53          {
54              UserMenu userMenu = new UserMenu(project, controller.Users, ↩
                    controller);
55              userMenu.ShowDialog();
56          }
57          private void BtnCancel_Click(object sender, EventArgs e)
58          {
59
60              Close();
61          }
62          private void BtnDelete_Click(object sender, EventArgs e)
63          {
64              if (MessageBox.Show("Le␣projet␣sera␣supprim .\n tes-vous␣↩
                    s r(e)?", "Attention", MessageBoxButton.YesNo, ↩
                    MessageBoxImage.Warning) == MessageBoxResult.Yes)
65              {
66                  DialogResult = true;
67                  Deleted = true;
68                  Close();
69              }
70          }
71
72      }
73  }
```

Listing 43 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectEdit.xaml.cs

2.41 ProjectMenu.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ProjectMenu.xaml
5   * Desc.     :    This file contains the basic template of the ProjectMenu view
6      -->
```

```
<Window x:Class="Scrum_o_wall.Views.ProjectMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Scrum'o'Wall" Height="450" Width="800" WindowState="Maximized" WindowStyle="None" FontSize="16" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
    <Window.ContextMenu >
        <ContextMenu >
            <MenuItem Header="Editer le projet" Name="Edit" Click="Edit_Click"/>
            <Separator/>
            <MenuItem Header="Retour" Name="Quit" Click="Quit_Click"/>
        </ContextMenu>
    </Window.ContextMenu>
    <Canvas x:Name="cnvsBacklog">
        <Label Name="lblProjectName" Content="Nom du projet" FontSize="30" Height="60"/>
        <Label Name="lblBacklog" Canvas.Top="65" Height="30">Product Backlog</Label>
        <GroupBox x:Name="gbxUserStories" Height="260" Width="280" Canvas.Top="100" Canvas.Left="10" Header="User stories" >
            <ScrollViewer Name="scrlUserStories" Margin="0,0,0,0">
                <Canvas x:Name="cnvsUserStories">

                </Canvas>
            </ScrollViewer>
        </GroupBox>
        <GroupBox x:Name="gbxSprints" Height="260" Width="220" Canvas.Top="100" Canvas.Right="10" Canvas.Left="295" Header="Sprints"  >
            <ScrollViewer Name="scrlSprints" Margin="0,0,0,0">
                <Canvas x:Name="cnvsSprints">

                </Canvas>
            </ScrollViewer>
        </GroupBox>
        <GroupBox x:Name="gbxMindMap" Height="260" Width="270" Canvas.Top="100" Canvas.Right="10" Canvas.Left="520" Header="MindMaps"   >
            <ScrollViewer Name="scrlMindMaps" Margin="0,0,0,0">
                <Canvas x:Name="cnvsMindMaps">

                </Canvas>
            </ScrollViewer>
        </GroupBox>
        <Button x:Name="btnReturn" TouchUp="Quit_Click" Click="Quit_Click" Content="â  " VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="75" Height="75" BorderBrush="Black" FontSize="45" Canvas.Top="365" Canvas.Left="363">
            <Button.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
        <Button x:Name="btnAddUserStory" TouchUp="BtnAddUserStory_Click" Click="BtnAddUserStory_Click" Content="+" VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="40" Height="40" BorderBrush="Black" FontSize="23" Canvas.Top="320" Canvas.Left="193">
            <Button.Resources>
                <Style TargetType="{x:Type Border}">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
        <Button x:Name="btnAddSprint" TouchUp="BtnAddSprint_Click" Click="BtnAddSprint_Click" Content="+" VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="40" Height="40" BorderBrush="Black" FontSize="23" Canvas.Top="320" Canvas.Left="380">
            <Button.Resources>
                <Style TargetType="{x:Type Border}">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
```

```
67          <Button x:Name="btnAddMindMap" TouchUp="BtnAddMindMap_Click" ↩
                Click="BtnAddMindMap_Click" Content="+" ↩
                VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="40" ↩
                Height="40" BorderBrush="Black" FontSize="23" Canvas.Top="333" ↩
                Canvas.Left="656">
68              <Button.Resources>
69                  <Style TargetType="{x:Type␣Border}">
70                      <Setter Property="CornerRadius" Value="50"/>
71                  </Style>
72              </Button.Resources>
73          </Button>
74      </Canvas>
75  </Window>
```

Listing 44 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectMenu.xaml

2.42 ProjectMenu.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    ProjectMenu.xaml.cs
5    * Desc.     :    This file contains the logic in the ProjectMenu view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Collections.Generic;
10  using System.Threading.Tasks;
11  using System.Windows;
12  using System.Windows.Controls;
13  using System.Windows.Input;
14  using System.Windows.Media;
15
16  namespace Scrum_o_wall.Views
17  {
18      /// <summary>
19      /// Logique d'interaction pour ProjectMenu.xaml
20      /// </summary>
21      public partial class ProjectMenu : Window
22      {
23          private readonly Project project;
24          private readonly Controller controller;
25          private List<UserControl> sprintUserControls = new List<UserControl>();
26          private List<UserControl> userStoriesControls = new List<UserControl>();
27          private List<UserControl> mindMapControls = new List<UserControl>();
28
29          private Dictionary<InputDevice, Point> currentPoint = new ↩
                Dictionary<InputDevice, Point>();
30          private Dictionary<InputDevice, UserControl> infos = new ↩
                Dictionary<InputDevice, UserControl>();
31
32
33          public ProjectMenu(Project aProject, Controller aController)
34          {
35              InitializeComponent();
36              project = aProject;
37              controller = aController;
38
39              lblProjectName.Content = aProject.Name;
40
41              Loaded += ProjectMenu_Loaded;
42              PreviewTouchMove += Canvas_PreviewTouchMove;
43              TouchUp += Canvas_TouchUp;
44
45          }
46
47          private void Canvas_TouchUp(object sender, TouchEventArgs e)
48          {
49              if (currentPoint.ContainsKey(e.Device))
50              {
51                  currentPoint.Remove(e.Device);
52                  infos.Remove(e.Device);
53              }
54          }
55
56          private void Refresh()
57          {
```

```
            int nbMindMaps = project.MindMaps.Count;
            foreach (UserControl mindmapControl in mindMapControls)
            {
                cnvsMindMaps.Children.Remove(mindmapControl);
            }
            mindMapControls.Clear();
            for (int i = 0; i < nbMindMaps; i++)
            {
                MindMap mindMap = project.MindMaps[i];
                //Create Sprint frame
                UserControl mindmapControl = new UserControl
                {
                    Content = mindMap.ToString(),
                    Width = cnvsMindMaps.Width - 40,
                    BorderThickness = new Thickness(1),
                    BorderBrush = Brushes.Black,
                    Cursor = Cursors.Hand,
                    Height = 50,
                    Tag = mindMap
                };
                mindmapControl.MouseDoubleClick += MindmapControl_Click;
                mindmapControl.TouchDown += MindmapControl_Click;

                cnvsMindMaps.Children.Add(mindmapControl);
                mindMapControls.Add(mindmapControl);

                Canvas.SetLeft(mindmapControl, 5);
                Canvas.SetTop(mindmapControl, 60 * i);
            }

            int nbSprints = project.Sprints.Count;
            foreach (UserControl sprintControl in sprintUserControls)
            {
                cnvsSprints.Children.Remove(sprintControl);
            }
            sprintUserControls.Clear();
            for (int i = 0; i < nbSprints; i++)
            {
                Sprint sprint = project.Sprints[i];
                //Create Sprint frame
                UserControl sprintControl = new UserControl
                {
                    Content = sprint.ToString(),
                    Width = cnvsSprints.Width - 40,
                    BorderThickness = new Thickness(1),
                    BorderBrush = Brushes.Black,
                    Cursor = Cursors.Hand,
                    Height = 50,
                    Tag = sprint
                };
                sprintControl.MouseDoubleClick += Sprint_Click;
                sprintControl.TouchDown += Sprint_Click;
                sprintControl.PreviewTouchUp += Sprint_PreviewTouchUp;
                sprintControl.TouchEnter += Sprint_TouchEnter;
                sprintControl.TouchLeave += Sprint_TouchLeave;
                //Change Color by DateRange
                if (DateTime.Now > sprint.Begin && DateTime.Now < sprint.End)
                {
                    //Actual
                    sprintControl.Background = Brushes.LightBlue;
                }
                else if (DateTime.Now < sprint.Begin)
                {
                    //Not beginned
                    sprintControl.Background = Brushes.LightGray;
                }
                else
                {
                    //Already passed
                    sprintControl.Background = Brushes.LightPink;
                }


                cnvsSprints.Children.Add(sprintControl);
                sprintUserControls.Add(sprintControl);

                Canvas.SetLeft(sprintControl, 5);
                Canvas.SetTop(sprintControl, 60 * i);
            }
```

```
137
138             int nbUserStories = project.AllUserStories.Count;
139             foreach (UserControl userStoryControl in userStoriesControls)
140             {
141                 cnvsUserStories.Children.Remove(userStoryControl);
142             }
143             userStoriesControls.Clear();
144             for (int i = 0; i < nbUserStories; i++)
145             {
146                 UserStory userStory = project.AllUserStories[i];
147                 //Create userStory frame
148                 TextBlock content = new TextBlock
149                 {
150                     Text = userStory.ToString(),
151                     TextWrapping = TextWrapping.Wrap
152                 };
153                 UserControl userStoryControl = new UserControl
154                 {
155                     Content = content,
156                     Cursor = Cursors.Hand,
157                     Height = 50,
158                     Tag = userStory,
159                     Width = cnvsUserStories.Width - 40,
160                     BorderBrush = Brushes.Black,
161                     BorderThickness = new Thickness(1)
162                 };
163                 // Change color by limit date (if exists and happened or not)
164                 if (userStory.DateLimit != null)
165                 {
166                     if (DateTime.Now > userStory.DateLimit)
167                     {
168                         userStoryControl.Background = Brushes.LightPink;
169                     }
170                     else
171                     {
172                         userStoryControl.Background = Brushes.LightBlue;
173                     }
174                 }
175                 else
176                 {
177                     userStoryControl.Background = Brushes.LightGray;
178                 }
179                 userStoryControl.MouseDoubleClick += UserStory_MouseDoubleClick;
180                 userStoryControl.PreviewTouchDown += UserStory_PreviewTouchDown;
181                 userStoryControl.TouchUp += UsrCtrlUserStory_TouchUp;
182 
183                 Stylus.SetIsPressAndHoldEnabled(userStoryControl, false);
184 
185                 cnvsUserStories.Children.Add(userStoryControl);
186                 userStoriesControls.Add(userStoryControl);
187 
188                 Canvas.SetLeft(userStoryControl, 5);
189                 Canvas.SetTop(userStoryControl, 60 * i);
190             }
191         }
192 
193         private void MindmapControl_Click(object sender, EventArgs e)
194         {
195             MindMap m = (sender as UserControl).Tag as MindMap;
196             MindmapMenu mindmapMenu = new MindmapMenu(m, controller);
197             mindmapMenu.ShowDialog();
198             Refresh();
199         }
200 
201         private void Sprint_TouchEnter(object sender, TouchEventArgs e)
202         {
203             if (currentPoint.ContainsKey(e.Device))
204             {
205                 (sender as UserControl).BorderThickness = new Thickness(3);
206             }
207         }
208         private void Sprint_TouchLeave(object sender, TouchEventArgs e)
209         {
210             if (currentPoint.ContainsKey(e.Device))
211             {
212                 (sender as UserControl).BorderThickness = new Thickness(1);
213             }
214         }
215 
```

```
216
217         private void UserStoryEditing(UserStory userStory)
218         {
219             UserStoryEdit userStoryEdit = new UserStoryEdit(userStory, project, ↩
                controller);
220
221             if (userStoryEdit.ShowDialog() == true)
222             {
223                 if (userStoryEdit.Deleted)
224                 {
225                     controller.Delete(userStory);
226                 }
227                 else
228                 {
229                     string desc = userStoryEdit.tbxDesc.Text.Trim();
230                     DateTime? dateLimite = ↩
                        userStoryEdit.dtpckrDateLimit.SelectedDate;
231                     int complexity = ↩
                        Convert.ToInt32(userStoryEdit.tbxComplexity.Text);
232                     int completedComplexity = ↩
                        Convert.ToInt32(userStoryEdit.tbxCompletedComplexity.Text);
233                     bool blocked = userStoryEdit.chckBxBlocked.IsChecked == true;
234                     Priority priority = ↩
                        (Priority)userStoryEdit.cbxPriority.SelectedItem;
235                     Classes.Type type = ↩
                        (Classes.Type)userStoryEdit.cbxType.SelectedItem;
236                     State state = userStory.State;
237
238                     controller.UpdateUserStory(desc, dateLimite, complexity, ↩
                        completedComplexity, blocked, priority, type, state, ↩
                        userStory);
239                 }
240                 Refresh();
241             }
242         }
243
244         private void ProjectMenu_Loaded(object sender, RoutedEventArgs e)
245         {
246             //set controls sizes
247             cnvsBacklog.Width = ActualWidth;
248             cnvsBacklog.Height = ActualHeight;
249
250             gbxUserStories.Width = (cnvsBacklog.Width - 25) / 3.0;
251             gbxUserStories.Height = (cnvsBacklog.Height - 250);
252
253             gbxSprints.Width = (cnvsBacklog.Width - 25) / 3.0;
254             gbxSprints.Height = (cnvsBacklog.Height - 250);
255
256             gbxMindMap.Width = (cnvsBacklog.Width - 25) / 3.0;
257             gbxMindMap.Height = (cnvsBacklog.Height - 250);
258
259             cnvsMindMaps.Width = gbxMindMap.Width;
260             cnvsMindMaps.Height = gbxMindMap.Height - 25;
261
262             cnvsSprints.Width = gbxSprints.Width;
263             cnvsSprints.Height = gbxSprints.Height - 25;
264
265             cnvsUserStories.Width = gbxUserStories.Width;
266             cnvsUserStories.Height = gbxUserStories.Height - 25;
267
268             //Set controls positions
269             Canvas.SetLeft(gbxUserStories, 10);
270             Canvas.SetLeft(gbxSprints, gbxUserStories.Width + ↩
                Canvas.GetLeft(gbxUserStories) + 5);
271             Canvas.SetLeft(gbxMindMap, gbxSprints.Width + ↩
                Canvas.GetLeft(gbxSprints) + 5);
272
273             Canvas.SetLeft(lblProjectName, (cnvsBacklog.Width - ↩
                lblProjectName.ActualWidth) / 2.0);
274             Canvas.SetLeft(lblBacklog, (cnvsBacklog.Width - ↩
                lblBacklog.ActualWidth) / 2.0);
275
276             Canvas.SetLeft(btnReturn, (cnvsBacklog.Width - ↩
                btnReturn.ActualWidth) / 2.0);
277             Canvas.SetTop(btnReturn, cnvsBacklog.Height - ↩
                btnReturn.ActualHeight - 10);
278
279             Canvas.SetLeft(btnAddSprint, Canvas.GetLeft(gbxSprints) + ↩
                (gbxSprints.Width - btnAddSprint.ActualWidth) / 2.0);
```

```
280             Canvas.SetTop(btnAddSprint, Canvas.GetTop(gbxSprints) + ↩
                    gbxSprints.Height + 10);
281
282             Canvas.SetLeft(btnAddMindMap, Canvas.GetLeft(gbxMindMap) + ↩
                    (gbxMindMap.Width - btnAddMindMap.ActualWidth) / 2.0);
283             Canvas.SetTop(btnAddMindMap, Canvas.GetTop(gbxMindMap) + ↩
                    gbxMindMap.Height + 10);
284
285             Canvas.SetLeft(btnAddUserStory, Canvas.GetLeft(gbxUserStories) + ↩
                    (gbxUserStories.Width - btnAddUserStory.ActualWidth) / 2.0);
286             Canvas.SetTop(btnAddUserStory, Canvas.GetTop(gbxUserStories) + ↩
                    gbxUserStories.Height + 10);
287
288             //Refresh the window
289             Refresh();
290         }
291         private void UserStory_MouseDoubleClick(object sender, ↩
                MouseButtonEventArgs e)
292         {
293             UserStory userStory = (sender as UserControl).Tag as UserStory;
294             UserStoryEditing(userStory);
295         }
296         private void Sprint_Click(object sender, EventArgs e)
297         {
298             Sprint s = (sender as UserControl).Tag as Sprint;
299             SprintMenu sprintMenu = new SprintMenu(s, controller);
300             sprintMenu.ShowDialog();
301             Refresh();
302         }
303         private void Quit_Click(object sender, EventArgs e)
304         {
305
306             Close();
307         }
308         private void BtnAddUserStory_Click(object sender, EventArgs e)
309         {
310             UserStoryCreate userStoryCreate = new UserStoryCreate(controller);
311             if (userStoryCreate.ShowDialog() == true)
312             {
313                 string desc = userStoryCreate.tbxDesc.Text.Trim();
314                 DateTime? dateLimite = ↩
                        userStoryCreate.dtpckrDateLimit.SelectedDate;
315                 int complexity = ↩
                        Convert.ToInt32(userStoryCreate.tbxComplexity.Text);
316                 Priority priority = ↩
                        (Priority)userStoryCreate.cbxPriority.SelectedItem;
317                 Classes.Type type = ↩
                        (Classes.Type)userStoryCreate.cbxType.SelectedItem;
318                 controller.CreateUserStory(desc, dateLimite, complexity, ↩
                        priority, type, project);
319                 Refresh();
320             }
321         }
322         private void BtnAddSprint_Click(object sender, EventArgs e)
323         {
324             SprintCreate sprintCreate = new SprintCreate();
325             if (sprintCreate.ShowDialog() == true)
326             {
327                 DateTime begin = ↩
                        (DateTime)sprintCreate.dtpckDateBegin.SelectedDate;
328                 DateTime end = (DateTime)sprintCreate.dtpckDateEnd.SelectedDate;
329                 controller.CreateSprint(begin, end, project);
330                 Refresh();
331             }
332         }
333         private void BtnAddMindMap_Click(object sender, EventArgs e)
334         {
335             MindmapCreate mindmapCreate = new MindmapCreate();
336             if (mindmapCreate.ShowDialog() == true)
337             {
338                 string name = mindmapCreate.tbxName.Text.Trim();
339                 controller.CreateMindMap(name, project);
340                 Refresh();
341             }
342         }
343         private void Sprint_PreviewTouchUp(object sender, TouchEventArgs e)
344         {
345             if (currentPoint.ContainsKey(e.Device))
346             {
```

```
347                 currentPoint[e.Device] = e.GetTouchPoint(null).Position;
348                 UserControl sprintControl = sender as UserControl;
349                 sprintControl.BorderThickness = new Thickness(1);
350                 double leftBound = Canvas.GetLeft(sprintControl) + ↩
                        Canvas.GetLeft(gbxSprints);
351                 double rightBound = leftBound + sprintControl.Width;
352                 double topBound = Canvas.GetTop(sprintControl) + ↩
                        Canvas.GetTop(cnvsSprints);
353                 double bottomBound = topBound + sprintControl.Height;
354 
355                 Sprint sprint = sprintControl.Tag as Sprint;
356                 UserStory userStory = (infos[e.Device] as UserControl).Tag as ↩
                        UserStory;
357 
358                 if (controller.AddUserStoryToSprint(userStory, sprint))
359                 {
360                     Task.Factory.StartNew(() => MessageBox.Show("Enregistrement␣↩
                            rajout ", "Confirmation", MessageBoxButton.OK, ↩
                            MessageBoxImage.Information));
361                 }
362                 else
363                 {
364                     Task.Factory.StartNew(() => MessageBox.Show("Enregistrement␣↩
                            d  j  ␣existant", "Erreur", MessageBoxButton.OK, ↩
                            MessageBoxImage.Error));
365                 }
366 
367                 currentPoint.Remove(e.Device);
368                 infos.Remove(e.Device);
369             }
370         }
371         private void UserStory_PreviewTouchDown(object sender, TouchEventArgs e)
372         {
373             if (currentPoint.ContainsKey(e.Device) || infos.ContainsKey(e.Device))
374             {
375                 currentPoint.Remove(e.Device);
376                 infos.Remove(e.Device);
377             }
378             currentPoint.Add(e.Device, e.GetTouchPoint(null).Position);
379             infos.Add(e.Device, sender as UserControl);
380         }
381         private void Canvas_PreviewTouchMove(object sender, TouchEventArgs e)
382         {
383             if (currentPoint.ContainsKey(e.Device))
384             {
385                 currentPoint[e.Device] = e.GetTouchPoint(null).Position;
386             }
387         }
388         private void UsrCtrlUserStory_TouchUp(object sender, TouchEventArgs e)
389         {
390             if (currentPoint.ContainsKey(e.Device))
391             {
392                 currentPoint.Remove(e.Device);
393                 infos.Remove(e.Device);
394             }
395             else
396             {
397                 UserStory userStory = (sender as UserControl).Tag as UserStory;
398                 UserStoryEditing(userStory);
399             }
400         }
401         private void Edit_Click(object sender, RoutedEventArgs e)
402         {
403             ProjectEdit projectEdit = new ProjectEdit(project, controller);
404             if (projectEdit.ShowDialog() == true)
405             {
406                 if (projectEdit.Deleted)
407                 {
408                     controller.Delete(project);
409                     DialogResult = true;
410                     Close();
411                 }
412                 else
413                 {
414                     string name = projectEdit.tbxName.Text.Trim();
415                     string desc = projectEdit.tbxDesc.Text.Trim();
416                     DateTime dateBegin = ↩
                            (DateTime)projectEdit.dtpckrDateBegin.SelectedDate;
417                     controller.UpdateProject(name, desc, dateBegin, project);
```

```
418                     Refresh();
419                 }
420             }
421         }
422
423     }
424 }
```

Listing 45 – ../../Scrum'o'wall/Scrum'o'wall/Views/ProjectMenu.xaml.cs

2.43 SprintCreate.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    SprintCreate.xaml
5   * Desc.     :    This file contains the basic template of the SprintCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.SprintCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation␣d'un␣sprint" Height="281.121" Width="462.967" ↩
               WindowStartupLocation="CenterScreen"  FontSize="16" >
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="140*"/>
18             <ColumnDefinition Width="249*"/>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition Height="84*"/>
22             <RowDefinition Height="43*"/>
23             <RowDefinition Height="43*"/>
24             <RowDefinition Height="53*"/>
25         </Grid.RowDefinitions>
26         <Label Grid.ColumnSpan="2" HorizontalAlignment="Center" ↩
               VerticalAlignment="Center" FontSize="36" Margin="10">Cr er un ↩
               sprint</Label>
27         <Label Grid.Row="1" HorizontalAlignment="Right" ↩
               VerticalAlignment="Center" >Date de d but :</Label>
28         <Label Grid.Row="2" HorizontalAlignment="Right" ↩
               VerticalAlignment="Center" >Date de fin :</Label>
29         <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" ↩
               Click="BtnCancel_Click" Grid.Row="3" Margin="10">Annuler</Button>
30
31         <DatePicker Grid.Row="1" Name="dtpckDateBegin" Grid.Column="1" ↩
               Margin="10" Stylus.IsPressAndHoldEnabled="False"/>
32         <DatePicker Grid.Row="2" Name="dtpckDateEnd" Grid.Column="1" ↩
               Margin="10" Stylus.IsPressAndHoldEnabled="False"/>
33         <Button TouchUp="BtnAddProject_Click" x:Name="btnAddProject" ↩
               Click="BtnAddProject_Click" Grid.Row="3" Grid.Column="1" ↩
               Margin="10">Confirmer</Button>
34     </Grid>
35 </Window>
```

Listing 46 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintCreate.xaml

2.44 SprintCreate.xaml.cs

```
1  /*
2   * Author    :    Ga l  Serge  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    SprintCreate.xaml.cs
5   * Desc.     :    This file contains the logic in the SprintCreate view
6   */
7  using System;
8  using System.Windows;
9
10 namespace Scrum_o_wall.Views
11 {
12     /// <summary>
```

```
13      /// Logique d'interaction pour SprintCreate.xaml
14      /// </summary>
15      public partial class SprintCreate : Window
16      {
17          public SprintCreate()
18          {
19              InitializeComponent();
20          }
21
22          private void BtnCancel_Click(object sender, EventArgs e)
23          {
24
25              Close();
26          }
27          private void BtnAddProject_Click(object sender, EventArgs e)
28          {
29              if (dtpckDateBegin.SelectedDate != null && ↩
                    dtpckDateEnd.SelectedDate != null)
30              {
31                  DialogResult = true;
32                  Close();
33              }
34              else
35              {
36                  MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
37              }
38          }
39
40      }
41  }
```

Listing 47 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintCreate.xaml.cs

2.45 SprintEdit.xaml

```
1   <!--
2    * Author    :    Ga l  Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    SprintEdit.xaml
5    * Desc.     :    This file contains the basic template of the SprintEdit view
6        -->
7   <Window x:Class="Scrum_o_wall.Views.SprintEdit"
8           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
13          mc:Ignorable="d"
14          Title="Modification␣d'un␣sprint" Height="281.121" Width="462.967" ↩
                WindowStartupLocation="CenterScreen"  FontSize="16" >
15      <Grid>
16          <Grid.ColumnDefinitions>
17              <ColumnDefinition Width="140*"/>
18              <ColumnDefinition Width="249*"/>
19          </Grid.ColumnDefinitions>
20          <Grid.RowDefinitions>
21              <RowDefinition Height="84*"/>
22              <RowDefinition Height="43*"/>
23              <RowDefinition Height="43*"/>
24              <RowDefinition Height="53*"/>
25          </Grid.RowDefinitions>
26
27          <Label Grid.ColumnSpan="2" HorizontalAlignment="Center" ↩
                VerticalAlignment="Center" FontSize="36" Margin="10">Modification un ↩
                sprint</Label>
28
29          <Label Grid.Row="1" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Date de d but :</Label>
30          <DatePicker Grid.Row="1" Name="dtpckDateBegin" Grid.Column="1" ↩
                Margin="10"/>
31
32          <Label Grid.Row="2" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Date de fin :</Label>
33          <DatePicker Grid.Row="2" Name="dtpckDateEnd" Grid.Column="1" Margin="10"/>
34
35          <Grid Grid.Row="5" Grid.ColumnSpan="2">
```

```
            <Grid.ColumnDefinitions>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition Width="2*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" TouchUp="BtnCancel_Click">Annuler</Button>
            <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" Click="BtnDelete_Click" TouchUp="BtnDelete_Click">Supprimer</Button>
            <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" Click="BtnConfirm_Click" TouchUp="BtnConfirm_Click">Confirmer</Button>
        </Grid>
    </Grid>
</Window>
```

Listing 48 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintEdit.xaml

2.46 SprintEdit.xaml.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    SprintEdit.xaml.cs
 * Desc.     :    This file contains the logic in the SprintEdit view
 */
using Scrum_o_wall.Classes;
using System;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour SprintEdit.xaml
    /// </summary>
    public partial class SprintEdit : Window
    {
        private readonly Sprint sprint;
        public bool Deleted = false;
        public SprintEdit(Sprint aSprint)
        {
            sprint = aSprint;
            InitializeComponent();

            dtpckDateBegin.SelectedDate = sprint.Begin;
            dtpckDateEnd.SelectedDate = sprint.End;
        }

        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            if (dtpckDateBegin.SelectedDate != null && dtpckDateEnd.SelectedDate != null)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Le sprint sera supprim .\n tes-vous sr(e)?", "Attention", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                DialogResult = true;
                Deleted = true;
                Close();
```

```
            }
        }

    }
}
```

Listing 49 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintEdit.xaml.cs

2.47 SprintMenu.xaml

```
<!--
 * Author   :   Ga l Mariot
 * Project  :   Scrum'o'wall
 * File     :   SprintMenu.xaml
 * Desc.    :   This file contains the basic template of the SprintMenu view
   -->
<Window x:Class="Scrum_o_wall.Views.SprintMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="SprintMenu" Height="450" Width="800" WindowState="Maximized" WindowStyle="None" WindowStartupLocation="CenterScreen" ResizeMode="NoResize" FontSize="16">
    <Window.ContextMenu >
        <ContextMenu >
            <MenuItem Header="Ajouter une colonne" Name="addColumn" Click="AddColumn_Click"/>
            <MenuItem Header="Ouvrir le burndown chart" Name="btnBurndownChartMenu" Click="BtnBurndownChart_Click"/>
            <MenuItem Header="Editer le sprint" Name="btnEditSprint" Click="BtnEditSprint_Click"/>
            <Separator/>
            <MenuItem Header="Retour" Name="Quit" Click="BtnReturn_Click"/>
        </ContextMenu>
    </Window.ContextMenu>
    <Canvas x:Name="cnvsSprint">
        <Label Name="lblProjectName" Content="Nom du projet" FontSize="30" Height="60"/>
        <Label Name="lblSprintName" Content="Nom du sprint" Canvas.Top="65" Height="30"/>
        <Button TouchUp="BtnReturn_Click" x:Name="btnReturn" Click="BtnReturn_Click"  Content="â  " VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="75" Height="75" BorderBrush="Black" FontSize="45" Canvas.Left="80">
            <Button.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
        <Button TouchUp="BtnBurndownChart_Click" x:Name="btnBurndownChart" Click="BtnBurndownChart_Click" Content="    " VerticalContentAlignment="Top"  Canvas.Bottom="10" Width="75" Height="75" BorderBrush="Black" FontSize="45">
            <Button.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="50"/>
                </Style>
            </Button.Resources>
        </Button>
    </Canvas>
</Window>
```

Listing 50 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintMenu.xaml

2.48 SprintMenu.xaml.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   SprintMenu.xaml.cs
```

```
 * Desc.    :   This file contains the logic in the SprintMenu view
 */
using Scrum_o_wall.Classes;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour SprintMenu.xaml
    /// </summary>
    public partial class SprintMenu : Window
    {
        private readonly Sprint sprint;
        private readonly Controller controller;
        private readonly List<GroupBox> columns = new List<GroupBox>();
        private readonly List<UserControl> userStories = new List<UserControl>();

        private readonly Dictionary<InputDevice, Point> currentPoint = new Dictionary<InputDevice, Point>();
        private readonly Dictionary<InputDevice, UserControl> infos = new Dictionary<InputDevice, UserControl>();

        public SprintMenu(Sprint aSprint, Controller aController)
        {
            InitializeComponent();
            sprint = aSprint;
            controller = aController;

            Loaded += SprintMenu_Loaded;
            PreviewTouchMove += SprintMenu_PreviewTouchMove;
            TouchUp += SprintMenu_PreviewTouchUp;

            lblProjectName.Content = sprint.Project.Name;
            lblSprintName.Content = aSprint.ToString();

        }


        private void CleanLists()
        {
            foreach (GroupBox gbx in columns)
            {
                cnvsSprint.Children.Remove(gbx);
            }
            columns.Clear();
            foreach (UserControl userControl in userStories)
            {
                cnvsSprint.Children.Remove(userControl);
            }
            userStories.Clear();
        }
        /// <summary>
        /// Refresh the view with it content
        /// </summary>
        private void Refresh()
        {
            CleanLists();

            //Declare variables for userstories positioning
            Dictionary<State, int> userStoriesPerState = new Dictionary<State, int>();

            foreach (KeyValuePair<int, State> keyValuePair in sprint.Project.States)
            {
                State state = keyValuePair.Value;
                CreateStateColumn(state);
                userStoriesPerState.Add(state, 0);
            }
            /// Place UserStories
            foreach (KeyValuePair<int, UserStory> item in sprint.OrderedUserStories)
            {
```

```
79                  UserStory userStory = item.Value;
80                  CreateUserStoryControl(userStory, ↩
                        userStoriesPerState[userStory.State]);
81                  userStoriesPerState[userStory.State]++;
82              }
83          }
84          private GroupBox CreateStateColumn(State state)
85          {
86              //Create the column
87              GroupBox gbx = new GroupBox
88              {
89                  Height = cnvsSprint.ActualHeight - 190,
90                  Width = cnvsSprint.ActualWidth / sprint.Project.States.Count,
91                  Name = "gbx" + state.Name.Replace("␣", "").Replace("'","),
92                  Header = state.Name,
93                  Tag = state,
94                  BorderBrush = Brushes.Black,
95                  BorderThickness = new Thickness(1)
96              };
97
98              //Positioning and put into canvas
99              cnvsSprint.Children.Add(gbx);
100             Canvas.SetTop(gbx, 100);
101             Canvas.SetBottom(gbx, 90);
102             Canvas.SetLeft(gbx, gbx.Width * columns.Count);
103
104             //added to list and returned
105             columns.Add(gbx);
106             return gbx;
107         }
108
109
110         private UserControl CreateUserStoryControl(UserStory userStory, int ↩
                cptTop = 0)
111         {
112             GroupBox gbx = columns.Where(c => c.Tag == userStory.State).First();
113
114             //Create userStory frame
115             TextBlock content = new TextBlock
116             {
117                 Text = userStory.ToString(),
118                 TextWrapping = TextWrapping.Wrap
119             };
120
121             UserControl userControl = new UserControl
122             {
123                 Content = content,
124                 Tag = userStory,
125                 Height = 50,
126                 Width = gbx.Width - 20,
127                 BorderBrush = Brushes.Black,
128                 Cursor = Cursors.Hand
129             };
130             // Set background color by limit date
131             if (userStory.DateLimit != null)
132             {
133                 if (DateTime.Now > userStory.DateLimit)
134                 {
135                     userControl.Background = Brushes.LightBlue;
136                 }
137                 else
138                 {
139                     userControl.Background = Brushes.LightPink;
140                 }
141             }
142             else
143             {
144                 userControl.Background = Brushes.LightGray;
145             }
146             userControl.MouseDoubleClick += UsrCtrlUserStory_Click;
147             userControl.TouchUp += UsrCtrlUserStory_Click;
148
149             //Events for drag'n'drop
150             userControl.PreviewTouchDown += UserStory_PreviewTouchDown;
151             Stylus.SetIsPressAndHoldEnabled(userControl, false);
152
153             //Add to lists, positionning and return
154             cnvsSprint.Children.Add(userControl);
155             Canvas.SetLeft(userControl, Canvas.GetLeft(gbx) + 10);
```

```
156             Canvas.SetTop(userControl, Canvas.GetTop(gbx) + 30 + 60 * cptTop);
157             userStories.Add(userControl);
158             return userControl;
159         }
160 
161         private void SprintMenu_Loaded(object sender, RoutedEventArgs e)
162         {
163             cnvsSprint.Width = ActualWidth;
164             cnvsSprint.Height = ActualHeight;
165 
166             //set control positions
167             Canvas.SetLeft(lblProjectName, (cnvsSprint.Width - ↩
                    lblProjectName.ActualWidth) / 2.0);
168             Canvas.SetLeft(lblSprintName, (cnvsSprint.Width - ↩
                    lblSprintName.ActualWidth) / 2.0);
169             Canvas.SetLeft(btnBurndownChart, (cnvsSprint.Width) / 2.0 + ↩
                    btnBurndownChart.ActualWidth);
170             Canvas.SetLeft(btnReturn, (cnvsSprint.Width) / 2.0 - ↩
                    btnReturn.ActualWidth);
171 
172             Refresh();
173         }
174         private void SprintMenu_PreviewTouchUp(object sender, TouchEventArgs e)
175         {
176             if (currentPoint.ContainsKey(e.Device))
177             {
178                 //Search the contained groupbox
179                 Point p = e.GetTouchPoint(null).Position;
180                 GroupBox gbxState = null;
181                 foreach (GroupBox col in columns)
182                 {
183                     double leftBound = Canvas.GetLeft(col);
184                     double rightBound = leftBound + col.Width;
185                     double topBound = Canvas.GetTop(col);
186                     double bottomBound = topBound + col.Height;
187                     if (p.X > leftBound && p.X < rightBound &&
188                         p.Y > topBound && p.Y < bottomBound)
189                     {
190                         gbxState = col;
191                         break;
192                     }
193                 }
194                 //if release in a groupbox, change state
195                 if (gbxState != null)
196                 {
197                     gbxState.BorderThickness = new Thickness(1);
198 
199                     State state = gbxState.Tag as State;
200                     UserStory userStory = (infos[e.Device] as UserControl).Tag ↩
                            as UserStory;
201                     controller.UserStorySwitchState(userStory, state);
202                 }
203                 currentPoint.Remove(e.Device);
204                 infos.Remove(e.Device);
205                 Refresh();
206             }
207         }
208         private void SprintMenu_PreviewTouchMove(object sender, TouchEventArgs e)
209         {
210             //Update position of point
211             if (currentPoint.ContainsKey(e.Device))
212             {
213                 currentPoint[e.Device] = e.GetTouchPoint(null).Position;
214             }
215             //Set border thickness to default
216             foreach (GroupBox col in columns)
217             {
218                 col.BorderThickness = new Thickness(1);
219             }
220             //Verify each points to change border thickness if in bounds
221             foreach (KeyValuePair<InputDevice, Point> keyValuePair in ↩
                    currentPoint)
222             {
223                 Point p = keyValuePair.Value;
224                 foreach (GroupBox col in columns)
225                 {
226                     double leftBound = Canvas.GetLeft(col);
227                     double rightBound = leftBound + col.Width;
228                     double topBound = Canvas.GetTop(col);
```

```
                    double bottomBound = topBound + col.Height;

                    if (p.X > leftBound && p.X < rightBound &&
                        p.Y > topBound && p.Y < bottomBound)
                    {
                        col.BorderThickness = new Thickness(3);
                    }
                }
            }
        }
        private void UserStory_PreviewTouchDown(object sender, TouchEventArgs e)
        {
            if (currentPoint.ContainsKey(e.Device))
            {
                currentPoint.Remove(e.Device);
                infos.Remove(e.Device);
            }

            currentPoint.Add(e.Device, e.GetTouchPoint(null).Position);
            infos.Add(e.Device, sender as UserControl);
        }
        private void State_DragLeave(object sender, DragEventArgs e)
        {
            GroupBox gbx = sender as GroupBox;
            gbx.BorderThickness = new Thickness(1);
        }
        private void State_DragEnter(object sender, DragEventArgs e)
        {
            GroupBox gbx = sender as GroupBox;
            gbx.BorderThickness = new Thickness(5);
        }
        private void State_Drop(object sender, DragEventArgs e)
        {
            GroupBox gbx = sender as GroupBox;
            gbx.BorderThickness = new Thickness(1);

            //Make userStory switch
            State state = gbx.Tag as State;
            UserStory userStory = e.Data.GetData("drag") as UserStory;
            controller.UserStorySwitchState(userStory, state);

            Refresh();
        }
        private void UsrCtrlUserStory_Click(object sender, EventArgs e)
        {
            UserStory userStory = (sender as UserControl).Tag as UserStory;
            UserStoryEdit userStoryEdit = new UserStoryEdit(userStory, ↩
                sprint.Project, controller);
            if (userStoryEdit.ShowDialog() == true)
            {
                if (userStoryEdit.Deleted)
                {
                    controller.Delete(userStory);
                }
                else
                {
                    string desc = userStoryEdit.tbxDesc.Text.Trim();
                    DateTime? dateLimite = ↩
                        userStoryEdit.dtpckrDateLimit.SelectedDate;
                    int complexity = ↩
                        Convert.ToInt32(userStoryEdit.tbxComplexity.Text);
                    int completedComplexity = ↩
                        Convert.ToInt32(userStoryEdit.tbxCompletedComplexity.Text);
                    bool blocked = userStoryEdit.chckBxBlocked.IsChecked == true;
                    Priority priority = ↩
                        (Priority)userStoryEdit.cbxPriority.SelectedItem;
                    Classes.Type type = ↩
                        (Classes.Type)userStoryEdit.cbxType.SelectedItem;
                    State state = userStory.State;

                    controller.UpdateUserStory(desc, dateLimite, complexity, ↩
                        completedComplexity, blocked, priority, type, state, ↩
                        userStory);
                }
                Refresh();
            }
        }
        private void AddColumn_Click(object sender, RoutedEventArgs e)
        {
```

```
300             StateMenu stateMenu = new StateMenu(sprint.Project, controller);
301             stateMenu.ShowDialog();
302             Refresh();
303         }
304         private void BtnBurndownChart_Click(object sender, EventArgs e)
305         {
306             BurndownChart burndownChart = new BurndownChart(sprint);
307             burndownChart.ShowDialog();
308         }
309         private void BtnReturn_Click(object sender, EventArgs e)
310         {
311             Close();
312         }
313 
314         private void BtnEditSprint_Click(object sender, RoutedEventArgs e)
315         {
316             SprintEdit sprintEdit = new SprintEdit(sprint);
317             if (sprintEdit.ShowDialog() == true)
318             {
319                 if (sprintEdit.Deleted)
320                 {
321                     controller.Delete(sprint);
322                     DialogResult = true;
323                     Close();
324                 }
325                 else
326                 {
327                     DateTime begin = sprintEdit.dtpckDateBegin.SelectedDate == ↩
                        null ? DateTime.Now : ↩
                        sprintEdit.dtpckDateBegin.SelectedDate.Value;
328                     DateTime end = sprintEdit.dtpckDateEnd.SelectedDate == null ↩
                        ? DateTime.Now + new TimeSpan(7, 0, 0, 0) : ↩
                        sprintEdit.dtpckDateEnd.SelectedDate.Value;
329                     controller.UpdateSprint(begin, end, sprint);
330                     Refresh();
331                 }
332             }
333         }
334     }
335 }
```

Listing 51 – ../../Scrum'o'wall/Scrum'o'wall/Views/SprintMenu.xaml.cs

2.49 StateCreate.xaml

```
1  <!--
2   * Author    :    Ga l  Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    StateCreate.xaml
5   * Desc.     :    This file contains the basic template of the StateCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.StateCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14            Title="Cr ation␣d'un␣ tat " Height="169.532" Width="500.533" ↩
                WindowStartupLocation="CenterScreen" FontSize="16">
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="174"></ColumnDefinition>
18             <ColumnDefinition></ColumnDefinition>
19         </Grid.ColumnDefinitions>
20         <Grid.RowDefinitions>
21             <RowDefinition></RowDefinition>
22             <RowDefinition></RowDefinition>
23             <RowDefinition></RowDefinition>
24         </Grid.RowDefinitions>
25         <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
               VerticalAlignment="Center">Cr ation d'un  tat </TextBlock>
26         <TextBlock Grid.Row="1" TextAlignment="Right" ↩
               VerticalAlignment="Center" Margin="10,13,10,14">Nom de l' tat  ↩
               :</TextBlock>
27         <TextBox Grid.Row="1" Grid.Column="1" MaxLength="30" Margin="10" ↩
               Name="tbxStateName" Stylus.IsPressAndHoldEnabled="False"></TextBox>
```

```
28          <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" Grid.Row="2" ↩
                Grid.Column="0" Margin="10" Click="BtnCancel_Click">Annuler</Button>
29          <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" Grid.Row="2" ↩
                Grid.Column="1" Margin="10" Click="BtnConfirm_Click">Confirmer</Button>
30      </Grid>
31  </Window>
```

Listing 52 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateCreate.xaml

2.50 StateCreate.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    StateCreate.xaml.cs
5    * Desc.     :    This file contains the logic in the StateCreate view
6    */
7   using System;
8   using System.Windows;
9
10  namespace Scrum_o_wall.Views
11  {
12      /// <summary>
13      /// Logique d'interaction pour StateCreate.xaml
14      /// </summary>
15      public partial class StateCreate : Window
16      {
17          public StateCreate()
18          {
19              InitializeComponent();
20          }
21
22          private void BtnCancel_Click(object sender, EventArgs e)
23          {
24
25              Close();
26          }
27          private void BtnConfirm_Click(object sender, EventArgs e)
28          {
29              if (tbxStateName.Text.Trim().Length > 0)
30              {
31                  DialogResult = true;
32                  Close();
33              }
34              else
35              {
36                  MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
37              }
38          }
39
40      }
41  }
```

Listing 53 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateCreate.xaml.cs

2.51 StateEdit.xaml

```
1   <!--
2    * Author    :    Ga l Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    StateEdit.xaml
5    * Desc.     :    This file contains the basic template of the StateEdit view
6        -->
7   <Window x:Class="Scrum_o_wall.Views.StateEdit"
8           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
13          mc:Ignorable="d"
14              Title="Modification␣d'un␣ tat " Height="169.532" Width="500.533" ↩
                    WindowStartupLocation="CenterScreen" FontSize="16">
15      <Grid>
```

```
16          <Grid.ColumnDefinitions>
17              <ColumnDefinition Width="174"></ColumnDefinition>
18              <ColumnDefinition></ColumnDefinition>
19          </Grid.ColumnDefinitions>
20          <Grid.RowDefinitions>
21              <RowDefinition></RowDefinition>
22              <RowDefinition></RowDefinition>
23              <RowDefinition></RowDefinition>
24          </Grid.RowDefinitions>
25          <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
                VerticalAlignment="Center">Modification d'un  tat </TextBlock>
26          <TextBlock Grid.Row="1" TextAlignment="Right" ↩
                VerticalAlignment="Center" Margin="10,13,10,14">Nom de l' tat  ↩
                :</TextBlock>
27          <TextBox Grid.Row="1" Grid.Column="1" MaxLength="30" Margin="10" ↩
                Name="tbxStateName" Stylus.IsPressAndHoldEnabled="False"></TextBox>
28          <Grid Grid.Row="5" Grid.ColumnSpan="2">
29              <Grid.ColumnDefinitions>
30                  <ColumnDefinition></ColumnDefinition>
31                  <ColumnDefinition></ColumnDefinition>
32                  <ColumnDefinition Width="2*"></ColumnDefinition>
33              </Grid.ColumnDefinitions>
34              <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" ↩
                    TouchUp="BtnCancel_Click">Annuler</Button>
35              <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click" ↩
                    TouchUp="BtnDelete_Click">Supprimer</Button>
36              <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click" ↩
                    TouchUp="BtnConfirm_Click">Confirmer</Button>
37          </Grid>
38      </Grid>
39  </Window>
```

Listing 54 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateEdit.xaml

2.52 StateEdit.xaml.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    StateEdit.xaml.cs
5   * Desc.     :    This file contains the logic in the StateEdit view
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Windows;
10
11 namespace Scrum_o_wall.Views
12 {
13     /// <summary>
14     /// Logique d'interaction pour StateEdit.xaml
15     /// </summary>
16     public partial class StateEdit : Window
17     {
18         private readonly State state;
19         public bool Deleted = false;
20         public StateEdit(State aState)
21         {
22             state = aState;
23             InitializeComponent();
24
25             tbxStateName.Text = state.Name;
26         }
27
28         private void BtnCancel_Click(object sender, EventArgs e)
29         {
30
31             Close();
32         }
33         private void BtnConfirm_Click(object sender, EventArgs e)
34         {
35             if (tbxStateName.Text.Trim().Length > 0)
36             {
37                 DialogResult = true;
38                 Close();
39             }
```

```
            else
            {
                MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                    "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("L' tat ␣sera␣supprim .\n tes-vous␣s r(e)?", ↩
                "Attention", MessageBoxButton.YesNo, MessageBoxImage.Warning) == ↩
                MessageBoxResult.Yes)
            {
                DialogResult = true;
                Deleted = true;
                Close();
            }
        }

    }
}
```

Listing 55 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateEdit.xaml.cs

2.53 StateMenu.xaml

```
<!--
 * Author    :    Ga l Mariot
 * Project   :    Scrum'o'wall
 * File      :    StateMenu.xaml
 * Desc.     :    This file contains the basic template of the StateMenu view
    -->
<Window x:Class="Scrum_o_wall.Views.StateMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Gestion␣des␣ tats " Height="450" Width="600" ↩
            Closing="StateMenu_Closing" WindowStartupLocation="CenterScreen" ↩
            FontSize="16" ResizeMode="NoResize">
    <Canvas x:Name="cnvsStates" Margin="0">
        <Grid x:Name="grdStates" Width="594" Height="421">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="80*"></ColumnDefinition>
                <ColumnDefinition Width="20*"></ColumnDefinition>
                <ColumnDefinition Width="80*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="7*"></RowDefinition>
                <RowDefinition Height="40*"></RowDefinition>
            </Grid.RowDefinitions>
            <TextBlock Grid.ColumnSpan="3" FontSize="32" TextAlignment="Center" ↩
                VerticalAlignment="Center" >Gestion des  tats </TextBlock>
            <GroupBox Margin="10" Header=" tats ␣possibles" Grid.Row="1">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition></RowDefinition>
                        <RowDefinition Height="70"></RowDefinition>
                    </Grid.RowDefinitions>
                    <ListBox Margin="10" Name="lstPossibleStates" ↩
                        MouseDoubleClick="Lst_MouseDoubleClick" ↩
                        Stylus.IsPressAndHoldEnabled="False">

                    </ListBox>
                    <Button TouchUp="BtnAddState_Click" ↩
                        Click="BtnAddState_Click" x:Name="btnAddState" ↩
                        Margin="10" Grid.Row="1">Ajouter</Button>
                </Grid>
            </GroupBox>
            <GroupBox Grid.Column="2"  Margin="10" Header=" tats ␣assign s" ↩
                Grid.Row="1">
                <ListBox Margin="10" Name="lstAssignedStates" ↩
                    Stylus.IsPressAndHoldEnabled="False">

                    </ListBox>
            </GroupBox>
```

```
44              <Grid Grid.Row="1" Grid.Column="1">
45                  <Grid.RowDefinitions>
46                      <RowDefinition></RowDefinition>
47                      <RowDefinition></RowDefinition>
48                      <RowDefinition></RowDefinition>
49                  </Grid.RowDefinitions>
50                  <Button TouchUp="BtnGoLeft_Click" Click="BtnGoLeft_Click" ↩
                        Margin="10" x:Name="btnGoLeft" Content="â  "></Button>
51                  <Button TouchUp="BtnGoRight_Click" Click="BtnGoRight_Click" ↩
                        Margin="10" x:Name="btnGoRight" Grid.Row="1" ↩
                        Content="â  "></Button>
52                  <Button TouchUp="BtnSave_Click" Click="BtnSave_Click" ↩
                        Margin="10,10,10,29" x:Name="btnSave" Grid.Row="2" ↩
                        Content="    "></Button>
53              </Grid>
54          </Grid>
55      </Canvas>
56  </Window>
```

Listing 56 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateMenu.xaml

2.54 StateMenu.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    StateMenu.xaml.cs
5    * Desc.     :    This file contains the logic in the StateMenu view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Collections.Generic;
10  using System.Linq;
11  using System.Windows;
12  using System.Windows.Controls;
13  using System.Windows.Input;
14
15  namespace Scrum_o_wall.Views
16  {
17      /// <summary>
18      /// Logique d'interaction pour StateMenu.xaml
19      /// </summary>
20      public partial class StateMenu : Window
21      {
22          private readonly Project project;
23          private readonly Controller controller;
24          public StateMenu(Project aProject, Controller aController)
25          {
26              project = aProject;
27              controller = aController;
28
29              InitializeComponent();
30
31              Refresh();
32          }
33
34          private void Refresh()
35          {
36              if (lstAssignedStates.Items.Count == 0)
37              {
38                  foreach (KeyValuePair<int, State> keyValuePair in project.States)
39                  {
40                      lstAssignedStates.Items.Add(keyValuePair.Value);
41                  }
42              }
43              lstPossibleStates.Items.Clear();
44              foreach (State state in controller.States.Where(s => ↩
                    !lstAssignedStates.Items.Contains(s)))
45              {
46                  lstPossibleStates.Items.Add(state);
47              }
48          }
49
50          private void StateMenu_Closing(object sender, EventArgs e)
51          {
52              if (MessageBox.Show("Les␣changements␣non␣sauvegard s␣seront␣↩
                    perdus.\nVoulez-vous␣sauvegarder␣les␣modifications␣?", ↩
```

```
                "Attention",MessageBoxButton.YesNo,MessageBoxImage.Warning) == ↩
                MessageBoxResult.Yes)
53              {
54                  Save();
55              }
56              Close();
57          }
58          private void BtnGoLeft_Click(object sender, EventArgs e)
59          {
60              if (lstAssignedStates.SelectedItem is State state)
61              {
62                  lstAssignedStates.Items.Remove(state);
63                  lstPossibleStates.Items.Add(state);
64              }
65          }
66          private void BtnGoRight_Click(object sender, EventArgs e)
67          {
68              if (lstPossibleStates.SelectedItem is State state)
69              {
70                  lstPossibleStates.Items.Remove(state);
71                  lstAssignedStates.Items.Add(state);
72              }
73          }
74          private void Save()
75          {
76              List<State> toRemove = new List<State>();
77              List<State> toAdd = new List<State>();
78              foreach (State state in controller.States)
79              {
80                  if (lstAssignedStates.Items.Contains(state) && ↩
                        !project.States.ContainsValue(state))
81                  {
82                      toAdd.Add(state);
83                  }
84                  else if (lstPossibleStates.Items.Contains(state) && ↩
                        project.States.ContainsValue(state))
85                  {
86                      toRemove.Add(state);
87                  }
88              }
89              foreach (State state in toAdd)
90              {
91                  controller.AddStateToProject(state, project);
92              }
93              foreach (State state in toRemove)
94              {
95                  controller.RemoveStateFromProject(state, project);
96              }
97          }
98          private void BtnSave_Click(object sender, EventArgs e)
99          {
100             Save();
101             Close();
102         }
103         private void BtnAddState_Click(object sender, EventArgs e)
104         {
105             StateCreate stateCreate = new StateCreate();
106             if (stateCreate.ShowDialog() == true)
107             {
108                 controller.CreateState(stateCreate.tbxStateName.Text);
109                 Refresh();
110             }
111         }
112
113         private void Lst_MouseDoubleClick(object sender, MouseButtonEventArgs e)
114         {
115             State state = (sender as ListBox).SelectedItem as State;
116             StateEdit stateEdit = new StateEdit(state);
117             if (stateEdit.ShowDialog() == true)
118             {
119                 if (stateEdit.Deleted)
120                 {
121                     controller.Delete(state);
122                 }
123                 else
124                 {
125                     string name = stateEdit.tbxStateName.Text.Trim();
126                     controller.UpdateState(name, state);
127                 }
```

```
                Refresh();
            }
        }
    }
}
```

Listing 57 – ../../Scrum'o'wall/Scrum'o'wall/Views/StateMenu.xaml.cs

2.55 UserCreate.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserCreate.xaml
 * Desc.     :    This file contains the basic template of the UserCreate view
    -->
<Window x:Class="Scrum_o_wall.Views.UserCreate"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Cr ation␣d'un␣utilisateur" Height="178.199" Width="458.533" ↩
            WindowStartupLocation="CenterScreen" FontSize="16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="179"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
            VerticalAlignment="Center">Cr ation d'un utilisateur</TextBlock>
        <TextBlock Grid.Row="1" TextAlignment="Right" ↩
            VerticalAlignment="Center" Margin="10,13,10,14">Nom de l'utilisateur ↩
            :</TextBlock>
        <TextBox Grid.Row="1" Name="tbxUserName" Grid.Column="1" ↩
            MaxLength="255" Margin="10" ↩
            Stylus.IsPressAndHoldEnabled="False"></TextBox>
        <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" ↩
            Click="BtnCancel_Click" Grid.Row="2" Grid.Column="0" ↩
            Margin="10">Annuler</Button>
        <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" ↩
            Click="BtnConfirm_Click" Grid.Row="2" Grid.Column="1" ↩
            Margin="10">Confirmer</Button>
    </Grid>
</Window>
```

Listing 58 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserCreate.xaml

2.56 UserCreate.xaml.cs

```
/*
 * Author    :    Ga l  Serge  Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserCreate.xaml.cs
 * Desc.     :    This file contains the logic in the UserCreate view
 */
using System;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour UserCreate.xaml
    /// </summary>
    public partial class UserCreate : Window
    {
        public UserCreate()
        {
```

```
19              InitializeComponent();
20          }
21
22          private void BtnCancel_Click(object sender, EventArgs e)
23          {
24
25              Close();
26          }
27          private void BtnConfirm_Click(object sender, EventArgs e)
28          {
29              if (tbxUserName.Text.Trim().Length > 0)
30              {
31                  DialogResult = true;
32                  Close();
33              }
34          }
35
36      }
37  }
```

Listing 59 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserCreate.xaml.cs

2.57 UserEdit.xaml

```
1   <!--
2    * Author    :    Ga l  Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    UserEdit.xaml
5    * Desc.     :    This file contains the basic template of the UserEdit view
6       -->
7   <Window x:Class="Scrum_o_wall.Views.UserEdit"
8           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12          xmlns:local="clr-namespace:Scrum_o_wall.Views"
13          mc:Ignorable="d"
14          Title="Modification␣d'un␣utilisateur" Height="166.987" Width="485.083" ↩
                WindowStartupLocation="CenterScreen" FontSize="16">
15      <Grid>
16          <Grid.ColumnDefinitions>
17              <ColumnDefinition Width="173"></ColumnDefinition>
18              <ColumnDefinition></ColumnDefinition>
19          </Grid.ColumnDefinitions>
20          <Grid.RowDefinitions>
21              <RowDefinition></RowDefinition>
22              <RowDefinition></RowDefinition>
23              <RowDefinition></RowDefinition>
24          </Grid.RowDefinitions>
25          <TextBlock Grid.ColumnSpan="2" TextAlignment="Center" FontSize="20" ↩
                VerticalAlignment="Center">Modification d'un utilisateur</TextBlock>
26          <TextBlock Grid.Row="1" TextAlignment="Right" ↩
                VerticalAlignment="Center" Margin="10,13,10,14">Nom de l'utilisateur ↩
                :</TextBlock>
27          <TextBox Grid.Row="1" Name="tbxUserName" Grid.Column="1" ↩
                MaxLength="255" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"></TextBox>
28          <Grid Grid.Row="5" Grid.ColumnSpan="2">
29              <Grid.ColumnDefinitions>
30                  <ColumnDefinition></ColumnDefinition>
31                  <ColumnDefinition></ColumnDefinition>
32                  <ColumnDefinition Width="2*"></ColumnDefinition>
33              </Grid.ColumnDefinitions>
34              <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" ↩
                    TouchUp="BtnCancel_Click">Annuler</Button>
35              <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click" ↩
                    TouchUp="BtnDelete_Click">Supprimer</Button>
36              <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click" ↩
                    TouchUp="BtnConfirm_Click">Confirmer</Button>
37          </Grid>
38      </Grid>
39  </Window>
```

Listing 60 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserEdit.xaml

2.58 UserEdit.xaml.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserEdit.xaml.cs
 * Desc.     :    This file contains the logic in the UserEdit view
 */
using Scrum_o_wall.Classes;
using System;
using System.Windows;

namespace Scrum_o_wall.Views
{
    /// <summary>
    /// Logique d'interaction pour UserEdit.xaml
    /// </summary>
    public partial class UserEdit : Window
    {
        private readonly User user;
        public bool Deleted = false;
        public UserEdit(User aUser)
        {
            user = aUser;

            InitializeComponent();

            tbxUserName.Text = user.Name;
        }

        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            if (tbxUserName.Text.Trim().Length > 0)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un ou plusieurs champ(s) n'est pas rempli !", "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnDelete_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("L'utilisateur sera supprim .\n tes -vous s r(e)?", "Attention", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                Deleted = true;
                DialogResult = true;
                Close();
            }
        }

    }
}
```

Listing 61 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserEdit.xaml.cs

2.59 UserMenu.xaml

```
<!--
 * Author    :    Ga l Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserMenu.xaml
 * Desc.     :    This file contains the basic template of the UserMenu view
    -->
<Window x:Class="Scrum_o_wall.Views.UserMenu"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Gestion␣des␣Utilisateurs" Height="450" Width="600" ↩
            Closing="UserMenu_Closing" WindowStartupLocation="CenterScreen" ↩
            FontSize="16" ResizeMode="NoResize">
    <Canvas x:Name="cnvsStates" Margin="0">
        <Grid x:Name="grdStates" Width="594" Height="421">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="80*"></ColumnDefinition>
                <ColumnDefinition Width="20*"></ColumnDefinition>
                <ColumnDefinition Width="80*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="7*"></RowDefinition>
                <RowDefinition Height="40*"></RowDefinition>
            </Grid.RowDefinitions>
            <TextBlock Grid.ColumnSpan="3" FontSize="32" TextAlignment="Center" ↩
               VerticalAlignment="Center" >Gestion des Utilisateurs</TextBlock>
            <GroupBox Margin="10" Header="Utilisateurs␣possibles" Grid.Row="1">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition></RowDefinition>
                        <RowDefinition Height="70"></RowDefinition>
                    </Grid.RowDefinitions>
                    <ListBox Margin="10" Name="lstPossibleUsers" ↩
                        MouseDoubleClick="Lst_MouseDoubleClick" ↩
                        Stylus.IsPressAndHoldEnabled="False">

                    </ListBox>
                    <Button TouchUp="BtnAddUser_Click" Click="BtnAddUser_Click" ↩
                        x:Name="btnAddUser" Margin="10" ↩
                        Grid.Row="1">Ajouter</Button>
                </Grid>
            </GroupBox>
            <GroupBox Grid.Column="2"  Margin="10" Header="Utilisateurs␣↩
                assign s" Grid.Row="1">
                <ListBox Margin="10" Name="lstAssignedUsers" ↩
                    MouseDoubleClick="Lst_MouseDoubleClick" ↩
                    Stylus.IsPressAndHoldEnabled="False">

                </ListBox>
            </GroupBox>
            <Grid Grid.Row="1" Grid.Column="1">
                <Grid.RowDefinitions>
                    <RowDefinition></RowDefinition>
                    <RowDefinition></RowDefinition>
                    <RowDefinition></RowDefinition>
                </Grid.RowDefinitions>
                <Button TouchUp="BtnGoLeft_Click" Click="BtnGoLeft_Click" ↩
                    Margin="10" x:Name="btnGoLeft" Content="â  "></Button>
                <Button TouchUp="BtnGoRight_Click" Click="BtnGoRight_Click" ↩
                    Margin="10" x:Name="btnGoRight" Grid.Row="1" ↩
                    Content="â  "></Button>
                <Button TouchUp="BtnSave_Click" Click="BtnSave_Click" ↩
                    Margin="10,10,10,28" x:Name="btnSave" Grid.Row="2" ↩
                    Content="    "></Button>
            </Grid>
        </Grid>
    </Canvas>
</Window>
```

Listing 62 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserMenu.xaml

2.60 UserMenu.xaml.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserMenu.xaml.cs
 * Desc.     :    This file contains the logic in the UserMenu view
 */
using Scrum_o_wall.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
```

```
11 using System.Windows.Controls;
12 using System.Windows.Input;
13
14 namespace Scrum_o_wall.Views
15 {
16     /// <summary>
17     /// Logique d'interaction pour UserMenu.xaml
18     /// </summary>
19     public partial class UserMenu : Window
20     {
21         private readonly Controller controller;
22         private readonly IUsersAssigned objectWithAssignedUsers;
23         private readonly List<User> possibleUsers;
24         public UserMenu(IUsersAssigned anObjectWithAssignedUsers, List<User> ↩
               possibilities, Controller aController)
25         {
26             controller = aController;
27             objectWithAssignedUsers = anObjectWithAssignedUsers;
28             possibleUsers = possibilities;
29
30             InitializeComponent();
31
32             Refresh();
33         }
34
35         private void UserMenu_Closing(object sender, EventArgs e)
36         {
37             if (MessageBox.Show("Les␣changements␣non␣sauvegard s␣seront␣↩
                   perdus.\nVoulez-vous␣sauvegarder␣les␣modifications␣?", ↩
                   "Attention", MessageBoxButton.YesNo, MessageBoxImage.Warning) == ↩
                   MessageBoxResult.Yes)
38             {
39                 Save();
40             }
41         }
42         private void Refresh()
43         {
44             lstAssignedUsers.Items.Clear();
45             lstPossibleUsers.Items.Clear();
46             foreach (User user in possibleUsers)
47             {
48                 if (objectWithAssignedUsers.GetUsers().Contains(user))
49                 {
50                     lstAssignedUsers.Items.Add(user);
51                 }
52                 else
53                 {
54                     lstPossibleUsers.Items.Add(user);
55                 }
56             }
57         }
58
59         private void BtnGoLeft_Click(object sender, EventArgs e)
60         {
61             if (lstAssignedUsers.SelectedItem is User state)
62             {
63                 lstAssignedUsers.Items.Remove(state);
64                 lstPossibleUsers.Items.Add(state);
65             }
66         }
67         private void BtnGoRight_Click(object sender, EventArgs e)
68         {
69             if (lstPossibleUsers.SelectedItem is User state)
70             {
71                 lstPossibleUsers.Items.Remove(state);
72                 lstAssignedUsers.Items.Add(state);
73             }
74         }
75         private void Save()
76         {
77             List<User> toRemove = new List<User>();
78             List<User> toAdd = new List<User>();
79             foreach (User user in controller.Users)
80             {
81                 if (lstAssignedUsers.Items.Contains(user) && ↩
                       !objectWithAssignedUsers.GetUsers().Contains(user))
82                 {
83                     toAdd.Add(user);
84                 }
```

```
85                  else if (lstPossibleUsers.Items.Contains(user) && ↩
                        objectWithAssignedUsers.GetUsers().Contains(user))
86                  {
87                      toRemove.Add(user);
88                  }
89              }
90              foreach (User user in toAdd)
91              {
92                  controller.AddUserToIUsersAssigned(user, objectWithAssignedUsers);
93              }
94              foreach (User user in toRemove)
95              {
96                  controller.RemoveUserFromIUsersAssigned(user, ↩
                        objectWithAssignedUsers);
97              }
98          }
99          private void BtnSave_Click(object sender, EventArgs e)
100         {
101             Save();
102             Close();
103         }
104         private void BtnAddUser_Click(object sender, EventArgs e)
105         {
106             UserCreate userCreate = new UserCreate();
107             if (userCreate.ShowDialog() == true)
108             {
109                 controller.CreateUser(userCreate.tbxUserName.Text, ↩
                        objectWithAssignedUsers);
110                 Refresh();
111             }
112         }
113
114         private void Lst_MouseDoubleClick(object sender, MouseButtonEventArgs e)
115         {
116             User user = (sender as ListBox).SelectedItem as User;
117             UserEdit userEdit = new UserEdit(user);
118             if (userEdit.ShowDialog() == true)
119             {
120                 if (userEdit.Deleted)
121                 {
122                     controller.Delete(user);
123                 }
124                 else
125                 {
126                     string name = userEdit.tbxUserName.Text.Trim();
127                     controller.UpdateUser(name, user);
128                 }
129                 Refresh();
130             }
131         }
132     }
133 }
```

Listing 63 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserMenu.xaml.cs

2.61 UserStoryCreate.xaml

```
1  <!--
2   * Author   :   Ga l Mariot
3   * Project  :   Scrum'o'wall
4   * File     :   UserStoryCreate.xaml
5   * Desc.    :   This file contains the basic template of the UserStoryCreate view
6      -->
7  <Window x:Class="Scrum_o_wall.Views.UserStoryCreate"
8          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
9          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
10         xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11         xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12         xmlns:local="clr-namespace:Scrum_o_wall.Views"
13         mc:Ignorable="d"
14         Title="Cr ation␣d'une␣User␣Story" Height="491" Width="508.713" ↩
               WindowStartupLocation="CenterScreen" FontSize="16"  >
15     <Grid>
16         <Grid.ColumnDefinitions>
17             <ColumnDefinition Width="163*"/>
18             <ColumnDefinition Width="226*"/>
19         </Grid.ColumnDefinitions>
```

```
20          <Grid.RowDefinitions>
21              <RowDefinition Height="56*"/>
22              <RowDefinition Height="148*"/>
23              <RowDefinition Height="44*"/>
24              <RowDefinition Height="42*"/>
25              <RowDefinition Height="42*"/>
26              <RowDefinition Height="42*"/>
27              <RowDefinition Height="45*"/>
28          </Grid.RowDefinitions>
29          <Label Grid.ColumnSpan="2" HorizontalAlignment="Center" ↩
                VerticalAlignment="Center" FontSize="36" Height="58" ↩
                Margin="44,-2,44,0" Width="336" >Cr er une User Story</Label>
30
31          <Label Grid.Row="1" HorizontalAlignment="Right" VerticalAlignment="Top" ↩
                >Description de la User Story:</Label>
32          <TextBox Grid.Row="1" TextWrapping="Wrap" MaxLength="65535" ↩
                Name="tbxDesc" Grid.Column="1" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
33
34          <Label Grid.Row="2" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Date limite:</Label>
35          <DatePicker x:Name="dtpckrDateLimit" Grid.Column="1" Margin="10" ↩
                Grid.Row="2" Stylus.IsPressAndHoldEnabled="False"/>
36
37          <Label Grid.Row="3" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center">Estimation de complexit  :</Label>
38          <TextBox Grid.Row="3" Name="tbxComplexity" MaxLength="5" ↩
                Grid.Column="1" KeyDown="TbxComplexity_KeyDown" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
39
40          <Label Grid.Row="4" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center">Priorit </Label>
41          <ComboBox Name="cbxPriority" Grid.Row="4" Margin="10" Grid.Column="10" ↩
                Stylus.IsPressAndHoldEnabled="False"></ComboBox>
42
43          <Label Grid.Row="5" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center">Type</Label>
44          <ComboBox Name="cbxType" Grid.Row="5" Margin="10" Grid.Column="10" ↩
                Stylus.IsPressAndHoldEnabled="False"></ComboBox>
45
46          <Button TouchUp="BtnConfirm_Click" x:Name="btnConfirm" ↩
                Click="BtnConfirm_Click" Grid.Row="6" Grid.Column="1" ↩
                Margin="10">Confirmer</Button>
47          <Button TouchUp="BtnCancel_Click" x:Name="btnCancel" ↩
                Click="BtnCancel_Click" Grid.Row="6" Margin="10">Annuler</Button>
48      </Grid>
49  </Window>
```

Listing 64 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserStoryCreate.xaml

2.62 UserStoryCreate.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    UserStoryCreate.xaml.cs
5    * Desc.     :    This file contains the logic in the UserStoryCreate view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Windows;
10  using System.Windows.Input;
11
12  namespace Scrum_o_wall.Views
13  {
14      /// <summary>
15      /// Logique d'interaction pour UserStoryCreate.xaml
16      /// </summary>
17      public partial class UserStoryCreate : Window
18      {
19          public UserStoryCreate(Controller aController)
20          {
21              InitializeComponent();
22
23              foreach (Priority priority in aController.Priorities)
24              {
25                  cbxPriority.Items.Add(priority);
```

```
            }
            foreach (Classes.Type type in aController.Types)
            {
                cbxType.Items.Add(type);
            }
        }

        private void TbxComplexity_KeyDown(object sender, KeyEventArgs e)
        {
            bool isNumber = (e.Key >= Key.D0 && e.Key <= Key.D9);
            if (!isNumber)
            {
                e.Handled = true;
            }
        }
        private void BtnCancel_Click(object sender, EventArgs e)
        {

            Close();
        }
        private void BtnConfirm_Click(object sender, EventArgs e)
        {
            int descLength = tbxDesc.Text.Trim().Length;
            int complexityLength = tbxComplexity.Text.Trim().Length;
            if (descLength  > 0 && complexityLength > 0 && ↩
                cbxPriority.SelectedIndex >= 0 && cbxType.SelectedIndex >= 0)
            {
                DialogResult = true;
                Close();
            }
            else
            {
                MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                    "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Listing 65 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserStoryCreate.xaml.cs

2.63 UserStoryEdit.xaml

```
<!--
 * Author    :    Ga l  Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserStoryEdit.xaml
 * Desc.     :    This file contains the basic template of the UserStoryEdit view
    -->
<Window x:Class="Scrum_o_wall.Views.UserStoryEdit"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Scrum_o_wall.Views"
        mc:Ignorable="d"
        Title="Modification␣d'une␣User␣Story" Height="722.912" Width="643.787" ↩
            WindowStartupLocation="CenterScreen"  FontSize="16">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="163*"/>
            <ColumnDefinition Width="226*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="55*"/>
            <RowDefinition Height="146*"/>
            <RowDefinition Height="44*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="42*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="41*"/>
            <RowDefinition Height="44*"/>
        </Grid.RowDefinitions>
```

```
33          <Label Grid.ColumnSpan="2" HorizontalAlignment="Center" ↩
                VerticalAlignment="Center" FontSize="36" >Modification d'une User ↩
                Story</Label>
34
35          <Label Grid.Row="1" HorizontalAlignment="Right" ↩
                VerticalAlignment="Top">Description de la User Story :</Label>
36          <TextBox Grid.Row="1" TextWrapping="Wrap" MaxLength="65535" ↩
                Name="tbxDesc" Grid.Column="1" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
37
38          <Label Grid.Row="2" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Date limite :</Label>
39          <DatePicker x:Name="dtpckrDateLimit" Grid.Column="1" Margin="10" ↩
                Grid.Row="2" Stylus.IsPressAndHoldEnabled="False"/>
40
41          <Label Grid.Row="3" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Estimation de complexit  :</Label>
42          <TextBox Grid.Row="3" Name="tbxComplexity" MaxLength="5" ↩
                Grid.Column="1" KeyDown="TbxComplexity_KeyDown" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
43
44          <Label Grid.Row="4" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Complexit  accomplie:</Label>
45          <TextBox Grid.Row="4" Name="tbxCompletedComplexity" MaxLength="5"  ↩
                Grid.Column="1" KeyDown="TbxComplexity_KeyDown" Margin="10" ↩
                Stylus.IsPressAndHoldEnabled="False"/>
46
47          <Label Grid.Row="5" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Priorit  :</Label>
48          <ComboBox Name="cbxPriority" Grid.Row="5" Margin="10" Grid.Column="1" ↩
                Stylus.IsPressAndHoldEnabled="False"></ComboBox>
49
50          <Label Grid.Row="6" HorizontalAlignment="Right" ↩
                VerticalAlignment="Center" >Type :</Label>
51          <ComboBox Name="cbxType" Grid.Row="6" Margin="10" Grid.Column="1" ↩
                Stylus.IsPressAndHoldEnabled="False"></ComboBox>
52
53          <CheckBox Grid.Row="7" x:Name="chckBxBlocked" ↩
                HorizontalAlignment="Center" VerticalAlignment="Center" ↩
                Grid.ColumnSpan="2"  ↩
                Stylus.IsPressAndHoldEnabled="False">Block </CheckBox>
54
55          <Grid Grid.Row="8" Grid.ColumnSpan="2">
56              <Grid.ColumnDefinitions>
57                  <ColumnDefinition></ColumnDefinition>
58                  <ColumnDefinition></ColumnDefinition>
59                  <ColumnDefinition></ColumnDefinition>
60              </Grid.ColumnDefinitions>
61              <Button TouchUp="BtnFiles_Click" Grid.Column="0" x:Name="btnFiles" ↩
                    Click="BtnFiles_Click" Margin="10">Fichiers</Button>
62              <Button TouchUp="BtnComments_Click" Grid.Column="1" ↩
                    x:Name="btnComments" Click="BtnComments_Click" ↩
                    Margin="10">Commentaires</Button>
63              <Button TouchUp="BtnChecklists_Click" Grid.Column="2" ↩
                    x:Name="btnChecklists" Click="BtnChecklists_Click" ↩
                    Margin="10">Checklists</Button>
64          </Grid>
65          <Grid Grid.Row="9" Grid.ColumnSpan="2">
66              <Grid.ColumnDefinitions>
67                  <ColumnDefinition></ColumnDefinition>
68                  <ColumnDefinition></ColumnDefinition>
69              </Grid.ColumnDefinitions>
70              <Button TouchUp="BtnActivities_Click" Grid.Column="0" ↩
                    x:Name="btnActivities" Click="BtnActivities_Click" ↩
                    Margin="10">Activit s</Button>
71              <Button TouchUp="BtnUserAssigned_Click" Grid.Column="1" ↩
                    x:Name="btnUserAssigned" Click="BtnUserAssigned_Click" ↩
                    Margin="10">Utilisateurs assign s</Button>
72          </Grid>
73
74          <Grid Grid.Row="10" Grid.ColumnSpan="2">
75              <Grid.ColumnDefinitions>
76                  <ColumnDefinition></ColumnDefinition>
77                  <ColumnDefinition></ColumnDefinition>
78                  <ColumnDefinition Width="2*"></ColumnDefinition>
79              </Grid.ColumnDefinitions>
80              <Button x:Name="btnCancel"  Margin="10" Click="BtnCancel_Click" ↩
                    TouchUp="BtnCancel_Click">Annuler</Button>
```

```
81              <Button x:Name="btnDelete"  Grid.Column="1" Margin="10" ↩
                    Click="BtnDelete_Click" ↩
                    TouchUp="BtnDelete_Click">Supprimer</Button>
82              <Button x:Name="btnConfirm"  Grid.Column="2" Margin="10" ↩
                    Click="BtnConfirm_Click" ↩
                    TouchUp="BtnConfirm_Click">Confirmer</Button>
83          </Grid>
84      </Grid>
85  </Window>
```

Listing 66 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserStoryEdit.xaml

2.64 UserStoryEdit.xaml.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    UserStoryEdit.xaml.cs
5    * Desc.     :    This file contains the logic in the UserStoryEdit view
6    */
7   using Scrum_o_wall.Classes;
8   using System;
9   using System.Windows;
10  using System.Windows.Input;
11
12  namespace Scrum_o_wall.Views
13  {
14      /// <summary>
15      /// Logique d'interaction pour UserStoryEdit.xaml
16      /// </summary>
17      public partial class UserStoryEdit : Window
18      {
19          private readonly UserStory userStory;
20          private readonly Controller controller;
21          private readonly Project project;
22          public bool Deleted = false;
23
24          public UserStoryEdit(UserStory aUserStory, Project aProject, Controller ↩
                aController)
25          {
26              InitializeComponent();
27              userStory = aUserStory;
28              controller = aController;
29              project = aProject;
30
31              tbxDesc.Text = userStory.Description;
32              dtpckrDateLimit.SelectedDate = userStory.DateLimit;
33              tbxComplexity.Text = userStory.ComplexityEstimation.ToString();
34              tbxCompletedComplexity.Text = ↩
                    userStory.CompletedComplexity.ToString();
35
36              foreach (Priority p in controller.Priorities)
37              {
38                  cbxPriority.Items.Add(p);
39              }
40              foreach (Classes.Type t in controller.Types)
41              {
42                  cbxType.Items.Add(t);
43              }
44
45
46              cbxPriority.SelectedItem = userStory.Priority;
47              cbxType.SelectedItem = userStory.Type;
48
49              chckBxBlocked.IsChecked = userStory.Blocked;
50          }
51
52          private void TbxComplexity_KeyDown(object sender, KeyEventArgs e)
53          {
54              bool isNumber = (e.Key >= Key.D0 && e.Key <= Key.D9);
55              if (!isNumber)
56              {
57                  e.Handled = true;
58              }
59          }
60          private void BtnActivities_Click(object sender, EventArgs e)
61          {
```

```
62              ActivitiesMenu activitiesMenu = new ↩
                    ActivitiesMenu(userStory.Activities);
63              activitiesMenu.ShowDialog();
64          }
65          private void BtnUserAssigned_Click(object sender, EventArgs e)
66          {
67              UserMenu userMenu = new UserMenu(userStory, project.GetUsers(), ↩
                    controller);
68              userMenu.ShowDialog();
69          }
70          private void BtnChecklists_Click(object sender, EventArgs e)
71          {
72              ChecklistMenu checklistMenu = new ChecklistMenu(userStory, ↩
                    controller);
73              checklistMenu.ShowDialog();
74          }
75          private void BtnComments_Click(object sender, EventArgs e)
76          {
77              CommentMenu commentMenu = new CommentMenu(userStory, controller);
78              commentMenu.ShowDialog();
79          }
80          private void BtnFiles_Click(object sender, EventArgs e)
81          {
82              FileMenu fileMenu = new FileMenu(userStory, controller);
83              fileMenu.ShowDialog();
84          }
85          private void BtnConfirm_Click(object sender, EventArgs e)
86          {
87              int descLength = tbxDesc.Text.Trim().Length;
88              int complexityLength = tbxComplexity.Text.Trim().Length;
89              int completedLength = tbxCompletedComplexity.Text.Trim().Length;
90              if (completedLength > 0 && complexityLength > 0 && descLength > 0)
91              {
92                  DialogResult = true;
93                  Close();
94              }
95              else
96              {
97                  MessageBox.Show("Un␣ou␣plusieurs␣champ(s)␣n'est␣pas␣rempli␣!", ↩
                        "Erreur", MessageBoxButton.OK, MessageBoxImage.Error);
98              }
99          }
100         private void BtnCancel_Click(object sender, EventArgs e)
101         {
102
103             Close();
104         }
105         private void BtnDelete_Click(object sender, EventArgs e)
106         {
107             if (MessageBox.Show("La␣user␣story␣sera␣supprim e.\n tes -vous␣↩
                    s r(e)?", "Attention", MessageBoxButton.YesNo, ↩
                    MessageBoxImage.Warning) == MessageBoxResult.Yes)
108             {
109                 DialogResult = true;
110                 Deleted = true;
111                 Close();
112             }
113         }
114
115     }
116 }
```

Listing 67 – ../../Scrum'o'wall/Scrum'o'wall/Views/UserStoryEdit.xaml.cs

3 Modèles

3.1 Activity.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    Activity.cs
 * Desc.     :    This file contains the structure of the Activity class
 */
using System;

namespace Scrum_o_wall.Classes
{
    public class Activity
    {
        private readonly int id;
        private int userStoryId;
        private UserStory userStory;

        public Activity(int id, string description, DateTime dateTime, int ↩
            userStoryId)
        {
            this.id = id;
            Description = description;
            DateTime = dateTime;
            this.userStoryId = userStoryId;
        }

        public int Id => id;
        public string Description { get; set; }
        public DateTime DateTime { get; set; }
        public int UserStoryId => userStoryId;
        public UserStory UserStory
        {
            get => userStory; set
            {
                userStory = value;
                userStoryId = value.Id;
            }
        }

        public override string ToString()
        {
            return DateTime.ToString("dd.MM.yyyy␣-␣HH:mm:ss") + "␣-␣" + ↩
                Description;
        }
    }
}
```

Listing 68 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Activity.cs

3.2 Checklist.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    Checklist.cs
 * Desc.     :    This file contains the structure of the Checklist class
 */
using System.Collections.Generic;

namespace Scrum_o_wall.Classes
{
    public class Checklist
    {
        private readonly int id;
        private int userStoryId;
        private UserStory userStory;

        public Checklist(int id, string name, int userStoryId)
        {
            this.id = id;
            Name = name;
            this.userStoryId = userStoryId;
        }
```

```
23
24          public int Id => id;
25          public string Name { get; set; }
26          public int UserStoryId => userStoryId;
27          public UserStory UserStory
28          {
29              get => userStory; set
30              {
31                  userStory = value;
32                  userStoryId = value.Id;
33              }
34          }
35          public List<ChecklistItem> ChecklistItems { get; set; } = new ↩
                List<ChecklistItem>();
36          public override string ToString()
37          {
38              return Name;
39          }
40      }
41  }
```

Listing 69 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Checklist.cs

3.3 ChecklistItem.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    ChecklistItem.cs
5    * Desc.     :    This file contains the structure of the ChecklistItem class
6    */
7   using System.Collections.Generic;
8
9   namespace Scrum_o_wall.Classes
10  {
11      public class ChecklistItem : IUsersAssigned
12      {
13          private readonly int id;
14          private int checklistId;
15          private Checklist checklist;
16          private readonly List<User> assignedUsers = new List<User>();
17
18          public ChecklistItem(int id, string nameItem, bool done, int checklistId)
19          {
20              this.id = id;
21              NameItem = nameItem;
22              Done = done;
23              this.checklistId = checklistId;
24          }
25
26          public int Id => id;
27          public string NameItem { get; set; }
28          public bool Done { get; set; }
29          public int ChecklistId => checklistId;
30          public Checklist Checklist
31          {
32              get => checklist; set
33              {
34                  checklist = value;
35                  checklistId = value.Id;
36              }
37          }
38
39          public void AddUser(User user)
40          {
41              assignedUsers.Add(user);
42          }
43
44          public List<User> GetUsers()
45          {
46              return assignedUsers;
47          }
48
49          public void RemoveUser(User user)
50          {
51              assignedUsers.Remove(user);
52          }
```

```
53          public override string ToString()
54          {
55              return NameItem;
56          }
57      }
58  }
```

Listing 70 – ../../Scrum'o'wall/Scrum'o'wall/Classes/ChecklistItem.cs

3.4 Comment.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    Comment.cs
5    * Desc.     :    This file contains the structure of the Comment class
6    */
7   using System;
8
9   namespace Scrum_o_wall.Classes
10  {
11      public class Comment
12      {
13          private readonly int id;
14          private int userStoryId;
15          private int userId;
16          private UserStory userStory;
17          private User user;
18
19          public Comment(int id, string description, DateTime dateTime, int ↩
                userStoryId, int userId)
20          {
21              this.id = id;
22              Description = description;
23              DateTime = dateTime;
24              this.userStoryId = userStoryId;
25              this.userId = userId;
26          }
27
28          public int Id => id;
29          public string Description { get; set; }
30          public DateTime DateTime { get; set; }
31          public int UserStoryId => userStoryId;
32          public UserStory UserStory
33          {
34              get => userStory; set
35              {
36                  userStory = value;
37                  userStoryId = value.Id;
38              }
39          }
40          public int UserId => userId;
41          public User User
42          {
43              get => user;
44              set
45              {
46                  user = value;
47                  userId = value.Id;
48              }
49          }
50          public override string ToString()
51          {
52              return User.Name + " - " + Description;
53          }
54      }
55  }
```

Listing 71 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Comment.cs

3.5 File.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
```

```
 * Project  :   Scrum'o'wall
 * File     :   File.cs
 * Desc.    :   This file contains the structure of the TaskFile class
 */

namespace Scrum_o_wall.Classes
{
    public class File
    {
        private readonly int id;
        private int userStoryId;
        private UserStory userStory;


        public File(int id, string name, string description, int userStoryId)
        {
            this.id = id;
            Name = name;
            Description = description;
            this.userStoryId = userStoryId;
        }

        public int Id => id;
        public string Name { get; set; }
        public string Description { get; set; }
        public UserStory UserStory
        {
            get => userStory; set
            {
                userStory = value;
                userStoryId = value.Id;
            }
        }
        public int UserStoryId => userStoryId;

        public override string ToString()
        {
            return Description;
        }
    }
}
```

Listing 72 – ../../Scrum'o'wall/Scrum'o'wall/Classes/File.cs

3.6 IUsersAssigned.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   IUsersAssigned.cs
 * Desc.    :   This file is the interface to some classes with user assigned
 */
using System.Collections.Generic;

namespace Scrum_o_wall.Classes
{
    public interface IUsersAssigned
    {
        List<User> GetUsers();
        void AddUser(User user);
        void RemoveUser(User user);

    }
}
```

Listing 73 – ../../Scrum'o'wall/Scrum'o'wall/Classes/IUsersAssigned.cs

3.7 MindMap.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   MindMap.cs
 * Desc.    :   This file contains the structure of the MindMap class
```

```
 */
using System.Collections.Generic;

namespace Scrum_o_wall.Classes
{
    public class MindMap
    {
        private readonly int id;
        private int projectId;
        private Project project;
        public MindMap(int anId, string aName, int aProjectId)
        {
            id = anId;
            Name = aName;
            projectId = aProjectId;
        }

        public Node Root { get; set; }
        public string Name { get; set; }
        public int Id => id;
        public int ProjectId => projectId;
        public Project Project
        {
            get => project;
            set
            {
                project = value;
                projectId = value.Id;
            }
        }
        public List<Node> GetAllNodes()
        {
            return Root.AllChildrens();
        }
        public override string ToString()
        {
            return Name;
        }
    }
}
```

Listing 74 – ../../Scrum'o'wall/Scrum'o'wall/Classes/MindMap.cs

3.8 Node.cs

```
/*
 * Author   :   Ga l Serge Mariot
 * Project  :   Scrum'o'wall
 * File     :   Node.cs
 * Desc.    :   This file contains the structure of the Node class
 */
using System.Collections.Generic;

namespace Scrum_o_wall.Classes
{
    public class Node
    {

        private readonly int id;
        private Node previous;
        private int? previousId;
        private int mindmapId;
        private MindMap mindMap;

        public Node(int anId, string aName, int? aPreviousId, int aMindmapId)
        {
            id = anId;
            Name = aName;
            previousId = aPreviousId;
            mindmapId = aMindmapId;
        }

        public int Id => id;
        public string Name { get; set; }
        public int? PreviousId => previousId;
        public int Level
        {
```

```
33              get
34              {
35                  int lvl = 0;
36                  Node n = this;
37                  while (n.Previous != null)
38                  {
39                      n = n.Previous;
40                      lvl++;
41                  }
42                  return lvl;
43              }
44          }
45          public Node Previous
46          {
47              get => previous;
48              set
49              {
50                  previous = value;
51                  if (value == null)
52                  {
53                      previousId = null;
54                  }
55                  else
56                  {
57                      previousId = value.Id;
58                  }
59              }
60          }
61          public int MindmapId => mindmapId;
62          public MindMap MindMap
63          {
64              get => mindMap;
65              set
66              {
67                  mindMap = value;
68                  mindmapId = value.Id;
69              }
70          }
71          public List<Node> Childrens { get; set; } = new List<Node>();
72          public List<Node> AllChildrens(List<Node> aList = null)
73          {
74              if (aList == null)
75              {
76                  aList = new List<Node>
77                  {
78                      this
79                  };
80              }
81              aList.AddRange(Childrens);
82              foreach (Node n in Childrens)
83              {
84                  n.AllChildrens(aList);
85              }
86              return aList;
87          }
88          public override string ToString()
89          {
90              return Name;
91          }
92
93      }
94  }
```

Listing 75 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Node.cs

3.9 Priority.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    Priority.cs
5   * Desc.     :    This file contains the structure of the Priority class
6   */
7
8  namespace Scrum_o_wall.Classes
9  {
10     public class Priority
```

```
11      {
12          private readonly int id;
13
14          public Priority(int id, string name)
15          {
16              this.id = id;
17              Name = name;
18          }
19
20          public int Id => id;
21          public string Name { get; set; }
22          public override string ToString()
23          {
24              return Name;
25          }
26      }
27  }
```

Listing 76 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Priority.cs

3.10 Project.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    Project.cs
5    * Desc.     :    This file contains the structure of the Project class
6    */
7   using System;
8   using System.Collections.Generic;
9
10  namespace Scrum_o_wall.Classes
11  {
12      public class Project : IUsersAssigned
13      {
14          private readonly int id;
15          private readonly List<User> assignedUsers = new List<User>();
16
17          /// <summary>
18          /// Create a project with name,description and date
19          /// </summary>
20          /// <param name="aName">the name of the project</param>
21          /// <param name="aDesc">the description of the project</param>
22          /// <param name="aBegin">the date of beginning of project</param>
23          public Project(int anId, string aName, string aDesc, DateTime aBegin)
24          {
25              id = anId;
26              Name = aName;
27              Description = aDesc;
28              Begin = aBegin;
29          }
30
31          //properties declaration
32          public int Id => id;
33          public DateTime Begin { get; set; }
34          public string Name { get; set; }
35          public string Description { get; set; }
36          public List<UserStory> AllUserStories { get; set; } = new ↩
                List<UserStory>();
37          public List<Sprint> Sprints { get; set; } = new List<Sprint>();
38          public List<MindMap> MindMaps { get; set; } = new List<MindMap>();
39          public Dictionary<int, State> States { get; set; } = new ↩
                Dictionary<int, State>();
40
41
42          public void AddUser(User user)
43          {
44              assignedUsers.Add(user);
45          }
46
47          public List<User> GetUsers()
48          {
49              return assignedUsers;
50          }
51
52          public void RemoveUser(User user)
53          {
```

```
54              assignedUsers.Remove(user);
55          }
56
57          public override string ToString()
58          {
59              return string.Format("Project NÂ°{00:0} - {1} - {2} - {3}", Id, ↩
                    Name, Begin, Description);
60          }
61
62      }
63  }
```

Listing 77 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Project.cs

3.11 Sprint.cs

```
1   /*
2    * Author    :    Ga l Serge Mariot
3    * Project   :    Scrum'o'wall
4    * File      :    Sprint.cs
5    * Desc.     :    This file contains the structure of the Sprint class
6    */
7   using System;
8   using System.Collections.Generic;
9
10  namespace Scrum_o_wall.Classes
11  {
12      public class Sprint
13      {
14          private readonly int id;
15          private int projectId;
16          private Project project;
17
18          public Sprint(int anId, DateTime aBegin, DateTime anEnd, int aProjectId)
19          {
20              id = anId;
21              Begin = aBegin;
22              End = anEnd;
23              projectId = aProjectId;
24          }
25          public int Id => id;
26          public DateTime Begin { get; set; }
27          public DateTime End { get; set; }
28          public int ProjectId => projectId;
29          public Project Project
30          {
31              get => project;
32              set
33              {
34                  project = value;
35                  projectId = value.Id;
36              }
37          }
38          public Dictionary<int, UserStory> OrderedUserStories { get; } = new ↩
                Dictionary<int, UserStory>();
39
40          #region Add/Remove userStories
41          public void AddUserStory(int order, UserStory toAdd)
42          {
43              OrderedUserStories.Add(order, toAdd);
44          }
45          public void AddListUserStories(Dictionary<int, UserStory> ListToAdd)
46          {
47              foreach (KeyValuePair<int, UserStory> item in ListToAdd)
48              {
49                  OrderedUserStories.Add(item.Key, item.Value);
50              }
51          }
52          public void RemoveUserStoryByOrder(int order)
53          {
54              OrderedUserStories.Remove(order);
55          }
56          #endregion
57
58          public override string ToString()
59          {
```

```
60              return string.Format("Sprint du {0} au {1}", ↩
                    Begin.ToShortDateString(), End.ToShortDateString());
61          }
62      }
63  }
```

Listing 78 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Sprint.cs

3.12 State.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    State.cs
5   * Desc.     :    This file contains the structure of the State class
6   */
7
8  namespace Scrum_o_wall.Classes
9  {
10     public class State
11     {
12         private readonly int id;
13
14         public State(int id, string name)
15         {
16             this.id = id;
17             Name = name;
18         }
19
20         public int Id => id;
21         public string Name { get; set; }
22         public override string ToString()
23         {
24             return Name;
25         }
26     }
27 }
```

Listing 79 – ../../Scrum'o'wall/Scrum'o'wall/Classes/State.cs

3.13 Type.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    Type.cs
5   * Desc.     :    This file contains the structure of the Type class
6   */
7
8  namespace Scrum_o_wall.Classes
9  {
10     public class Type
11     {
12         private readonly int id;
13
14         public Type(int id, string name)
15         {
16             this.id = id;
17             Name = name;
18         }
19
20         public int Id => id;
21         public string Name { get; set; }
22         public override string ToString()
23         {
24             return Name;
25         }
26     }
27 }
```

Listing 80 – ../../Scrum'o'wall/Scrum'o'wall/Classes/Type.cs

3.14 User.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    User.cs
 * Desc.     :    This file contains the structure of the User class
 */

namespace Scrum_o_wall.Classes
{
    public class User
    {
        private readonly int id;

        public User(int id, string name)
        {
            this.id = id;
            Name = name;
        }

        public int Id => id;
        public string Name { get; set; }
        public override string ToString()
        {
            return Name;
        }
    }
}
```

Listing 81 – ../../Scrum'o'wall/Scrum'o'wall/Classes/User.cs

3.15 UserStory.cs

```
/*
 * Author    :    Ga l Serge Mariot
 * Project   :    Scrum'o'wall
 * File      :    UserStory.cs
 * Desc.     :    This file contains the structure of the UserStory class
 */
using System;
using System.Collections.Generic;

namespace Scrum_o_wall.Classes
{
    public class UserStory : IUsersAssigned
    {
        private readonly int id;
        private int stateId;
        private int projectId;
        private int typeId;
        private int priorityId;
        private readonly List<User> assignedUsers = new List<User>();
        private Type type;
        private Priority priority;
        private State state;
        private Project project;

        public UserStory(int anId, string aDesc, DateTime? aDateLimit, int ↩
            aComplexity, int aCompletedComplexity, bool isBlocked, int ↩
            aProjectId, int aStateId, int aTypeId, int aPriorityId)
        {
            id = anId;
            Description = aDesc;
            DateLimit = aDateLimit;
            ComplexityEstimation = aComplexity;
            CompletedComplexity = aCompletedComplexity;
            Blocked = isBlocked;
            projectId = aProjectId;
            stateId = aStateId;
            typeId = aTypeId;
            priorityId = aPriorityId;
        }

        public int Id => id;
        public int StateId => stateId;
```

```
41          public int ProjectId => projectId;
42          public int TypeId => typeId;
43          public int PriorityId => priorityId;
44          public string Description { get; set; }
45          public State State
46          {
47              get => state;
48              set
49              {
50                  state = value;
51                  stateId = value.Id;
52              }
53          }
54          public DateTime? DateLimit { get; set; }
55          public int CompletedComplexity { get; set; }
56          public int ComplexityEstimation { get; set; }
57          public bool Blocked { get; set; }
58          public Type Type
59          {
60              get => type;
61              set
62              {
63                  type = value;
64                  typeId = value.Id;
65              }
66          }
67          public Project Project
68          {
69              get => project;
70              set
71              {
72                  project = value;
73                  projectId = value.Id;
74              }
75          }
76          public List<File> Files { get; set; } = new List<File>();
77          public List<Comment> Comments { get; set; } = new List<Comment>();
78          public List<Activity> Activities { get; set; } = new List<Activity>();
79          public List<Checklist> Checklists { get; set; } = new List<Checklist>();
80          public Priority Priority
81          {
82              get => priority;
83              set
84              {
85                  priority = value;
86                  priorityId = value.Id;
87              }
88          }
89
90
91          public void AddUser(User user)
92          {
93              assignedUsers.Add(user);
94          }
95
96          public List<User> GetUsers()
97          {
98              return assignedUsers;
99          }
100
101         public void RemoveUser(User user)
102         {
103             assignedUsers.Remove(user);
104         }
105         public override string ToString()
106         {
107             return Description;
108         }
109     }
110 }
```

Listing 82 – ../../Scrum'o'wall/Scrum'o'wall/Classes/UserStory.cs

4 Contrôleurs

4.1 Controller.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    Controller.cs
5   * Desc.     :    This file is the control class. It is used by the view to ↩
        communicate with the database.
6   */
7  using Scrum_o_wall.Classes;
8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11
12 namespace Scrum_o_wall
13 {
14     public class Controller
15     {
16         private List<Project> projects;
17         private List<User> users;
18
19         private List<Classes.Type> types;
20         private List<Priority> priorities;
21         private List<State> states;
22
23         public List<Project> Projects => projects;
24         public List<User> Users => users;
25         public List<Classes.Type> Types => types;
26         public List<Priority> Priorities => priorities;
27         public List<State> States => states;
28
29         public Controller()
30         {
31             GetDatas();
32         }
33
34         /// <summary>
35         /// Get All datas and links all objects between them
36         /// </summary>
37         /// <returns>a list of project containing all infos</returns>
38         private void GetDatas()
39         {
40             //Initialise variables
41             List<Project> allProjects;
42             List<UserStory> allUserStories;
43             List<Sprint> allSprints;
44             List<State> allStates;
45             List<User> allUsers;
46             List<Classes.File> allFiles;
47             List<Activity> allActivities;
48             List<Checklist> allChecklists;
49             List<ChecklistItem> allChecklistItems;
50             List<Priority> allPriorities;
51             List<Comment> allComments;
52             List<Classes.Type> allTypes;
53             List<Node> allNodes;
54             List<MindMap> allMindMaps;
55
56             List<int[]> projectStates;
57             List<int[]> userStoriesSprint;
58             List<int[]> userUserStories;
59             List<int[]> userChecklistItem;
60             List<int[]> userProjects;
61
62
63             allProjects = DB.GetProjects();
64             allSprints = DB.GetSprints();
65             allUserStories = DB.GetUserStories();
66             allStates = DB.GetStates();
67             allUsers = DB.GetUsers();
68             allFiles = DB.GetFiles();
69             allActivities = DB.GetActivities();
70             allChecklists = DB.GetChecklists();
71             allChecklistItems = DB.GetChecklistItems();
72             allPriorities = DB.GetPriorities();
73             allComments = DB.GetComments();
```

```
74              allTypes = DB.GetTypes();
75              allNodes = DB.GetNodes();
76              allMindMaps = DB.GetMindMaps();
77
78
79              #region Link State with project
80              projectStates = DB.GetProjectStates();
81              // 0:IdProject,1:IdState,2:order
82              foreach (int[] item in projectStates)
83              {
84                  List<Project> project = allProjects.Where(p => p.Id == ↩
                        item[0]).ToList();
85                  List<State> state = allStates.Where(s => s.Id == ↩
                        item[1]).ToList();
86
87                  if (project.Count == 1 && state.Count == 1)
88                  {
89                      project[0].States.Add(item[2], state[0]);
90                  }
91
92              }
93              #endregion
94              #region Link UserStory with other classes
95              foreach (UserStory u in allUserStories)
96              {
97                  List<Project> project = allProjects.Where(p => p.Id == ↩
                        u.ProjectId).ToList();
98                  List<State> state = allStates.Where(s => s.Id == ↩
                        u.StateId).ToList();
99                  List<Priority> priority = allPriorities.Where(p => p.Id == ↩
                        u.PriorityId).ToList();
100                 List<Classes.Type> type = allTypes.Where(t => t.Id == ↩
                        u.TypeId).ToList();
101
102                 List<Activity> activities = allActivities.Where(a => ↩
                        a.UserStoryId == u.Id).ToList();
103                 List<Classes.File> files = allFiles.Where(f => f.UserStoryId == ↩
                        u.Id).ToList();
104                 List<Checklist> checklists = allChecklists.Where(c => ↩
                        c.UserStoryId == u.Id).ToList();
105                 List<Comment> comments = allComments.Where(c => c.UserStoryId ↩
                        == u.Id).ToList();
106
107                 if (state.Count == 1)
108                 {
109                     u.State = state[0];
110                 }
111                 if (priority.Count == 1)
112                 {
113                     u.Priority = priority[0];
114                 }
115                 if (type.Count == 1)
116                 {
117                     u.Type = type[0];
118                 }
119                 if (project.Count == 1)
120                 {
121                     project[0].AllUserStories.Add(u);
122                     u.Project = project[0];
123                 }
124                 u.Activities = activities;
125                 foreach (Activity a in activities)
126                 {
127                     a.UserStory = u;
128                 }
129                 u.Checklists = checklists;
130                 foreach (Checklist c in checklists)
131                 {
132                     c.UserStory = u;
133                 }
134                 u.Comments = comments;
135                 foreach (Comment c in comments)
136                 {
137                     List<User> users = allUsers.Where(user => user.Id == ↩
                            c.UserId).ToList();
138                     if (users.Count == 1)
139                     {
140                         c.User = users[0];
141                         c.UserStory = u;
```

```
142                     }
143                 }
144 
145                 u.Files = files;
146                 foreach (Classes.File f in files)
147                 {
148                     f.UserStory = u;
149                 }
150             }
151             #endregion
152             #region Link UserStory with Sprint
153             userStoriesSprint = DB.GetUserStoriesSprint();
154             // 0:IdUserStory,1:IdSprint,2:Order
155             foreach (int[] item in userStoriesSprint)
156             {
157                 List<UserStory> userStory = allUserStories.Where(u => u.Id == ↩
                        item[0]).ToList();
158                 List<Sprint> sprint = allSprints.Where(s => s.Id == ↩
                        item[1]).ToList();
159                 if (userStory.Count == 1 && sprint.Count == 1)
160                 {
161                     sprint[0].AddUserStory(item[2], userStory[0]);
162                 }
163             }
164             #endregion
165             #region Link Sprint with Project
166             foreach (Sprint s in allSprints)
167             {
168                 List<Project> project = allProjects.Where(p => p.Id == ↩
                        s.ProjectId).ToList();
169                 if (project.Count == 1)
170                 {
171                     s.Project = project[0];
172                     project[0].Sprints.Add(s);
173                 }
174             }
175             #endregion
176             #region Link Checklist with ChecklistItems
177             foreach (Checklist chk in allChecklists)
178             {
179                 List<ChecklistItem> checklistItems = allChecklistItems.Where(c ↩
                        => c.ChecklistId == chk.Id).ToList();
180                 chk.ChecklistItems = checklistItems;
181                 foreach (ChecklistItem chkItm in checklistItems)
182                 {
183                     chkItm.Checklist = chk;
184                 }
185 
186             }
187             #endregion
188             #region Link user with UserStory
189             userUserStories = DB.GetUserUserStory();
190             foreach (int[] item in userUserStories)
191             {
192                 // 0:IdUser,1:IdUserStories
193                 List<User> user = allUsers.Where(u => u.Id == item[0]).ToList();
194                 List<UserStory> userStory = allUserStories.Where(us => us.Id == ↩
                        item[1]).ToList();
195                 if (userStory.Count == 1 && user.Count == 1)
196                 {
197                     userStory[0].AddUser(user[0]);
198                 }
199             }
200             #endregion
201             #region Link user with CheckListItem
202             userChecklistItem = DB.GetUserChecklistItem();
203             foreach (int[] item in userChecklistItem)
204             {
205                 // 0:IdUser,1:IdChecklistItem
206                 List<User> user = allUsers.Where(u => u.Id == item[0]).ToList();
207                 List<ChecklistItem> checklistItem = allChecklistItems.Where(c ↩
                        => c.Id == item[1]).ToList();
208                 if (checklistItem.Count == 1 && user.Count == 1)
209                 {
210                     checklistItem[0].AddUser(user[0]);
211                 }
212             }
213             #endregion
214             #region Link user with Project
```

```
215             userProjects = DB.GetUserProject();
216             foreach (int[] item in userProjects)
217             {
218                 // 0:IdUser,1:IdProject
219                 List<User> user = allUsers.Where(u => u.Id == item[0]).ToList();
220                 List<Project> project = allProjects.Where(p => p.Id == ↩
                        item[1]).ToList();
221                 if (project.Count == 1 && user.Count == 1)
222                 {
223                     project[0].AddUser(user[0]);
224                 }
225             }
226             #endregion
227             #region Link Node with Nodes and mindmaps
228             foreach (Node node in allNodes)
229             {
230                 List<MindMap> mindMaps = allMindMaps.Where(m => m.Id == ↩
                        node.MindmapId).ToList();
231                 if (mindMaps.Count == 1)
232                 {
233                     node.MindMap = mindMaps[0];
234                 }
235 
236                 //verify if root node or not
237                 if (node.PreviousId == null)
238                 {
239                     if (mindMaps.Count == 1)
240                     {
241                         mindMaps[0].Root = node;
242                     }
243                 }
244                 else
245                 {
246                     List<Node> nodes = allNodes.Where(n => n.Id == ↩
                            node.PreviousId).ToList();
247                     if (nodes.Count == 1)
248                     {
249                         node.Previous = nodes[0];
250                     }
251                 }
252 
253                 List<Node> childrens = allNodes.Where(n => n.PreviousId == ↩
                        node.Id).ToList();
254                 node.Childrens = childrens;
255 
256             }
257             #endregion
258             #region Link MindMaps with projects
259             foreach (MindMap mindMap in allMindMaps)
260             {
261                 List<Project> projects = allProjects.Where(p => p.Id == ↩
                        mindMap.ProjectId).ToList();
262                 if (projects.Count == 1)
263                 {
264                     mindMap.Project = projects[0];
265                     projects[0].MindMaps.Add(mindMap);
266                 }
267             }
268             #endregion
269 
270             projects = allProjects;
271             users = allUsers;
272             types = allTypes;
273             priorities = allPriorities;
274             states = allStates;
275         }
276 
277 
278 
279         #region Remove Methods
280         //All the creations methods remove the object in controller's values ↩
                and tell DB to remove them
281         public bool Delete(Project project)
282         {
283             bool result = DB.Delete(project);
284             projects.Remove(project);
285             return result;
286         }
287         public bool Delete(Checklist checklist)
```

```
288         {
289             bool result = DB.Delete(checklist);
290             checklist.UserStory.Checklists.Remove(checklist);
291             return result;
292         }
293         public bool Delete(User user)
294         {
295             bool result = DB.Delete(user);
296             users.Remove(user);
297             return result;
298         }
299         public bool Delete(State state)
300         {
301             List<Project> removingState = projects.Where(p => ↩
                    p.States.ContainsValue(state)).ToList();
302             foreach (Project project in removingState)
303             {
304                 RemoveStateFromProject(state, project);
305             }
306             this.States.Remove(state);
307             bool result = DB.Delete(state);
308             return result;
309         }
310         public bool Delete(Sprint sprint)
311         {
312             sprint.Project.Sprints.Remove(sprint);
313             bool result = DB.Delete(sprint);
314             return result;
315         }
316         public bool Delete(ChecklistItem checklistItem)
317         {
318             checklistItem.Checklist.ChecklistItems.Remove(checklistItem);
319             bool result = DB.Delete(checklistItem);
320             return result;
321         }
322         public bool Delete(UserStory userStory)
323         {
324             List<Sprint> sprintRemoveUserStory = ↩
                    userStory.Project.Sprints.Where(s => ↩
                    s.OrderedUserStories.ContainsValue(userStory)).ToList();
325             foreach (Sprint sprint in sprintRemoveUserStory)
326             {
327                 RemoveUserStoryFromSprint(userStory, sprint);
328             }
329             userStory.Project.AllUserStories.Remove(userStory);
330             bool result = DB.Delete(userStory);
331             return result;
332         }
333         public bool Delete(Comment comment)
334         {
335             bool result = DB.Delete(comment);
336             return result;
337         }
338         public bool Delete(Activity activity)
339         {
340             bool result = DB.Delete(activity);
341             return result;
342         }
343         public bool Delete(Classes.File file)
344         {
345             bool result = DB.Delete(file);
346             file.UserStory.Files.Remove(file);
347             return result;
348         }
349         public bool RemoveStateFromProject(State state, Project project)
350         {
351             int order = -1;
352             if (project.States.ContainsValue(state) && project.States.Count > 1)
353             {
354                 foreach (KeyValuePair<int, State> item in project.States)
355                 {
356                     if (item.Value == state)
357                     {
358                         project.States.Remove(item.Key);
359                         order = item.Key;
360 
361                         foreach (UserStory userStory in ↩
                                project.AllUserStories.Where(u => u.State.Id == ↩
                                state.Id))
```

```
362                         {
363                             UserStorySwitchState(userStory, ↩
                                    project.States.First().Value);
364                         }
365 
366                         break;
367                     }
368                 }
369                 DB.RemoveStateFromProject(project, order);
370             }
371             else
372             {
373                 return false;
374             }
375             return true;
376         }
377         public bool RemoveUserFromIUsersAssigned(User user, IUsersAssigned ↩
                usersAssigned)
378         {
379             string typeName = usersAssigned.GetType().Name;
380             switch (typeName)
381             {
382                 case "Project":
383                     RemoveUserFromProject(user, usersAssigned as Project);
384                     break;
385                 case "UserStory":
386                     RemoveUserFromUserStory(user, usersAssigned as UserStory);
387                     CreateActivity(string.Format("{0} a  t  d sassign ", ↩
                            user), usersAssigned as UserStory);
388                     break;
389                 case "ChecklistItem":
390                     RemoveUserFromChecklistItem(user, usersAssigned as ↩
                            ChecklistItem);
391                     break;
392                 default:
393                     return false;
394             }
395             return true;
396         }
397         private bool RemoveUserFromChecklistItem(User user, ChecklistItem ↩
                checklistItem)
398         {
399             bool result = DB.RemoveUserFromChecklistItem(user, checklistItem);
400             checklistItem.RemoveUser(user);
401             return result;
402         }
403         private bool RemoveUserFromUserStory(User user, UserStory userStory)
404         {
405             bool result = DB.RemoveUserFromUserStory(user, userStory);
406             CreateActivity(string.Format("\"{0}\" a  t  d sassign ", user), ↩
                    userStory);
407             userStory.RemoveUser(user);
408             return result;
409         }
410         private bool RemoveUserFromProject(User user, Project project)
411         {
412             bool result = DB.RemoveUserFromProject(user, project);
413             project.RemoveUser(user);
414             return result;
415         }
416         public bool RemoveUserStoryFromSprint(UserStory userStory, Sprint sprint)
417         {
418             bool result = false;
419             if (sprint.OrderedUserStories.ContainsValue(userStory))
420             {
421                 int order = 0;
422                 while (!sprint.OrderedUserStories.ContainsKey(order) || ↩
                        sprint.OrderedUserStories[order] != userStory)
423                 {
424                     order++;
425                 }
426                 CreateActivity(string.Format("A  t  supprim du sprint↩
                        \"{0}\"", sprint), userStory);
427                 sprint.RemoveUserStoryByOrder(order);
428                 result = DB.RemoveUserStoryFromSprint(userStory, sprint, order);
429             }
430             return result;
431         }
432         public bool Delete(MindMap mindMap)
```

```
433         {
434             bool result = DB.Delete(mindMap);
435             mindMap.Project.MindMaps.Remove(mindMap);
436             return result;
437         }
438         public bool Delete(Node node)
439         {
440             bool result = DB.Delete(node);
441             if (!result)
442             {
443                 return result;
444             }
445             foreach (Node n in node.Childrens)
446             {
447                 n.Previous = node.Previous;
448                 node.Previous.Childrens.Add(n);
449             }
450             node.Previous.Childrens.Remove(node);
451             return result;
452         }
453 
454         #endregion
455 
456         #region Update Methods
457         //All the creations methods update the objects in controller's values and send them to DB
458         public bool UserStorySwitchState(UserStory userStory, State state)
459         {
460             bool result = false;
461             if (userStory.State != state)
462             {
463                 CreateActivity(string.Format("Pass de l' tat \"{0}\"  l' tat \"{1}\"", userStory.State, state), userStory);
464                 result = DB.UpdateUserStory(userStory.Description, userStory.DateLimit, userStory.ComplexityEstimation, userStory.CompletedComplexity, userStory.Blocked, userStory.Priority, state, userStory.Type, userStory);
465                 userStory.State = state;
466             }
467             return result;
468         }
469         public bool UpdateCheckListItem(string nameItem, bool done, ChecklistItem item)
470         {
471             bool result = DB.UpdateCheckListItem(nameItem, done, item);
472             item.NameItem = nameItem;
473             item.Done = done;
474             return result;
475         }
476         public bool UpdateUserStory(string description, DateTime? selectedDate, int complexity, int completedComplexity, bool blocked, Priority aPriority, Classes.Type aType, State aState, UserStory userStory)
477         {
478             bool result = false;
479             //Verify if anything is different to create an activity consequently
480             if (userStory.Description != description ||
481                 userStory.DateLimit != selectedDate ||
482                 userStory.ComplexityEstimation != complexity ||
483                 userStory.CompletedComplexity != completedComplexity ||
484                 userStory.Blocked != blocked ||
485                 userStory.Priority != aPriority ||
486                 userStory.Type != aType ||
487                 userStory.State != aState)
488             {
489                 CreateActivity("Les informations ont  t  mises  jour", userStory);
490                 result = DB.UpdateUserStory(description, selectedDate, complexity, completedComplexity, blocked, aPriority, aState, aType, userStory);
491                 userStory.Description = description;
492                 userStory.DateLimit = selectedDate;
493                 userStory.ComplexityEstimation = complexity;
494                 userStory.CompletedComplexity = completedComplexity;
495                 userStory.Blocked = blocked;
496                 userStory.State = aState;
497                 userStory.Priority = aPriority;
498                 userStory.Type = aType;
499             }
500             return result;
```

```
501         }
502         public bool UpdateFile(string fileDescription, Classes.File file)
503         {
504             bool result = DB.UpdateFile(fileDescription, file);
505             file.Description = fileDescription;
506             return result;
507         }
508         public bool UpdateCheckList(string name, List<ChecklistItem> items, ↩
                Checklist checklist)
509         {
510             bool result = DB.UpdateCheckList(name, checklist);
511             checklist.Name = name;
512             checklist.ChecklistItems.Clear();
513             foreach (ChecklistItem item in items)
514             {
515                 if (item.Id == -1)
516                 {
517                     checklist.ChecklistItems.Add(DB.CreateCheckListItem(item.NameItem, ↩
                        checklist));
518                 }
519                 else
520                 {
521                     UpdateCheckListItem(item.NameItem, item.Done, item);
522                     checklist.ChecklistItems.Add(item);
523                 }
524             }
525             return result;
526         }
527         public bool UpdateProject(string name, string description, DateTime ↩
                dateTime, Project aProject)
528         {
529             bool result = DB.UpdateProject(name, description, dateTime, aProject);
530             aProject.Name = name;
531             aProject.Description = description;
532             aProject.Begin = dateTime;
533             return result;
534         }
535         public bool UpdateState(string text, State state)
536         {
537             bool result = DB.UpdateState(text, state);
538             state.Name = text;
539             return result;
540         }
541         public bool UpdateSprint(DateTime begin, DateTime end, Sprint sprint)
542         {
543             bool result = DB.UpdateSprint(begin, end, sprint);
544             sprint.Begin = begin;
545             sprint.End = end;
546             return result;
547         }
548         public bool UpdateUser(string text, User user)
549         {
550             bool result = DB.UpdateUser(text, user);
551             user.Name = text;
552             return result;
553         }
554         public bool UpdateMindMap(string text, MindMap mindMap)
555         {
556             //Update root node to have same text
557             bool result = DB.UpdateMindMap(text, mindMap);
558             mindMap.Name = text;
559             result = result && UpdateNode(text, null, mindMap.Root);
560             return result;
561         }
562         public bool UpdateNode(string text, Node previous, Node node)
563         {
564             bool result = DB.UpdateNode(text, previous, node);
565             if (!result)
566             {
567                 return result;
568             }
569 
570             if (node.Previous != previous)
571             {
572                 //if current previous is a children, redirect to current parent
573                 if (node.AllChildrens().Contains(previous))
574                 {
575                     List<Node> directChilds = node.Childrens;
576                     for (int i = 0; i < directChilds.Count; i++)
```

```
577                     {
578                         Node n = directChilds[i];
579                         UpdateNode(n.Name, node.Previous, n);
580                     }
581                 }
582                 //assign new values
583                 node.Previous.Childrens.Remove(node);
584                 node.Previous = previous;
585                 previous.Childrens.Add(node);
586             }
587             node.Name = text;
588 
589             return result;
590         }
591         #endregion
592 
593         #region Creation Methods
594         //All the creations methods add the objects in controller's values and ↩
                send them to DB
595         public bool AddUserStoryToSprint(UserStory userStory, Sprint sprint)
596         {
597             //Create sprint at the next order, create an activity and return ↩
                    success
598             if (!sprint.OrderedUserStories.ContainsValue(userStory))
599             {
600                 int order = 0;
601                 while (sprint.OrderedUserStories.ContainsKey(order))
602                 {
603                     order++;
604                 }
605                 sprint.AddUserStory(order, userStory);
606                 DB.AddUserStoryToSprint(userStory, sprint, order);
607                 CreateActivity(string.Format("A␣ t ␣ajout ␣au␣sprint␣↩
                        \"{0}\"", sprint), userStory);
608                 return true;
609             }
610             else
611             {
612                 return false;
613             }
614         }
615         public bool AddStateToProject(State state, Project project)
616         {
617             //add state to next after creating and checking
618             int i = 0;
619             while (project.States.ContainsKey(i))
620             {
621                 i++;
622             }
623             bool result = DB.AddStateToProject(state, project, i);
624             project.States.Add(i, state);
625             return result;
626         }
627         public bool AddUserToIUsersAssigned(User user, IUsersAssigned ↩
                usersAssigned)
628         {
629             //Verify which IUsersAssigned it is
630             string typeName = usersAssigned.GetType().Name;
631             switch (typeName)
632             {
633                 case "Project":
634                     AddUserToProject(user, usersAssigned as Project);
635                     break;
636                 case "UserStory":
637                     AddUserToUserStory(user, usersAssigned as UserStory);
638                     break;
639                 case "ChecklistItem":
640                     AddUserToChecklistItem(user, usersAssigned as ChecklistItem);
641                     break;
642                 default:
643                     return false;
644             }
645             return true;
646         }
647         private bool AddUserToUserStory(User user, UserStory userStory)
648         {
649             //create, check,create activity, assign values and return
650             bool result = DB.AddUserToUserStory(user, userStory);
```

```
651             CreateActivity(string.Format("\"{0}\"␣a␣  t  ␣assign ", user), ↩
                    userStory);
652             userStory.AddUser(user);
653             return result;
654         }
655         private bool AddUserToChecklistItem(User user, ChecklistItem ↩
                checklistItem)
656         {
657             //create, check, assign values and return
658             bool result = DB.AddUserToChecklistItem(user, checklistItem);
659             checklistItem.AddUser(user);
660             return result;
661         }
662         private bool AddUserToProject(User user, Project project)
663         {
664             //create, check, assign values and return
665             bool result = DB.AddUserToProject(user, project);
666             project.AddUser(user);
667             return result;
668         }
669         public bool CreateActivity(string description, UserStory userStory)
670         {
671             //create, check, assign values and return
672             Activity activity = DB.CreateActivity(description, DateTime.Now, ↩
                    userStory);
673             bool result = activity != null;
674             userStory.Activities.Add(activity);
675             activity.UserStory = userStory;
676             return result;
677         }
678         public bool CreateUserStory(string description, DateTime? selectedDate, ↩
                int complexity, Priority aPriority, Classes.Type aType, Project ↩
                aProject)
679         {
680             //create and check
681             UserStory userStory = DB.CreateUserStory(description, selectedDate, ↩
                    complexity, aPriority, aType, aProject.States.First().Value, ↩
                    aProject);
682             bool result = userStory != null;
683 
684             //assign values, create activity and return
685             userStory.Priority = aPriority;
686             userStory.Type = aType;
687             userStory.State = aProject.States.First().Value;
688             userStory.Project = aProject;
689             aProject.AllUserStories.Add(userStory);
690             CreateActivity("A␣  t  ␣cr   ", userStory);
691             return result;
692         }
693         public bool CreateSprint(DateTime dateBegin, DateTime dateEnd, Project ↩
                aProject)
694         {
695             //create, check, assign values and return
696             Sprint sprint = DB.CreateSprint(dateBegin, dateEnd, aProject);
697             bool result = sprint != null;
698             sprint.Project = aProject;
699             aProject.Sprints.Add(sprint);
700             return result;
701         }
702         public bool CreateFile(string fileName, string description, UserStory ↩
                userStory)
703         {
704             //create and check
705             Classes.File file = DB.CreateFile(fileName, description, userStory);
706             bool result = file != null;
707 
708             //assign values, create activity and return
709             userStory.Files.Add(file);
710             file.UserStory = userStory;
711             CreateActivity(string.Format("\"{0}\"␣a␣  t  ␣li ", file), ↩
                    userStory);
712             return result;
713         }
714         public bool CreateUser(string name,IUsersAssigned usersAssigned)
715         {
716             //create and check
717             User user = DB.CreateUser(name);
718             bool result = user != null;
719 
```

```
720             //Add user to parent to be viewed in userMenu directly
721             string typeName = usersAssigned.GetType().Name;
722             switch (typeName)
723             {
724                 case "UserStory":
725                     AddUserToProject(user, (usersAssigned as UserStory).Project);
726                     break;
727                 case "ChecklistItem":
728                     AddUserToUserStory(user, (usersAssigned as ↩
                            ChecklistItem).Checklist.UserStory);
729                     break;
730                 case "Project":
731                 default:
732                     break;
733             }
734 
735             //assign values and return
736             users.Add(user);
737             return result;
738         }
739         public bool CreateState(string name)
740         {
741             //create,check, assign values and return
742             State state = DB.CreateState(name);
743             bool result = state != null;
744             states.Add(state);
745             return result;
746         }
747         public bool CreateComment(string text, User user, UserStory userStory)
748         {
749             //create and verify
750             Comment comment = DB.CreateComment(text, userStory, user);
751             bool result = comment != null;
752 
753             //assign values, create activity and return
754             comment.UserStory = userStory;
755             userStory.Comments.Add(comment);
756             comment.User = user;
757 
758             CreateActivity(string.Format("\"{0}\"␣a␣comment ", user), userStory);
759             return result;
760         }
761         public bool CreateCheckListItem(string aName, Checklist checklist)
762         {
763             //create and verify
764             ChecklistItem checklistItem = DB.CreateCheckListItem(aName, ↩
                    checklist);
765             bool result = checklistItem != null;
766 
767             //assign values and return
768             checklistItem.Checklist = checklist;
769             checklist.ChecklistItems.Add(checklistItem);
770             return result;
771         }
772         public Checklist CreateCheckList(string aName, UserStory aUserStory)
773         {
774             //Create, assign values and create activity
775             Checklist checklist = DB.CreateCheckList(aName, aUserStory);
776             aUserStory.Checklists.Add(checklist);
777             checklist.UserStory = aUserStory;
778             CreateActivity(string.Format("La␣checklist␣\"{0}\"␣a␣  t  ␣↩
                    cr   e ", checklist), aUserStory);
779             return checklist;
780         }
781         public bool CreateProject(string aName, string aDesc, DateTime aDate)
782         {
783             //Create and determine if created correctly
784             Project project = DB.CreateProject(aName, aDesc, aDate);
785             bool result = project != null;
786             projects.Add(project);
787 
788             //Assign 3 states by default and create if necessary
789             if (states.Count < 3)
790             {
791                 CreateState("A␣faire");
792                 CreateState("En␣cours");
793                 CreateState("Accompli");
794             }
795             project.States.Add(0, states[0]);
```

```
796             project.States.Add(1, states[1]);
797             project.States.Add(2, states[2]);
798             DB.AddStateToProject(states[0], project, 0);
799             DB.AddStateToProject(states[1], project, 1);
800             DB.AddStateToProject(states[2], project, 2);
801 
802             return result;
803         }
804         public bool CreateMindMap(string aName, Project project)
805         {
806             //Create and verify
807             MindMap mindMap = DB.CreateMindmap(aName, project);
808             bool result = mindMap != null;
809             //assign values and return
810             mindMap.Project = project;
811             mindMap.Root = DB.CreateNode(aName, null, mindMap);
812             project.MindMaps.Add(mindMap);
813             return result;
814         }
815         public bool CreateNode(string aName, Node previous, MindMap mindMap)
816         {
817             //Create and verify
818             Node node = DB.CreateNode(aName, previous, mindMap);
819             bool result = node != null;
820 
821             //set root if previous null and assign values
822             if (previous != null)
823             {
824                 previous.Childrens.Add(node);
825             }
826             else
827             {
828                 mindMap.Root = node;
829             }
830             node.Previous = previous;
831             node.MindMap = mindMap;
832             return result;
833         }
834         #endregion
835 
836     }
837 }
```

Listing 83 – ../../Scrum'o'wall/Scrum'o'wall/Controls/Controller.cs

4.2 DB.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    DB.cs
5   * Desc.     :    This file is a class to access to the database
6   */
7  using Microsoft.Win32;
8  using Scrum_o_wall.Classes;
9  using System;
10 using System.Collections.Generic;
11 using System.Data.OleDb;
12 using System.IO;
13 using System.Windows;
14 
15 namespace Scrum_o_wall
16 {
17     public static class DB
18     {
19         public static string DbFileName;
20         private static OleDbConnection connection;
21         private static OleDbConnection GetConnection()
22         {
23             if (connection is null && DbFileName == null)
24             {
25                 OpenFileDialog opf = new OpenFileDialog
26                 {
27                     Title = "Quel␣base␣de␣donn es␣Access␣utiliser␣?",
28                     DefaultExt = ".accdb|*.accdb",
29                     Filter = "Fichier␣Acces␣(*.accdb)|*.accdb"
30                 };
```

```
31                  do
32                  {
33                      if (opf.ShowDialog() != true)
34                      {
35                          Application.Current.Shutdown();
36                      }
37                  } while (!opf.SafeFileName.Contains(".accdb"));
38                  connection = new ↩
                        OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data␣↩
                        Source=" + opf.FileName + ";Persist␣Security␣Info=False;");
39              }
40              else if (connection is null)
41              {
42                  connection = new ↩
                        OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data␣↩
                        Source=" + DbFileName + ";Persist␣Security␣Info=False;");
43              }
44
45              return connection;
46          }
47
48
49          #region UPDATE
50          //Update the objects and send true if exactly one line has changed
51          public static bool UpdateUserStory(string description, DateTime? ↩
                selectedDate, int complexity, int completedComplexity, bool blocked, ↩
                Priority priority, State state, Classes.Type type, UserStory userStory)
52          {
53              //Initialize variables
54              OleDbCommand cmd;
55              bool result;
56
57              //Open database, build sql statement and prepare
58              DB.GetConnection().Open();
59              cmd = DB.GetConnection().CreateCommand();
60              cmd.CommandText = "UPDATE␣TUserStories␣SET␣DescriptionUserStory␣=␣↩
                    ?,␣DateLimite␣=␣?,␣ComplexityEstimation␣=␣?,␣CompletedComplexity␣↩
                    =␣?,Blocked␣=␣?,IdPriority␣=␣?,␣IdState␣=␣?,␣IdType␣=␣?␣WHERE␣↩
                    IdUserStory␣=␣?;";
61              cmd.Parameters.Add("DescriptionUserStory", OleDbType.LongVarChar, ↩
                    65535);
62              cmd.Parameters.Add("DateLimite", OleDbType.Date);
63              cmd.Parameters.Add("ComplexityEstimation", OleDbType.Integer);
64              cmd.Parameters.Add("CompletedComplexity", OleDbType.Integer);
65              cmd.Parameters.Add("Blocked", OleDbType.Boolean);
66              cmd.Parameters.Add("IdPriority", OleDbType.Integer);
67              cmd.Parameters.Add("IdState", OleDbType.Integer);
68              cmd.Parameters.Add("IdType", OleDbType.Integer);
69              cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
70              cmd.Parameters[0].Value = description;
71              if (selectedDate == null)
72              {
73                  cmd.Parameters[1].Value = DBNull.Value;
74              }
75              else
76              {
77                  cmd.Parameters[1].Value = selectedDate;
78              }
79              cmd.Parameters[2].Value = complexity;
80              cmd.Parameters[3].Value = completedComplexity;
81              cmd.Parameters[4].Value = blocked;
82              cmd.Parameters[5].Value = priority.Id;
83              cmd.Parameters[6].Value = state.Id;
84              cmd.Parameters[7].Value = type.Id;
85              cmd.Parameters[8].Value = userStory.Id;
86
87              //Execute sql statement
88              cmd.Prepare();
89              result = cmd.ExecuteNonQuery() == 1;
90
91              //Close database
92              DB.GetConnection().Close();
93              return result;
94          }
95          public static bool UpdateFile(string fileDescription, Classes.File file)
96          {
97
98              //Initialize variables
99              OleDbCommand cmd;
```

```
100             bool result;
101
102             //Open database, build sql statement and prepare
103             DB.GetConnection().Open();
104             cmd = DB.GetConnection().CreateCommand();
105             cmd.CommandText = "UPDATE␣TFiles␣SET␣DescriptionFile␣=␣?␣WHERE␣↩
                    IdFile␣=␣?;";
106             cmd.Parameters.Add("DescriptionFile", OleDbType.LongVarChar, 65535);
107             cmd.Parameters.Add("IdFile", OleDbType.Integer);
108             cmd.Parameters[0].Value = fileDescription;
109             cmd.Parameters[1].Value = file.Id;
110
111             //Execute sql statement
112             cmd.Prepare();
113             result = cmd.ExecuteNonQuery() == 1;
114
115             //Close database
116             DB.GetConnection().Close();
117             return result;
118         }
119         public static bool UpdateCheckList(string name, Checklist checklist)
120         {
121             //Initialize variables
122             OleDbCommand cmd;
123             bool result;
124
125             //Open database, build sql statement and prepare
126             DB.GetConnection().Open();
127             cmd = DB.GetConnection().CreateCommand();
128             cmd.CommandText = "UPDATE␣TChecklists␣SET␣NameChecklist␣=␣?␣WHERE␣↩
                    IdChecklist␣=␣?;";
129             cmd.Parameters.Add("NameChecklist", OleDbType.VarChar, 255);
130             cmd.Parameters.Add("IdChecklist", OleDbType.Integer);
131             cmd.Parameters[0].Value = name;
132             cmd.Parameters[1].Value = checklist.Id;
133
134             //Execute sql statement
135             cmd.Prepare();
136             result = cmd.ExecuteNonQuery() == 1;
137
138             //Close database
139             DB.GetConnection().Close();
140             return result;
141         }
142         public static bool UpdateCheckListItem(string nameItem, bool done, ↩
                ChecklistItem checklistItem)
143         {
144             //Initialize variables
145             OleDbCommand cmd;
146             bool result;
147
148             //Open database, build sql statement and prepare
149             DB.GetConnection().Open();
150             cmd = DB.GetConnection().CreateCommand();
151             cmd.CommandText = "UPDATE␣TChecklistItems␣SET␣NameItem␣=␣?,␣Done␣=␣↩
                    ?␣WHERE␣IdChecklistItem␣=␣?;";
152             cmd.Parameters.Add("NameItem", OleDbType.VarChar, 255);
153             cmd.Parameters.Add("Done", OleDbType.Boolean);
154             cmd.Parameters.Add("IdChecklistItem", OleDbType.Integer);
155             cmd.Parameters[0].Value = nameItem;
156             cmd.Parameters[1].Value = done;
157             cmd.Parameters[2].Value = checklistItem.Id;
158
159             //Execute sql statement
160             cmd.Prepare();
161             result = cmd.ExecuteNonQuery() == 1;
162
163             //Close database
164             DB.GetConnection().Close();
165             return result;
166         }
167         public static bool UpdateProject(string name, string description, ↩
                DateTime dateTime, Project project)
168         {
169             //Initialize variables
170             OleDbCommand cmd;
171             bool result;
172
173             //Open database, build sql statement and prepare
```

```
174             DB.GetConnection().Open();
175             cmd = DB.GetConnection().CreateCommand();
176             cmd.CommandText = "UPDATE␣TProjects␣SET␣NameProject␣=␣?,Description␣↩
                    =␣?,␣DateBegin␣=␣?␣WHERE␣IdProject␣=␣?;";
177             cmd.Parameters.Add("NameProject", OleDbType.VarChar, 50);
178             cmd.Parameters.Add("Description", OleDbType.LongVarChar, 65535);
179             cmd.Parameters.Add("DateBegin", OleDbType.Date);
180             cmd.Parameters.Add("IdProject", OleDbType.Integer);
181             cmd.Parameters[0].Value = name;
182             cmd.Parameters[1].Value = description;
183             cmd.Parameters[2].Value = dateTime;
184             cmd.Parameters[3].Value = project.Id;
185 
186             //Execute sql statement
187             cmd.Prepare();
188             result = cmd.ExecuteNonQuery() == 1;
189 
190             //Close database
191             DB.GetConnection().Close();
192             return result;
193         }
194         public static bool UpdateSprint(DateTime secBegin, DateTime secEnd, ↩
                Sprint sprint)
195         {
196             //Initialize variables
197             OleDbCommand cmd;
198             bool result;
199 
200             //Open database, build sql statement and prepare
201             DB.GetConnection().Open();
202             cmd = DB.GetConnection().CreateCommand();
203             cmd.CommandText = "UPDATE␣TSprints␣SET␣DateEnd␣=␣?,DateBegin␣=␣?␣↩
                    WHERE␣IdSprint␣=␣?;";
204             cmd.Parameters.Add("DateEnd", OleDbType.Date);
205             cmd.Parameters.Add("DateBegin", OleDbType.Date);
206             cmd.Parameters.Add("IdProject", OleDbType.Integer);
207             cmd.Parameters[0].Value = secEnd;
208             cmd.Parameters[1].Value = secBegin;
209             cmd.Parameters[2].Value = sprint.Id;
210 
211             //Execute sql statement
212             cmd.Prepare();
213             result = cmd.ExecuteNonQuery() == 1;
214 
215             //Close database
216             DB.GetConnection().Close();
217             return result;
218         }
219         public static bool UpdateUser(string text, User user)
220         {
221             //Initialize variables
222             OleDbCommand cmd;
223             bool result;
224 
225             //Open database, build sql statement and prepare
226             DB.GetConnection().Open();
227             cmd = DB.GetConnection().CreateCommand();
228             cmd.CommandText = "UPDATE␣TUsers␣SET␣NameUser␣=␣?␣WHERE␣IdUser␣=␣?;";
229             cmd.Parameters.Add("NameUser", OleDbType.Date);
230             cmd.Parameters.Add("IdUser", OleDbType.Integer);
231             cmd.Parameters[0].Value = text;
232             cmd.Parameters[1].Value = user.Id;
233 
234             //Execute sql statement
235             cmd.Prepare();
236             result = cmd.ExecuteNonQuery() == 1;
237 
238             //Close database
239             DB.GetConnection().Close();
240             return result;
241         }
242         public static bool UpdateState(string text, State state)
243         {
244             //Initialize variables
245             OleDbCommand cmd;
246             bool result;
247 
248             //Open database, build sql statement and prepare
249             DB.GetConnection().Open();
```

```
250             cmd = DB.GetConnection().CreateCommand();
251             cmd.CommandText = "UPDATE TStates SET NameState = ? WHERE IdState = ↩
                    ?;";
252             cmd.Parameters.Add("NameState", OleDbType.VarChar,100);
253             cmd.Parameters.Add("IdState", OleDbType.Integer);
254             cmd.Parameters[0].Value = text;
255             cmd.Parameters[1].Value = state.Id;
256 
257             //Execute sql statement
258             cmd.Prepare();
259             result = cmd.ExecuteNonQuery() == 1;
260 
261             //Close database
262             DB.GetConnection().Close();
263             return result;
264         }
265         public static bool UpdateMindMap(string text, MindMap mindMap)
266         {
267             //Initialize variables
268             OleDbCommand cmd;
269             bool result;
270 
271             //Open database, build sql statement and prepare
272             DB.GetConnection().Open();
273             cmd = DB.GetConnection().CreateCommand();
274             cmd.CommandText = "UPDATE TMindMaps SET NameMindMap = ? WHERE ↩
                    IdMindMap = ?;";
275             cmd.Parameters.Add("NameMindMap", OleDbType.VarChar, 255);
276             cmd.Parameters.Add("IdMindMap", OleDbType.Integer);
277             cmd.Parameters[0].Value = text;
278             cmd.Parameters[1].Value = mindMap.Id;
279 
280             //Execute sql statement
281             cmd.Prepare();
282             result = cmd.ExecuteNonQuery() == 1;
283 
284             //Close database
285             DB.GetConnection().Close();
286             return result;
287         }
288         public static bool UpdateNode(string text, Node previous, Node node)
289         {
290             if ((previous == null && node.PreviousId != null) || ↩
                    (node.PreviousId == null && previous != null))
291             {
292                 return false;
293             }
294             //Initialize variables
295             OleDbCommand cmd;
296             bool result;
297 
298             //Open database, build sql statement and prepare
299             DB.GetConnection().Open();
300             cmd = DB.GetConnection().CreateCommand();
301             cmd.CommandText = "UPDATE TNodes SET NameNode = ?, PreviousIdNode = ↩
                    ? WHERE IdNode = ?;";
302             cmd.Parameters.Add("NameNode", OleDbType.VarChar, 255);
303             cmd.Parameters.Add("PreviousIdNode", OleDbType.Integer);
304             cmd.Parameters.Add("IdNode", OleDbType.Integer);
305             cmd.Parameters[0].Value = text;
306             if (previous == null)
307             {
308                 cmd.Parameters[1].Value = DBNull.Value;
309             }
310             else
311             {
312                 cmd.Parameters[1].Value = previous.Id;
313             }
314             cmd.Parameters[2].Value = node.Id;
315 
316             //Execute sql statement
317             cmd.Prepare();
318             result = cmd.ExecuteNonQuery() == 1;
319 
320             //Close database
321             DB.GetConnection().Close();
322             return result;
323         }
324 
```

```
325         #endregion
326         #region ADD
327         //All the methods send back the objects after getting back the created id (except linking methods)
328         public static Project CreateProject(string aName, string aDesc, DateTime aDate)
329         {
330             //Initialize variables
331             OleDbCommand cmd;
332             int id;
333 
334             //Open database, build sql statement and prepare
335             DB.GetConnection().Open();
336             cmd = DB.GetConnection().CreateCommand();
337             cmd.CommandText = "INSERT INTO TProjects (NameProject,Description,DateBegin) VALUES (?,?,?);";
338             cmd.Parameters.Add("NameProject", OleDbType.VarChar, 50);
339             cmd.Parameters.Add("Description", OleDbType.LongVarChar, 65535);
340             cmd.Parameters.Add("DateBegin", OleDbType.Date);
341             cmd.Parameters[0].Value = aName;
342             cmd.Parameters[1].Value = aDesc;
343             cmd.Parameters[2].Value = aDate;
344 
345             //Execute sql statement
346             cmd.Prepare();
347             cmd.ExecuteNonQuery();
348 
349             //Get last inserted id
350             cmd.CommandText = "SELECT @@Identity;";
351             id = (int)cmd.ExecuteScalar();
352 
353             //Close database
354             DB.GetConnection().Close();
355 
356             //return created project
357             Project project = new Project(id, aName, aDesc, aDate);
358             return project;
359         }
360         public static ChecklistItem CreateCheckListItem(string aName, Checklist checklist)
361         {
362             //Initialize variables
363             OleDbCommand cmd;
364             int id;
365 
366             //Open database, build sql statement and prepare
367             DB.GetConnection().Open();
368             cmd = DB.GetConnection().CreateCommand();
369             cmd.CommandText = "INSERT INTO TChecklistItems (NameItem,Done,IdChecklist) VALUES (?,?,?);";
370             cmd.Parameters.Add("NameItem", OleDbType.VarChar, 255);
371             cmd.Parameters.Add("Done", OleDbType.Boolean);
372             cmd.Parameters.Add("IdChecklist", OleDbType.Integer);
373             cmd.Parameters[0].Value = aName;
374             cmd.Parameters[1].Value = false;
375             cmd.Parameters[2].Value = checklist.Id;
376 
377             //Execute sql statement
378             cmd.Prepare();
379             cmd.ExecuteNonQuery();
380 
381             //Get last inserted id
382             cmd.CommandText = "SELECT @@Identity;";
383             id = (int)cmd.ExecuteScalar();
384 
385             //Close database
386             DB.GetConnection().Close();
387 
388             //return created project
389             ChecklistItem checklistItem = new ChecklistItem(id, aName, false, checklist.Id);
390             return checklistItem;
391         }
392         public static Checklist CreateCheckList(string aName, UserStory userStory)
393         {
394             //Initialize variables
395             OleDbCommand cmd;
396             int id;
397 
```

```
398             //Open database, build sql statement and prepare
399             DB.GetConnection().Open();
400             cmd = DB.GetConnection().CreateCommand();
401             cmd.CommandText = "INSERT␣INTO␣TChecklists␣↩
                    (NameChecklist,IdUserStory)␣VALUES␣(?,?);";
402             cmd.Parameters.Add("NameChecklist", OleDbType.VarChar, 255);
403             cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
404             cmd.Parameters[0].Value = aName;
405             cmd.Parameters[1].Value = userStory.Id;
406 
407             //Execute sql statement
408             cmd.Prepare();
409             cmd.ExecuteNonQuery();
410 
411             //Get last inserted id
412             cmd.CommandText = "SELECT␣@@Identity;";
413             id = (int)cmd.ExecuteScalar();
414 
415             //Close database
416             DB.GetConnection().Close();
417 
418             //return created project
419             Checklist checklist = new Checklist(id, aName, userStory.Id);
420             return checklist;
421         }
422         public static Activity CreateActivity(string description, DateTime now, ↩
                UserStory userStory)
423         {
424             //Initialize variables
425             OleDbCommand cmd;
426             int id;
427 
428             //Open database, build sql statement and prepare
429             DB.GetConnection().Open();
430             cmd = DB.GetConnection().CreateCommand();
431             cmd.CommandText = "INSERT␣INTO␣TActivities␣↩
                    (Description,DateTimeActivity,IdUserStory)␣VALUES␣(?,?,?);";
432             cmd.Parameters.Add("Description", OleDbType.LongVarChar, 65535);
433             cmd.Parameters.Add("DateTimeActivity", OleDbType.Date);
434             cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
435             cmd.Parameters[0].Value = description;
436             cmd.Parameters[1].Value = now;
437             cmd.Parameters[2].Value = userStory.Id;
438 
439             //Execute sql statement
440             cmd.Prepare();
441             cmd.ExecuteNonQuery();
442 
443             //Get last inserted id
444             cmd.CommandText = "SELECT␣@@Identity;";
445             id = (int)cmd.ExecuteScalar();
446 
447             //Close database
448             DB.GetConnection().Close();
449 
450             //return created project
451             Activity activity = new Activity(id, description, now, userStory.Id);
452             return activity;
453         }
454         public static UserStory CreateUserStory(string description, DateTime? ↩
                selectedDate, int complexity, Priority priority, Classes.Type type, ↩
                State state, Project project)
455         {
456             //Initialize variables
457             OleDbCommand cmd;
458             int id;
459 
460             //Open database, build sql statement and prepare
461             DB.GetConnection().Open();
462             cmd = DB.GetConnection().CreateCommand();
463             cmd.CommandText = "INSERT␣INTO␣TUserStories␣(DescriptionUserStory,␣↩
                    DateLimite,␣ComplexityEstimation,␣CompletedComplexity,␣Blocked,␣↩
                    IdProject,␣IdState,␣IdType,␣IdPriority)␣VALUES␣(?,␣?,␣?,␣?,␣?,␣↩
                    ?,␣?,␣?,␣?);";
464             cmd.Parameters.Add("DescriptionUserStory", OleDbType.LongVarChar, ↩
                    65535);
465             cmd.Parameters.Add("DateLimite", OleDbType.Date);
466             cmd.Parameters.Add("ComplexityEstimation", OleDbType.Integer);
467             cmd.Parameters.Add("CompletedComplexity", OleDbType.Integer);
```

```
468             cmd.Parameters.Add("Blocked", OleDbType.Boolean);
469             cmd.Parameters.Add("IdProject", OleDbType.Integer);
470             cmd.Parameters.Add("IdState", OleDbType.Integer);
471             cmd.Parameters.Add("IdType", OleDbType.Integer);
472             cmd.Parameters.Add("IdPriority", OleDbType.Integer);
473             cmd.Parameters[0].Value = description;
474             if (selectedDate == null)
475             {
476                 cmd.Parameters[1].Value = DBNull.Value;
477             }
478             else
479             {
480                 cmd.Parameters[1].Value = selectedDate;
481             }
482             cmd.Parameters[2].Value = complexity;
483             cmd.Parameters[3].Value = 0;
484             cmd.Parameters[4].Value = false;
485             cmd.Parameters[5].Value = project.Id;
486             cmd.Parameters[6].Value = state.Id;
487             cmd.Parameters[7].Value = type.Id;
488             cmd.Parameters[8].Value = priority.Id;
489 
490             //Execute sql statement
491             cmd.Prepare();
492             cmd.ExecuteNonQuery();
493 
494             //Get last inserted id
495             cmd.CommandText = "SELECT␣@@Identity;";
496             id = (int)cmd.ExecuteScalar();
497 
498             //Close database
499             DB.GetConnection().Close();
500 
501             //return created project
502             UserStory userStory = new UserStory(id, description, selectedDate, ↩
                    complexity, 0, false, project.Id, state.Id, type.Id, priority.Id);
503             return userStory;
504         }
505         public static State CreateState(string name)
506         {
507             //Initialize variables
508             OleDbCommand cmd;
509             int id;
510 
511             //Open database, build sql statement and prepare
512             DB.GetConnection().Open();
513             cmd = DB.GetConnection().CreateCommand();
514             cmd.CommandText = "INSERT␣INTO␣TStates␣(NameState)␣VALUES␣(?);";
515             cmd.Parameters.Add("NameState", OleDbType.VarChar, 30);
516             cmd.Parameters[0].Value = name;
517 
518             //Execute sql statement
519             cmd.Prepare();
520             cmd.ExecuteNonQuery();
521 
522             //Get last inserted id
523             cmd.CommandText = "SELECT␣@@Identity;";
524             id = (int)cmd.ExecuteScalar();
525 
526             //Close database
527             DB.GetConnection().Close();
528 
529             //return created project
530             State state = new State(id, name);
531             return state;
532         }
533         public static Classes.File CreateFile(string fileName, string ↩
                description, UserStory userStory)
534         {
535             //Initialize variables
536             OleDbCommand cmd;
537             int id;
538 
539             //Open database, build sql statement and prepare
540             DB.GetConnection().Open();
541             cmd = DB.GetConnection().CreateCommand();
542             cmd.CommandText = "INSERT␣INTO␣TFiles␣↩
                    (NameFile,DescriptionFile,IdUserStory)␣VALUES␣(?,?,?);";
543             cmd.Parameters.Add("NameState", OleDbType.LongVarChar, 65535);
```

```
544             cmd.Parameters.Add("DescriptionFile", OleDbType.LongVarChar, 65535);
545             cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
546             cmd.Parameters[0].Value = fileName;
547             cmd.Parameters[1].Value = description;
548             cmd.Parameters[2].Value = userStory.Id;
549 
550             //Execute sql statement
551             cmd.Prepare();
552             cmd.ExecuteNonQuery();
553 
554             //Get last inserted id
555             cmd.CommandText = "SELECT␣@@Identity;";
556             id = (int)cmd.ExecuteScalar();
557 
558             //Close database
559             DB.GetConnection().Close();
560 
561             //return created project
562             Classes.File file = new Classes.File(id, fileName, description, ↩
                    userStory.Id);
563             return file;
564         }
565         public static User CreateUser(string name)
566         {
567             //Initialize variables
568             OleDbCommand cmd;
569             int id;
570 
571             //Open database, build sql statement and prepare
572             DB.GetConnection().Open();
573             cmd = DB.GetConnection().CreateCommand();
574             cmd.CommandText = "INSERT␣INTO␣TUsers␣(NameUser)␣VALUES␣(?);";
575             cmd.Parameters.Add("NameUser", OleDbType.VarChar, 255);
576             cmd.Parameters[0].Value = name;
577 
578             //Execute sql statement
579             cmd.Prepare();
580             cmd.ExecuteNonQuery();
581 
582             //Get last inserted id
583             cmd.CommandText = "SELECT␣@@Identity;";
584             id = (int)cmd.ExecuteScalar();
585 
586             //Close database
587             DB.GetConnection().Close();
588 
589             //return created project
590             User user = new User(id, name);
591             return user;
592         }
593         public static Comment CreateComment(string name, UserStory userStory, ↩
                User user)
594         {
595             //Initialize variables
596             OleDbCommand cmd;
597             int id;
598             DateTime dateTime = DateTime.Now;
599 
600             //Open database, build sql statement and prepare
601             DB.GetConnection().Open();
602             cmd = DB.GetConnection().CreateCommand();
603             cmd.CommandText = "INSERT␣INTO␣TComments␣↩
                    (DescriptionComment,CreationDateTime,IdUserStory,IdUser)␣VALUES␣↩
                    (?,?,?,?);";
604             cmd.Parameters.Add("DescriptionComment", OleDbType.LongVarChar, ↩
                    65535);
605             cmd.Parameters.Add("DateTime", OleDbType.Date);
606             cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
607             cmd.Parameters.Add("IdUser", OleDbType.Integer);
608             cmd.Parameters[0].Value = name;
609             cmd.Parameters[1].Value = dateTime;
610             cmd.Parameters[2].Value = userStory.Id;
611             cmd.Parameters[3].Value = user.Id;
612 
613             //Execute sql statement
614             cmd.Prepare();
615             cmd.ExecuteNonQuery();
616 
617             //Get last inserted id
```

```
618             cmd.CommandText = "SELECT␣@@Identity;";
619             id = (int)cmd.ExecuteScalar();
620 
621             //Close database
622             DB.GetConnection().Close();
623 
624             Comment comment = new Comment(id, name, dateTime, userStory.Id, ↩
                    user.Id);
625             return comment;
626         }
627         public static Sprint CreateSprint(DateTime dateBegin, DateTime dateEnd, ↩
                Project project)
628         {
629             //Initialize variables
630             OleDbCommand cmd;
631             int id;
632 
633             //Open database, build sql statement and prepare
634             DB.GetConnection().Open();
635             cmd = DB.GetConnection().CreateCommand();
636             cmd.CommandText = "INSERT␣INTO␣TSprints␣↩
                    (DateBegin,DateEnd,IdProject)␣VALUES␣(?,?,?);";
637             cmd.Parameters.Add("DateBegin", OleDbType.Date);
638             cmd.Parameters.Add("DateEnd", OleDbType.Date);
639             cmd.Parameters.Add("IdProject", OleDbType.Integer);
640             cmd.Parameters[0].Value = dateBegin;
641             cmd.Parameters[1].Value = dateEnd;
642             cmd.Parameters[2].Value = project.Id;
643 
644             //Execute sql statement
645             cmd.Prepare();
646             cmd.ExecuteNonQuery();
647 
648             //Get last inserted id
649             cmd.CommandText = "SELECT␣@@Identity;";
650             id = (int)cmd.ExecuteScalar();
651 
652             //Close database
653             DB.GetConnection().Close();
654 
655             Sprint sprint = new Sprint(id, dateBegin, dateEnd, project.Id);
656             return sprint;
657 
658         }
659         public static MindMap CreateMindmap(string name, Project project)
660         {
661             //Initialize variables
662             OleDbCommand cmd;
663             int id;
664 
665             //Open database, build sql statement and prepare
666             DB.GetConnection().Open();
667             cmd = DB.GetConnection().CreateCommand();
668             cmd.CommandText = "INSERT␣INTO␣TMindMaps␣(NameMindMap,IdProject)␣↩
                    VALUES␣(?,?);";
669             cmd.Parameters.Add("NameMindMap", OleDbType.VarChar, 255);
670             cmd.Parameters.Add("IdProject", OleDbType.Integer);
671             cmd.Parameters[0].Value = name;
672             cmd.Parameters[1].Value = project.Id;
673 
674             //Execute sql statement
675             cmd.Prepare();
676             cmd.ExecuteNonQuery();
677 
678             //Get last inserted id
679             cmd.CommandText = "SELECT␣@@Identity;";
680             id = (int)cmd.ExecuteScalar();
681 
682             //Close database
683             DB.GetConnection().Close();
684 
685             MindMap mindMap = new MindMap(id, name, project.Id);
686             return mindMap;
687         }
688         public static Node CreateNode(string name, Node previous, MindMap mindMap)
689         {
690             //Initialize variables
691             OleDbCommand cmd;
692             int id;
```

```
693
694             //Open database, build sql statement and prepare
695             DB.GetConnection().Open();
696             cmd = DB.GetConnection().CreateCommand();
697             cmd.CommandText = "INSERT␣INTO␣TNodes␣↩
                    (NameNode,PreviousIdNode,IdMindMap)␣VALUES␣(?,?,?);";
698             cmd.Parameters.Add("NameNode", OleDbType.VarChar, 255);
699             cmd.Parameters.Add("PreviousIdNode", OleDbType.Integer);
700             cmd.Parameters.Add("IdMindMap", OleDbType.Integer);
701             cmd.Parameters[0].Value = name;
702             if (previous == null)
703             {
704                 cmd.Parameters[1].Value = DBNull.Value;
705             }
706             else
707             {
708                 cmd.Parameters[1].Value = previous.Id;
709             }
710             cmd.Parameters[2].Value = mindMap.Id;
711
712             //Execute sql statement
713             cmd.Prepare();
714             cmd.ExecuteNonQuery();
715
716             //Get last inserted id
717             cmd.CommandText = "SELECT␣@@Identity;";
718             id = (int)cmd.ExecuteScalar();
719
720             //Close database
721             DB.GetConnection().Close();
722
723             int? previousId = null;
724             if (previous != null)
725             {
726                 previousId = previous.Id;
727             }
728             Node node = new Node(id, name, previousId, mindMap.Id);
729             return node;
730         }
731         public static bool AddStateToProject(State state, Project project, int ↩
                order)
732         {
733             //Initialize variables
734             OleDbCommand cmd;
735             bool result;
736
737             //Open database, build sql statement and prepare
738             DB.GetConnection().Open();
739             cmd = DB.GetConnection().CreateCommand();
740             cmd.CommandText = "INSERT␣INTO␣TProjectStates␣↩
                    (IdProject,IdState,orderState)␣VALUES␣(?,?,?)␣;";
741             cmd.Parameters.Add("IdProject", OleDbType.Integer);
742             cmd.Parameters.Add("IdState", OleDbType.Integer);
743             cmd.Parameters.Add("orderState", OleDbType.Integer);
744             cmd.Parameters[0].Value = project.Id;
745             cmd.Parameters[1].Value = state.Id;
746             cmd.Parameters[2].Value = order;
747
748             //Execute sql statement
749             cmd.Prepare();
750             result = cmd.ExecuteNonQuery() == 1;
751
752             //Close database
753             DB.GetConnection().Close();
754             return result;
755         }
756         public static bool AddUserToChecklistItem(User user, ChecklistItem ↩
                checklistItem)
757         {
758             //Initialize variables
759             OleDbCommand cmd;
760             bool result;
761
762             //Open database, build sql statement and prepare
763             DB.GetConnection().Open();
764             cmd = DB.GetConnection().CreateCommand();
765             cmd.CommandText = "INSERT␣INTO␣TUserChecklistItem␣↩
                    (IdUser,IdChecklistItem)␣VALUES␣(?,?);";
766             cmd.Parameters.Add("IdUser", OleDbType.Integer);
```

```
767             cmd.Parameters.Add("IdProject", OleDbType.Integer);
768             cmd.Parameters[0].Value = user.Id;
769             cmd.Parameters[1].Value = checklistItem.Id;
770 
771             //Execute sql statement
772             cmd.Prepare();
773             result = cmd.ExecuteNonQuery() == 1;
774 
775             //Close database
776             DB.GetConnection().Close();
777             return result;
778         }
779         public static bool AddUserToUserStory(User user, UserStory userStory)
780         {
781 
782             //Initialize variables
783             OleDbCommand cmd;
784             bool result;
785 
786             //Open database, build sql statement and prepare
787             DB.GetConnection().Open();
788             cmd = DB.GetConnection().CreateCommand();
789             cmd.CommandText = "INSERT␣INTO␣TUserUserStory(IdUser,IdUserStory)␣↩
                    VALUES␣(?,?);";
790             cmd.Parameters.Add("IdUser", OleDbType.Integer);
791             cmd.Parameters.Add("IdProject", OleDbType.Integer);
792             cmd.Parameters[0].Value = user.Id;
793             cmd.Parameters[1].Value = userStory.Id;
794 
795             //Execute sql statement
796             cmd.Prepare();
797             result = cmd.ExecuteNonQuery() == 1;
798 
799             //Close database
800             DB.GetConnection().Close();
801             return result;
802         }
803         public static bool AddUserToProject(User user, Project project)
804         {
805             //Initialize variables
806             OleDbCommand cmd;
807             bool result;
808 
809             //Open database, build sql statement and prepare
810             DB.GetConnection().Open();
811             cmd = DB.GetConnection().CreateCommand();
812             cmd.CommandText = "INSERT␣INTO␣TUserProject␣(IdUser,IdProject)␣↩
                    VALUES␣(?,?);";
813             cmd.Parameters.Add("IdUser", OleDbType.Integer);
814             cmd.Parameters.Add("IdProject", OleDbType.Integer);
815             cmd.Parameters[0].Value = user.Id;
816             cmd.Parameters[1].Value = project.Id;
817 
818             //Execute sql statement
819             cmd.Prepare();
820             result = cmd.ExecuteNonQuery() == 1;
821 
822             //Close database
823             DB.GetConnection().Close();
824             return result;
825         }
826         public static bool AddUserStoryToSprint(UserStory userStory, Sprint ↩
                sprint, int order)
827         {
828             //Initialize variables
829             OleDbCommand cmd;
830             bool result;
831 
832             //Open database, build sql statement and prepare
833             DB.GetConnection().Open();
834             cmd = DB.GetConnection().CreateCommand();
835             cmd.CommandText = "INSERT␣INTO␣TUserStoriesSprint␣(IdUserStory␣,␣↩
                    IdSprint␣,␣OrderUserStory)␣VALUES␣(?␣,␣?␣,␣?)␣;";
836 
837             cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
838             cmd.Parameters.Add("IdSprint", OleDbType.Integer);
839             cmd.Parameters.Add("OrderUserStory", OleDbType.Integer);
840 
841             cmd.Parameters[0].Value = userStory.Id;
```

```
842             cmd.Parameters[1].Value = sprint.Id;
843             cmd.Parameters[2].Value = order;
844 
845             //Execute sql statement
846             cmd.Prepare();
847             result = cmd.ExecuteNonQuery() == 1;
848 
849             //Close database
850             DB.GetConnection().Close();
851             return result;
852         }
853 
854         #endregion
855         #region REMOVE
856         //Set the deletion flag to true
857         public static bool Delete(Activity activity)
858         {
859             //Initialize variables
860             OleDbCommand cmd;
861             bool result;
862 
863             //Open database, build sql statement and prepare
864             DB.GetConnection().Open();
865             cmd = DB.GetConnection().CreateCommand();
866             cmd.CommandText = "UPDATE␣TActivities␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                IdActivity␣=␣?;";
867             cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
868             cmd.Parameters.Add("IdActivity", OleDbType.Integer);
869             cmd.Parameters[0].Value = true;
870             cmd.Parameters[1].Value = activity.Id;
871 
872             //Execute sql statement
873             cmd.Prepare();
874             result = cmd.ExecuteNonQuery() == 1;
875 
876             //Close database
877             DB.GetConnection().Close();
878             return result;
879         }
880         public static bool Delete(ChecklistItem checklistItem)
881         {
882             //Initialize variables
883             OleDbCommand cmd;
884             bool result;
885 
886             //Open database, build sql statement and prepare
887             DB.GetConnection().Open();
888             cmd = DB.GetConnection().CreateCommand();
889             cmd.CommandText = "UPDATE␣TChecklistItems␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                IdChecklistItem␣=␣?;";
890             cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
891             cmd.Parameters.Add("IdChecklistItem", OleDbType.Integer);
892             cmd.Parameters[0].Value = true;
893             cmd.Parameters[1].Value = checklistItem.Id;
894 
895             //Execute sql statement
896             cmd.Prepare();
897             result = cmd.ExecuteNonQuery() == 1;
898 
899             //Close database
900             DB.GetConnection().Close();
901             return result;
902         }
903         public static bool Delete(Checklist checklist)
904         {
905             //Initialize variables
906             OleDbCommand cmd;
907             bool result;
908 
909             //Open database, build sql statement and prepare
910             DB.GetConnection().Open();
911             cmd = DB.GetConnection().CreateCommand();
912             cmd.CommandText = "UPDATE␣TChecklists␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                IdChecklist␣=␣?;";
913             cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
914             cmd.Parameters.Add("IdChecklist", OleDbType.Integer);
915             cmd.Parameters[0].Value = true;
916             cmd.Parameters[1].Value = checklist.Id;
917 
```

```
            //Execute sql statement
            cmd.Prepare();
            result = cmd.ExecuteNonQuery() == 1;

            //Close database
            DB.GetConnection().Close();
            return result;
        }
        public static bool Delete(Comment comment)
        {
            //Initialize variables
            OleDbCommand cmd;
            bool result;

            //Open database, build sql statement and prepare
            DB.GetConnection().Open();
            cmd = DB.GetConnection().CreateCommand();
            cmd.CommandText = "UPDATE TComments SET deletedFlag = ? WHERE IdComment = ?;";
            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
            cmd.Parameters.Add("IdComment", OleDbType.Integer);
            cmd.Parameters[0].Value = true;
            cmd.Parameters[1].Value = comment.Id;

            //Execute sql statement
            cmd.Prepare();
            result = cmd.ExecuteNonQuery() == 1;

            //Close database
            DB.GetConnection().Close();
            return result;
        }
        public static bool Delete(Classes.File File)
        {
            //Initialize variables
            OleDbCommand cmd;
            bool result;

            //Open database, build sql statement and prepare
            DB.GetConnection().Open();
            cmd = DB.GetConnection().CreateCommand();
            cmd.CommandText = "UPDATE TFiles SET deletedFlag = ? WHERE IdFile = ?;";
            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
            cmd.Parameters.Add("IdFile", OleDbType.Integer);
            cmd.Parameters[0].Value = true;
            cmd.Parameters[1].Value = File.Id;

            //Execute sql statement
            cmd.Prepare();
            result = cmd.ExecuteNonQuery() == 1;

            //Close database
            DB.GetConnection().Close();
            return result;
        }
        public static bool Delete(Project project)
        {
            //Initialize variables
            OleDbCommand cmd;
            bool result;

            //Open database, build sql statement and prepare
            DB.GetConnection().Open();
            cmd = DB.GetConnection().CreateCommand();
            cmd.CommandText = "UPDATE TProjects SET deletedFlag = ? WHERE IdProject = ?;";
            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
            cmd.Parameters.Add("IdProject", OleDbType.Integer);
            cmd.Parameters[0].Value = true;
            cmd.Parameters[1].Value = project.Id;

            //Execute sql statement
            cmd.Prepare();
            result = cmd.ExecuteNonQuery() == 1;

            //Close database
            DB.GetConnection().Close();
            return result;
```

```
994         }
995         public static bool Delete(Sprint sprint)
996         {
997             //Initialize variables
998             OleDbCommand cmd;
999             bool result;
1000
1001            //Open database, build sql statement and prepare
1002            DB.GetConnection().Open();
1003            cmd = DB.GetConnection().CreateCommand();
1004            cmd.CommandText = "UPDATE␣TSprints␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                   IdSprint␣=␣?;";
1005            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1006            cmd.Parameters.Add("IdSprint", OleDbType.Integer);
1007            cmd.Parameters[0].Value = true;
1008            cmd.Parameters[1].Value = sprint.Id;
1009
1010            //Execute sql statement
1011            cmd.Prepare();
1012            result = cmd.ExecuteNonQuery() == 1;
1013
1014            //Close database
1015            DB.GetConnection().Close();
1016            return result;
1017        }
1018        public static bool Delete(State state)
1019        {
1020            //Initialize variables
1021            OleDbCommand cmd;
1022            bool result;
1023
1024            //Open database, build sql statement and prepare
1025            DB.GetConnection().Open();
1026            cmd = DB.GetConnection().CreateCommand();
1027            cmd.CommandText = "UPDATE␣TStates␣SET␣deletedFlag␣=␣?␣WHERE␣IdState␣↩
                   =␣?;";
1028            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1029            cmd.Parameters.Add("IdState", OleDbType.Integer);
1030            cmd.Parameters[0].Value = true;
1031            cmd.Parameters[1].Value = state.Id;
1032
1033            //Execute sql statement
1034            cmd.Prepare();
1035            result = cmd.ExecuteNonQuery() == 1;
1036
1037            //Close database
1038            DB.GetConnection().Close();
1039            return result;
1040        }
1041        public static bool Delete(User user)
1042        {
1043            //Initialize variables
1044            OleDbCommand cmd;
1045            bool result;
1046
1047            //Open database, build sql statement and prepare
1048            DB.GetConnection().Open();
1049            cmd = DB.GetConnection().CreateCommand();
1050            cmd.CommandText = "UPDATE␣TUsers␣SET␣deletedFlag␣=␣?␣WHERE␣IdUser␣=␣↩
                   ?;";
1051            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1052            cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
1053            cmd.Parameters[0].Value = true;
1054            cmd.Parameters[1].Value = user.Id;
1055
1056            //Execute sql statement
1057            cmd.Prepare();
1058            result = cmd.ExecuteNonQuery() == 1;
1059
1060            //Close database
1061            DB.GetConnection().Close();
1062            return result;
1063        }
1064        public static bool Delete(UserStory userStory)
1065        {
1066            //Initialize variables
1067            OleDbCommand cmd;
1068            bool result;
1069
```

```
1070            //Open database, build sql statement and prepare
1071            DB.GetConnection().Open();
1072            cmd = DB.GetConnection().CreateCommand();
1073            cmd.CommandText = "UPDATE␣TUserStories␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                    IdUserStory␣=␣?;";
1074            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1075            cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
1076            cmd.Parameters[0].Value = true;
1077            cmd.Parameters[1].Value = userStory.Id;
1078
1079            //Execute sql statement
1080            cmd.Prepare();
1081            result = cmd.ExecuteNonQuery() == 1;
1082
1083            //Close database
1084            DB.GetConnection().Close();
1085            return result;
1086        }
1087        public static bool RemoveUserFromUserStory(User user, UserStory userStory)
1088        {
1089            //Initialize variables
1090            OleDbCommand cmd;
1091            bool result;
1092
1093            //Open database, build sql statement and prepare
1094            DB.GetConnection().Open();
1095            cmd = DB.GetConnection().CreateCommand();
1096            cmd.CommandText = "DELETE␣FROM␣TUserUserStory␣WHERE␣IdUser␣=␣?␣AND␣↩
                    IdUserStory␣=␣?;";
1097            cmd.Parameters.Add("IdUser", OleDbType.Integer);
1098            cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
1099            cmd.Parameters[0].Value = user.Id;
1100            cmd.Parameters[1].Value = userStory.Id;
1101
1102            //Execute sql statement
1103            cmd.Prepare();
1104            result = cmd.ExecuteNonQuery() == 1;
1105
1106            //Close database
1107            DB.GetConnection().Close();
1108            return result;
1109        }
1110        public static bool RemoveUserFromProject(User user, Project project)
1111        {
1112            //Initialize variables
1113            OleDbCommand cmd;
1114            bool result;
1115
1116            //Open database, build sql statement and prepare
1117            DB.GetConnection().Open();
1118            cmd = DB.GetConnection().CreateCommand();
1119            cmd.CommandText = "DELETE␣FROM␣TUserProject␣WHERE␣IdUser␣=␣?␣AND␣↩
                    IdProject␣=␣?;";
1120            cmd.Parameters.Add("IdUser", OleDbType.Integer);
1121            cmd.Parameters.Add("IdProject", OleDbType.Integer);
1122            cmd.Parameters[0].Value = user.Id;
1123            cmd.Parameters[1].Value = project.Id;
1124
1125            //Execute sql statement
1126            cmd.Prepare();
1127            result = cmd.ExecuteNonQuery() == 1;
1128
1129            //Close database
1130            DB.GetConnection().Close();
1131            return result;
1132        }
1133        public static bool RemoveUserFromChecklistItem(User user, ChecklistItem ↩
                checklistItem)
1134        {
1135            //Initialize variables
1136            OleDbCommand cmd;
1137            bool result;
1138
1139            //Open database, build sql statement and prepare
1140            DB.GetConnection().Open();
1141            cmd = DB.GetConnection().CreateCommand();
1142            cmd.CommandText = "DELETE␣FROM␣TUserChecklistItem␣WHERE␣IdUser␣=␣?␣↩
                    AND␣IdChecklistItem␣=␣?;";
1143            cmd.Parameters.Add("IdUser", OleDbType.Integer);
```

```
1144            cmd.Parameters.Add("IdChecklistItem", OleDbType.Integer);
1145            cmd.Parameters[0].Value = user.Id;
1146            cmd.Parameters[1].Value = checklistItem.Id;
1147
1148            //Execute sql statement
1149            cmd.Prepare();
1150            result = cmd.ExecuteNonQuery() == 1;
1151
1152            //Close database
1153            DB.GetConnection().Close();
1154            return result;
1155        }
1156        public static bool RemoveUserStoryFromSprint(UserStory userStory, ↩
                Sprint sprint, int order)
1157        {
1158            //Initialize variables
1159            OleDbCommand cmd;
1160            bool result;
1161
1162            //Open database, build sql statement and prepare
1163            DB.GetConnection().Open();
1164            cmd = DB.GetConnection().CreateCommand();
1165            cmd.CommandText = "DELETE␣FROM␣TUserStoriesSprint␣WHERE␣IdUserStory␣↩
                =␣?␣AND␣IdSprint␣=␣?␣AND␣OrderUserStory␣=␣?␣;";
1166
1167            cmd.Parameters.Add("IdUserStory", OleDbType.Integer);
1168            cmd.Parameters.Add("IdSprint", OleDbType.Integer);
1169            cmd.Parameters.Add("OrderUserStory", OleDbType.Integer);
1170
1171            cmd.Parameters[0].Value = userStory.Id;
1172            cmd.Parameters[1].Value = sprint.Id;
1173            cmd.Parameters[2].Value = order;
1174
1175            //Execute sql statement
1176            cmd.Prepare();
1177            result = cmd.ExecuteNonQuery() == 1;
1178
1179            //Close database
1180            DB.GetConnection().Close();
1181            return result;
1182        }
1183        public static bool RemoveStateFromProject(Project project, int order)
1184        {
1185            //Initialize variables
1186            OleDbCommand cmd;
1187            bool result;
1188
1189            //Open database, build sql statement and prepare
1190            DB.GetConnection().Open();
1191            cmd = DB.GetConnection().CreateCommand();
1192            cmd.CommandText = "DELETE␣FROM␣TProjectStates␣WHERE␣IdProject␣=␣?␣↩
                AND␣orderState␣=␣?;";
1193            cmd.Parameters.Add("IdProject", OleDbType.Integer);
1194            cmd.Parameters.Add("orderState", OleDbType.Integer);
1195            cmd.Parameters[0].Value = project.Id;
1196            cmd.Parameters[1].Value = order;
1197
1198            //Execute sql statement
1199            cmd.Prepare();
1200            result = cmd.ExecuteNonQuery() == 1;
1201
1202            //Close database
1203            DB.GetConnection().Close();
1204            return result;
1205        }
1206        public static bool Delete(MindMap mindmap)
1207        {
1208            //Initialize variables
1209            OleDbCommand cmd;
1210            bool result;
1211
1212            //Open database, build sql statement and prepare
1213            DB.GetConnection().Open();
1214            cmd = DB.GetConnection().CreateCommand();
1215            cmd.CommandText = "UPDATE␣TMindMaps␣SET␣deletedFlag␣=␣?␣WHERE␣↩
                IdMindMap␣=␣?;";
1216            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1217            cmd.Parameters.Add("IdMindMap", OleDbType.Integer);
1218            cmd.Parameters[0].Value = true;
```

```
1219            cmd.Parameters[1].Value = mindmap.Id;
1220
1221            //Execute sql statement
1222            cmd.Prepare();
1223            result = cmd.ExecuteNonQuery() == 1;
1224
1225            //Close database
1226            DB.GetConnection().Close();
1227            return result;
1228        }
1229        public static bool Delete(Node node)
1230        {
1231            if (node.PreviousId == null)
1232            {
1233                return false;
1234            }
1235            //Initialize variables
1236            OleDbCommand cmd;
1237            bool result;
1238
1239            //Open database, build sql statement and prepare
1240            DB.GetConnection().Open();
1241            cmd = DB.GetConnection().CreateCommand();
1242            cmd.CommandText = "UPDATE␣TNodes␣SET␣deletedFlag␣=␣?␣WHERE␣IdNode␣=␣↩
                ?;";
1243            cmd.Parameters.Add("deletedFlag", OleDbType.Boolean);
1244            cmd.Parameters.Add("IdNode", OleDbType.Integer);
1245            cmd.Parameters[0].Value = true;
1246            cmd.Parameters[1].Value = node.Id;
1247
1248            //Execute sql statement
1249            cmd.Prepare();
1250            result = cmd.ExecuteNonQuery() == 1;
1251
1252            //Close database
1253            DB.GetConnection().Close();
1254            return result;
1255        }
1256        #endregion
1257        #region GET
1258        //get objects with deletedflag on false
1259        public static List<int[]> GetUserStoriesSprint()
1260        {
1261            //Declare variables
1262            OleDbCommand cmd;
1263            OleDbDataReader rdr;
1264            object[] values;
1265            List<int[]> valuesPair = new List<int[]>();
1266
1267            //Open Database
1268            DB.GetConnection().Open();
1269            cmd = DB.GetConnection().CreateCommand();
1270
1271            //Execute SQL Command
1272            cmd.CommandText = "SELECT␣*␣FROM␣TUserStoriesSprint;";
1273
1274            //Read and assign states with project
1275            rdr = cmd.ExecuteReader();
1276            values = new object[3];
1277            while (rdr.Read())
1278            {
1279                rdr.GetValues(values);
1280                // 0:IdUserStory,1:IdSprint,2:Order
1281                valuesPair.Add(new int[] { (int)values[0], (int)values[1], ↩
                    (int)values[2] });
1282            }
1283
1284            //Close database and reader
1285            rdr.Close();
1286            DB.GetConnection().Close();
1287
1288            return valuesPair;
1289        }
1290        public static List<int[]> GetProjectStates()
1291        {
1292            //Declare variables
1293            OleDbCommand cmd;
1294            OleDbDataReader rdr;
1295            object[] values;
```

```
1296            List<int[]> valuesPair = new List<int[]>();
1297
1298            //Open Database
1299            DB.GetConnection().Open();
1300            cmd = DB.GetConnection().CreateCommand();
1301
1302            //Execute SQL Command
1303            cmd.CommandText = "SELECT␣*␣FROM␣TProjectStates;";
1304
1305            //Read and assign states with project
1306            rdr = cmd.ExecuteReader();
1307            values = new object[3];
1308            while (rdr.Read())
1309            {
1310                rdr.GetValues(values);
1311                // 0:IdProject,1:IdState,2:order
1312                valuesPair.Add(new int[] { (int)values[0], (int)values[1], ↩
                        (int)values[2] });
1313            }
1314
1315            //Close database and reader
1316            rdr.Close();
1317            DB.GetConnection().Close();
1318
1319            return valuesPair;
1320        }
1321        public static List<int[]> GetUserProject()
1322        {
1323            //Declare variables
1324            OleDbCommand cmd;
1325            OleDbDataReader rdr;
1326            object[] values;
1327            List<int[]> valuesPair = new List<int[]>();
1328
1329            //Open Database
1330            DB.GetConnection().Open();
1331            cmd = DB.GetConnection().CreateCommand();
1332
1333            //Execute SQL Command
1334            cmd.CommandText = "SELECT␣*␣FROM␣TUserProject;";
1335
1336            //Read and assign states with project
1337            rdr = cmd.ExecuteReader();
1338            values = new object[2];
1339            while (rdr.Read())
1340            {
1341                rdr.GetValues(values);
1342                // 0:IdUser,1:IdProject
1343                valuesPair.Add(new int[] { (int)values[0], (int)values[1] });
1344            }
1345
1346            //Close database and reader
1347            rdr.Close();
1348            DB.GetConnection().Close();
1349
1350            return valuesPair;
1351        }
1352        public static List<int[]> GetUserChecklistItem()
1353        {
1354            //Declare variables
1355            OleDbCommand cmd;
1356            OleDbDataReader rdr;
1357            object[] values;
1358            List<int[]> valuesPair = new List<int[]>();
1359
1360            //Open Database
1361            DB.GetConnection().Open();
1362            cmd = DB.GetConnection().CreateCommand();
1363
1364            //Execute SQL Command
1365            cmd.CommandText = "SELECT␣*␣FROM␣TUserChecklistItem;";
1366
1367            //Read and assign states with project
1368            rdr = cmd.ExecuteReader();
1369            values = new object[2];
1370            while (rdr.Read())
1371            {
1372                rdr.GetValues(values);
1373                // 0:IdUser,1:IdChecklistItem
```

```
1374                valuesPair.Add(new int[] { (int)values[0], (int)values[1] });
1375            }
1376
1377            //Close database and reader
1378            rdr.Close();
1379            DB.GetConnection().Close();
1380
1381            return valuesPair;
1382        }
1383        public static List<int[]> GetUserUserStory()
1384        {
1385            //Declare variables
1386            OleDbCommand cmd;
1387            OleDbDataReader rdr;
1388            object[] values;
1389            List<int[]> valuesPair = new List<int[]>();
1390
1391            //Open Database
1392            DB.GetConnection().Open();
1393            cmd = DB.GetConnection().CreateCommand();
1394
1395            //Execute SQL Command
1396            cmd.CommandText = "SELECT␣*␣FROM␣TUserUserStory;";
1397
1398            //Read and assign states with project
1399            rdr = cmd.ExecuteReader();
1400            values = new object[2];
1401            while (rdr.Read())
1402            {
1403                rdr.GetValues(values);
1404                // 0:IdUser,1:IdUserStory
1405                valuesPair.Add(new int[] { (int)values[0], (int)values[1] });
1406            }
1407
1408            //Close database and reader
1409            rdr.Close();
1410            DB.GetConnection().Close();
1411
1412            return valuesPair;
1413        }
1414        public static List<Project> GetProjects()
1415        {
1416            //Declare variables
1417            OleDbCommand cmd;
1418            OleDbDataReader rdr;
1419            object[] values;
1420            List<Project> projects = new List<Project>();
1421
1422            //Open Database
1423            DB.GetConnection().Open();
1424            cmd = DB.GetConnection().CreateCommand();
1425
1426            //Execute SQL Command
1427            cmd.CommandText = "SELECT␣*␣FROM␣TProjects␣WHERE␣deletedFlag␣=␣↩
                FALSE;";
1428            cmd.Connection = DB.GetConnection();
1429
1430            //Read and put projects in a list
1431            rdr = cmd.ExecuteReader();
1432            values = new object[4];
1433            while (rdr.Read())
1434            {
1435                rdr.GetValues(values);
1436                //0:IdProject,1:NameProject,2:Description,3:DateBegin
1437                Project p = new Project((int)values[0], (string)values[1], ↩
                    (string)values[2], (DateTime)values[3]);
1438                projects.Add(p);
1439            }
1440
1441            //Close database and reader
1442            rdr.Close();
1443            GetConnection().Close();
1444
1445            return projects;
1446        }
1447        public static List<Sprint> GetSprints()
1448        {
1449            //Declare variables
1450            OleDbCommand cmd;
```

```
1451            OleDbDataReader rdr;
1452            object[] values;
1453            List<Sprint> sprints = new List<Sprint>();
1454
1455            //Open Database
1456            DB.GetConnection().Open();
1457            cmd = DB.GetConnection().CreateCommand();
1458
1459            //Execute SQL Command
1460            cmd.CommandText = "SELECT␣*␣FROM␣TSprints␣WHERE␣deletedFlag␣=␣FALSE;";
1461            cmd.Connection = DB.GetConnection();
1462
1463            //Read and put entries in a list of objects
1464            rdr = cmd.ExecuteReader();
1465            values = new object[4];
1466            while (rdr.Read())
1467            {
1468                rdr.GetValues(values);
1469                // 0:idSprint,1:DateBegin,2:DateEnd
1470                Sprint s = new Sprint((int)values[0], (DateTime)values[1], ↩
                        (DateTime)values[2], (int)values[3]);
1471                sprints.Add(s);
1472            }
1473
1474            //Close database and reader
1475            rdr.Close();
1476            GetConnection().Close();
1477
1478            return sprints;
1479        }
1480        public static List<UserStory> GetUserStories()
1481        {
1482            //Declare variables
1483            OleDbCommand cmd;
1484            OleDbDataReader rdr;
1485            object[] values;
1486            List<UserStory> userStories = new List<UserStory>();
1487
1488            //Open Database
1489            DB.GetConnection().Open();
1490            cmd = DB.GetConnection().CreateCommand();
1491
1492            //Execute SQL Command
1493            cmd.CommandText = "SELECT␣*␣FROM␣TUserStories␣WHERE␣deletedFlag␣=␣↩
                FALSE;";
1494            cmd.Connection = DB.GetConnection();
1495
1496            //Read and put entries in a list of objects
1497            rdr = cmd.ExecuteReader();
1498            values = new object[10];
1499            while (rdr.Read())
1500            {
1501                rdr.GetValues(values);
1502                // 0:idUserStory, 1:DescriptionUserStory, 2:DateLimite, ↩
                    3:ComplexityEstimation, 4:CompletedComplexity, 5:Blocked, ↩
                    6:ProjectId, 7:StateId, 8:TypeId, 9:PriorityId
1503                UserStory u = new UserStory((int)values[0], (string)values[1], ↩
                        values[2] as DateTime?, (int)values[3], (int)values[4], ↩
                        (bool)values[5], (int)values[6], (int)values[7], ↩
                        (int)values[8], (int)values[9]);
1504                userStories.Add(u);
1505            }
1506
1507            //Close database and reader
1508            rdr.Close();
1509            GetConnection().Close();
1510
1511            return userStories;
1512        }
1513        public static List<User> GetUsers()
1514        {
1515            //Declare variables
1516            OleDbCommand cmd;
1517            OleDbDataReader rdr;
1518            object[] values;
1519            List<User> users = new List<User>();
1520
1521            //Open Database
1522            DB.GetConnection().Open();
```

```
1523            cmd = DB.GetConnection().CreateCommand();
1524
1525            //Execute SQL Command
1526            cmd.CommandText = "SELECT␣*␣FROM␣TUsers␣WHERE␣deletedFlag␣=␣FALSE;";
1527            cmd.Connection = DB.GetConnection();
1528
1529            //Read and put entries in a list of objects
1530            rdr = cmd.ExecuteReader();
1531            values = new object[2];
1532            while (rdr.Read())
1533            {
1534                rdr.GetValues(values);
1535                // 0:idUser,1:NameUser
1536                User u = new User((int)values[0], (string)values[1]);
1537                users.Add(u);
1538            }
1539
1540            //Close database and reader
1541            rdr.Close();
1542            GetConnection().Close();
1543
1544            return users;
1545        }
1546        public static List<Classes.Type> GetTypes()
1547        {
1548            //Declare variables
1549            OleDbCommand cmd;
1550            OleDbDataReader rdr;
1551            object[] values;
1552            List<Classes.Type> types = new List<Classes.Type>();
1553
1554            //Open Database
1555            DB.GetConnection().Open();
1556            cmd = DB.GetConnection().CreateCommand();
1557
1558            //Execute SQL Command
1559            cmd.CommandText = "SELECT␣*␣FROM␣TTypes␣WHERE␣deletedFlag␣=␣FALSE;";
1560            cmd.Connection = DB.GetConnection();
1561
1562            //Read and put entries in a list of objects
1563            rdr = cmd.ExecuteReader();
1564            values = new object[2];
1565            while (rdr.Read())
1566            {
1567                rdr.GetValues(values);
1568                // 0:idType,1:NameType
1569                Classes.Type t = new Classes.Type((int)values[0], ↩
                        (string)values[1]);
1570                types.Add(t);
1571            }
1572
1573            //Close database and reader
1574            rdr.Close();
1575            GetConnection().Close();
1576
1577            return types;
1578        }
1579        public static List<Priority> GetPriorities()
1580        {
1581            //Declare variables
1582            OleDbCommand cmd;
1583            OleDbDataReader rdr;
1584            object[] values;
1585            List<Priority> priorities = new List<Priority>();
1586
1587            //Open Database
1588            DB.GetConnection().Open();
1589            cmd = DB.GetConnection().CreateCommand();
1590
1591            //Execute SQL Command
1592            cmd.CommandText = "SELECT␣*␣FROM␣TPriorities␣WHERE␣deletedFlag␣=␣↩
                    FALSE;";
1593            cmd.Connection = DB.GetConnection();
1594
1595            //Read and put entries in a list of objects
1596            rdr = cmd.ExecuteReader();
1597            values = new object[2];
1598            while (rdr.Read())
1599            {
```

```
1600                rdr.GetValues(values);
1601                // 0:idPriority,1:NamePriority
1602                Priority u = new Priority((int)values[0], (string)values[1]);
1603                priorities.Add(u);
1604            }
1605
1606            //Close database and reader
1607            rdr.Close();
1608            GetConnection().Close();
1609
1610            return priorities;
1611        }
1612        public static List<Classes.File> GetFiles()
1613        {
1614            //Declare variables
1615            OleDbCommand cmd;
1616            OleDbDataReader rdr;
1617            object[] values;
1618            List<Classes.File> files = new List<Classes.File>();
1619
1620            //Open Database
1621            DB.GetConnection().Open();
1622            cmd = DB.GetConnection().CreateCommand();
1623
1624            //Execute SQL Command
1625            cmd.CommandText = "SELECT␣*␣FROM␣TFiles␣WHERE␣deletedFlag␣=␣FALSE;";
1626            cmd.Connection = DB.GetConnection();
1627
1628            //Read and put entries in a list of objects
1629            rdr = cmd.ExecuteReader();
1630            values = new object[4];
1631            while (rdr.Read())
1632            {
1633                rdr.GetValues(values);
1634                // 0:idFile,1:nameFile,2:DescFile,3:idUserStory
1635                Classes.File f = new Classes.File((int)values[0], ↩
                        (string)values[1], (string)values[2], (int)values[3]);
1636                files.Add(f);
1637            }
1638
1639            //Close database and reader
1640            rdr.Close();
1641            GetConnection().Close();
1642
1643            return files;
1644        }
1645        public static List<Activity> GetActivities()
1646        {
1647            //Declare variables
1648            OleDbCommand cmd;
1649            OleDbDataReader rdr;
1650            object[] values;
1651            List<Activity> activities = new List<Activity>();
1652
1653            //Open Database
1654            DB.GetConnection().Open();
1655            cmd = DB.GetConnection().CreateCommand();
1656
1657            //Execute SQL Command
1658            cmd.CommandText = "SELECT␣*␣FROM␣TActivities␣WHERE␣deletedFlag␣=␣↩
                    FALSE;";
1659            cmd.Connection = DB.GetConnection();
1660
1661            //Read and put entries in a list of objects
1662            rdr = cmd.ExecuteReader();
1663            values = new object[4];
1664            while (rdr.Read())
1665            {
1666                rdr.GetValues(values);
1667                // 0:idFileType,1:description,2:DateTimeActivity,3:IdUserStory
1668                Activity a = new Activity((int)values[0], (string)values[1], ↩
                        (DateTime)values[2], (int)values[3]);
1669                activities.Add(a);
1670            }
1671
1672            //Close database and reader
1673            rdr.Close();
1674            GetConnection().Close();
1675
```

```
1676            return activities;
1677        }
1678        public static List<Checklist> GetChecklists()
1679        {
1680            //Declare variables
1681            OleDbCommand cmd;
1682            OleDbDataReader rdr;
1683            object[] values;
1684            List<Checklist> checklists = new List<Checklist>();
1685
1686            //Open Database
1687            DB.GetConnection().Open();
1688            cmd = DB.GetConnection().CreateCommand();
1689
1690            //Execute SQL Command
1691            cmd.CommandText = "SELECT␣*␣FROM␣TChecklists␣WHERE␣deletedFlag␣=␣↩
                FALSE;";
1692            cmd.Connection = DB.GetConnection();
1693
1694            //Read and put entries in a list of objects
1695            rdr = cmd.ExecuteReader();
1696            values = new object[3];
1697            while (rdr.Read())
1698            {
1699                rdr.GetValues(values);
1700                // 0:idChecklist,1:description,2:IdUserStory
1701                Checklist c = new Checklist((int)values[0], (string)values[1], ↩
                    (int)values[2]);
1702                checklists.Add(c);
1703            }
1704
1705            //Close database and reader
1706            rdr.Close();
1707            GetConnection().Close();
1708
1709            return checklists;
1710        }
1711        public static List<ChecklistItem> GetChecklistItems()
1712        {
1713            //Declare variables
1714            OleDbCommand cmd;
1715            OleDbDataReader rdr;
1716            object[] values;
1717            List<ChecklistItem> checklistItems = new List<ChecklistItem>();
1718
1719            //Open Database
1720            DB.GetConnection().Open();
1721            cmd = DB.GetConnection().CreateCommand();
1722
1723            //Execute SQL Command
1724            cmd.CommandText = "SELECT␣*␣FROM␣TChecklistItems␣WHERE␣deletedFlag␣↩
                =␣FALSE;";
1725            cmd.Connection = DB.GetConnection();
1726
1727            //Read and put entries in a list of objects
1728            rdr = cmd.ExecuteReader();
1729            values = new object[4];
1730            while (rdr.Read())
1731            {
1732                rdr.GetValues(values);
1733                // 0:idChecklistItem,1:nameItem,2:done,3:idChecklist
1734                ChecklistItem c = new ChecklistItem((int)values[0], ↩
                    (string)values[1], (bool)values[2], (int)values[3]);
1735                checklistItems.Add(c);
1736            }
1737
1738            //Close database and reader
1739            rdr.Close();
1740            GetConnection().Close();
1741
1742            return checklistItems;
1743        }
1744        public static List<Comment> GetComments()
1745        {
1746            //Declare variables
1747            OleDbCommand cmd;
1748            OleDbDataReader rdr;
1749            object[] values;
1750            List<Comment> comments = new List<Comment>();
```

```
1751
1752            //Open Database
1753            DB.GetConnection().Open();
1754            cmd = DB.GetConnection().CreateCommand();
1755
1756            //Execute SQL Command
1757            cmd.CommandText = "SELECT␣*␣FROM␣TComments␣WHERE␣deletedFlag␣=␣↩
                    FALSE;";
1758            cmd.Connection = DB.GetConnection();
1759
1760            //Read and put entries in a list of objects
1761            rdr = cmd.ExecuteReader();
1762            values = new object[5];
1763            while (rdr.Read())
1764            {
1765                rdr.GetValues(values);
1766                // 0:idComment,1:desc,2:dateTime,3:idUserStory,4:idUser
1767                Comment c = new Comment((int)values[0], (string)values[1], ↩
                        (DateTime)values[2], (int)values[3], (int)values[4]);
1768                comments.Add(c);
1769            }
1770
1771            //Close database and reader
1772            rdr.Close();
1773            GetConnection().Close();
1774
1775            return comments;
1776        }
1777        public static List<State> GetStates()
1778        {
1779            //Declare variables
1780            OleDbCommand cmd;
1781            OleDbDataReader rdr;
1782            object[] values;
1783            List<State> states = new List<State>();
1784
1785            //Open Database
1786            DB.GetConnection().Open();
1787            cmd = DB.GetConnection().CreateCommand();
1788
1789            //Execute SQL Command
1790            cmd.CommandText = "SELECT␣*␣FROM␣TStates␣WHERE␣deletedFlag␣=␣FALSE;";
1791            cmd.Connection = DB.GetConnection();
1792
1793            //Read and put entries in a list of objects
1794            rdr = cmd.ExecuteReader();
1795            values = new object[2];
1796            while (rdr.Read())
1797            {
1798                rdr.GetValues(values);
1799                // 0:IdState,1:NameState
1800                states.Add(new State((int)values[0], (string)values[1]));
1801            }
1802
1803            //Close database and reader
1804            rdr.Close();
1805            GetConnection().Close();
1806
1807            return states;
1808
1809        }
1810        public static List<MindMap> GetMindMaps()
1811        {
1812            //Declare variables
1813            OleDbCommand cmd;
1814            OleDbDataReader rdr;
1815            object[] values;
1816            List<MindMap> mindmaps = new List<MindMap>();
1817
1818            //Open Database
1819            DB.GetConnection().Open();
1820            cmd = DB.GetConnection().CreateCommand();
1821
1822            //Execute SQL Command
1823            cmd.CommandText = "SELECT␣*␣FROM␣TMindMaps␣WHERE␣deletedFlag␣=␣↩
                    FALSE;";
1824            cmd.Connection = DB.GetConnection();
1825
1826            //Read and put entries in a list of objects
```

```
1827                rdr = cmd.ExecuteReader();
1828                values = new object[4];
1829                while (rdr.Read())
1830                {
1831                    rdr.GetValues(values);
1832                    // 0:idMindMap,1:NameMindMap,2:idProject,3:deletedFlag
1833                    MindMap m = new MindMap((int)values[0], (string)values[1], ↩
                        (int)values[2]);
1834                    mindmaps.Add(m);
1835                }
1836
1837                //Close database and reader
1838                rdr.Close();
1839                GetConnection().Close();
1840
1841                return mindmaps;
1842            }
1843            public static List<Node> GetNodes()
1844            {
1845                //Declare variables
1846                OleDbCommand cmd;
1847                OleDbDataReader rdr;
1848                object[] values;
1849                List<Node> nodes = new List<Node>();
1850
1851                //Open Database
1852                DB.GetConnection().Open();
1853                cmd = DB.GetConnection().CreateCommand();
1854
1855                //Execute SQL Command
1856                cmd.CommandText = "SELECT␣*␣FROM␣TNodes␣WHERE␣deletedFlag␣=␣FALSE;";
1857                cmd.Connection = DB.GetConnection();
1858
1859                //Read and put entries in a list of objects
1860                rdr = cmd.ExecuteReader();
1861                values = new object[5];
1862                while (rdr.Read())
1863                {
1864                    rdr.GetValues(values);
1865
1866                    if (values[2] == DBNull.Value)
1867                    {
1868                        values[2] = null;
1869                    }
1870                    // 0:idNode,1:NameNode,2:PreviousIdNode,3:idMindmap,4:deletedFlag
1871                    Node n = new Node((int)values[0], (string)values[1], ↩
                        (int?)values[2], (int)values[3]);
1872                    nodes.Add(n);
1873                }
1874
1875                //Close database and reader
1876                rdr.Close();
1877                GetConnection().Close();
1878
1879                return nodes;
1880            }
1881            #endregion
1882
1883        }
1884    }
```

Listing 84 – ../../Scrum'o'wall/Scrum'o'wall/Controls/DB.cs

5 Tests

5.1 ControllerTests.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    ControllerTests.cs
5   * Desc.     :    This file test the Controller class
6   */
7  using Microsoft.VisualStudio.TestTools.UnitTesting;
8  using Scrum_o_wall.Classes;
9  using System;
10 using System.Collections.Generic;
11 using System.Linq;
12
13 namespace Scrum_o_wall.Tests
14 {
15     [TestClass()]
16     public class ControllerTests
17     {
18         public static Controller ctrl;
19         [TestInitialize]
20         public void TestInitialize()
21         {
22             DB.DbFileName = @"C:\Users\redwo\OneDrive\Scrum-o-Wall\TestDB.accdb";
23
24             ctrl = new Controller();
25         }
26
27
28         [TestMethod]
29         public void ControllerTest()
30         {
31             Assert.IsNotNull(ctrl);
32         }
33
34         [TestMethod()]
35         public void CRDUserStoriesSprintTest()
36         {
37             ctrl.CreateProject("a␣name␣for␣a␣project", "a␣desc␣for␣a␣project", ↩
                   DateTime.Now);
38             Project project = ctrl.Projects.Last();
39
40             ctrl.CreateSprint(DateTime.Now, DateTime.Now, project);
41             Sprint sprint = project.Sprints[0];
42
43             //if not exists, the database is incorrect
44             Priority prio = ctrl.Priorities[0];
45             Classes.Type type = ctrl.Types[0];
46             ctrl.CreateUserStory("a␣description", null, 2, prio, type, project);
47             UserStory userStory = project.AllUserStories[0];
48
49             Assert.IsTrue(ctrl.AddUserStoryToSprint(userStory, sprint));
50             Assert.IsTrue(sprint.OrderedUserStories.ContainsValue(userStory));
51             Assert.IsTrue(ctrl.RemoveUserStoryFromSprint(userStory, sprint));
52
53             ctrl.Delete(userStory);
54             ctrl.Delete(sprint);
55             ctrl.Delete(project);
56         }
57         [TestMethod()]
58         public void CRDProjectStatesTest()
59         {
60             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                   DateTime.Now);
61             Project project = ctrl.Projects.Last();
62
63             ctrl.CreateState("a␣new␣state");
64             State state = ctrl.States.Last();
65
66
67             Assert.IsTrue(ctrl.AddStateToProject(state, project));
68             Assert.IsTrue(project.States.ContainsValue(state));
69             Assert.IsTrue(ctrl.RemoveStateFromProject(state, project));
70
71             ctrl.Delete(project);
72             ctrl.Delete(state);
```

```
73          }
74
75          [TestMethod()]
76          public void CRDUserIUsersAssignedTest()
77          {
78              ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
79              Project project = ctrl.Projects.Last();
80
81              ctrl.CreateUser("a␣user␣name",project);
82              User user = ctrl.Users.Last();
83
84              Assert.IsTrue(ctrl.AddUserToIUsersAssigned(user, project));
85              Assert.IsTrue(project.GetUsers().Contains(user));
86              Assert.IsTrue(ctrl.RemoveUserFromIUsersAssigned(user, project));
87
88
89              ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
90              UserStory userStory = project.AllUserStories[0];
91
92              Assert.IsTrue(ctrl.AddUserToIUsersAssigned(user, userStory));
93              Assert.IsTrue(userStory.GetUsers().Contains(user));
94              Assert.IsTrue(ctrl.RemoveUserFromIUsersAssigned(user, userStory));
95
96              Checklist checklist = ctrl.CreateCheckList("a␣bioutifoul␣name", ↩
                    userStory);
97              ctrl.CreateCheckListItem("a␣better␣name", checklist);
98              ChecklistItem checklistItem = checklist.ChecklistItems[0];
99
100             Assert.IsTrue(ctrl.AddUserToIUsersAssigned(user, checklistItem));
101             Assert.IsTrue(checklistItem.GetUsers().Contains(user));
102             Assert.IsTrue(ctrl.RemoveUserFromIUsersAssigned(user, checklistItem));
103
104             ctrl.Delete(user);
105             ctrl.Delete(checklistItem);
106             ctrl.Delete(checklist);
107             ctrl.Delete(userStory);
108             ctrl.Delete(project);
109         }
110         [TestMethod]
111         public void CRDActivity()
112         {
113             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
114             Project project = ctrl.Projects.Last();
115             ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
116             UserStory userStory = project.AllUserStories[0];
117             string aDesc = "a␣desc␣for␣my␣activity";
118
119             Assert.IsTrue(ctrl.CreateActivity(aDesc, userStory));
120             Activity activity = userStory.Activities.Last();
121             Assert.AreEqual(aDesc, activity.Description);
122             Assert.IsTrue(ctrl.Delete(activity));
123
124             ctrl.Delete(userStory);
125             ctrl.Delete(project);
126         }
127
128         [TestMethod]
129         public void CRUDChecklist()
130         {
131             //Test Create
132             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
133             Project project = ctrl.Projects.Last();
134             ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
135             UserStory userStory = project.AllUserStories[0];
136             string aName = "a␣first␣name";
137
138             Checklist checklist = ctrl.CreateCheckList(aName, userStory);
139
140             Assert.AreEqual(aName, checklist.Name);
141             Assert.AreEqual(userStory.Id, checklist.UserStoryId);
142
143             //Test Update
144             string afterName = "checklist␣second␣name";
```

```
145
146             Assert.IsTrue(ctrl.UpdateCheckList(afterName, new ↩
                    List<ChecklistItem>(), checklist));
147
148             Assert.AreEqual(afterName, checklist.Name);
149
150             //Test Remove
151             Assert.IsTrue(ctrl.Delete(checklist));
152
153             ctrl.Delete(userStory);
154             ctrl.Delete(project);
155         }
156         [TestMethod]
157         public void CRUDChecklistItem()
158         {
159             //Test Create
160             string aName = "checlistItem␣first␣name";
161             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
162             Project project = ctrl.Projects.Last();
163             ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
164             UserStory userStory = project.AllUserStories[0];
165             ctrl.CreateCheckList("aCheck", userStory);
166             Checklist checklist = userStory.Checklists[0];
167
168             Assert.IsTrue(ctrl.CreateCheckListItem(aName, checklist));
169             ChecklistItem checklistItem = checklist.ChecklistItems[0];
170
171             Assert.AreEqual(aName, checklistItem.NameItem);
172             Assert.IsFalse(checklistItem.Done);
173             Assert.AreEqual(checklist, checklistItem.Checklist);
174
175             //Test Update
176             string afterName = "checklistItem␣second␣name";
177
178             Assert.IsTrue(ctrl.UpdateCheckListItem(afterName, true, ↩
                    checklistItem));
179
180             Assert.AreEqual(afterName, checklistItem.NameItem);
181             Assert.IsTrue(checklistItem.Done);
182
183             //Test Remove
184             Assert.IsTrue(ctrl.Delete(checklistItem));
185
186             ctrl.Delete(checklist);
187             ctrl.Delete(userStory);
188             ctrl.Delete(project);
189         }
190         [TestMethod]
191         public void CRDComment()
192         {
193             //Test Create
194             string aName = "comment␣first␣name";
195             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
196             Project project = ctrl.Projects.Last();
197             ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
198             UserStory userStory = project.AllUserStories[0];
199
200             ctrl.CreateUser("a␣user␣name", userStory);
201             User user = ctrl.Users.Last();
202
203             Assert.IsTrue(ctrl.CreateComment(aName, user, userStory));
204             Comment comment = userStory.Comments[0];
205
206             Assert.AreEqual(aName, comment.Description);
207             Assert.AreEqual(userStory.Id, comment.UserStoryId);
208             Assert.AreEqual(user.Id, comment.UserId);
209
210             Assert.IsTrue(ctrl.Delete(comment));
211
212             ctrl.Delete(user);
213             ctrl.Delete(userStory);
214             ctrl.Delete(project);
215         }
216         [TestMethod]
217         public void CRUDFile()
```

```
218         {
219             //Declare needed variables
220             string afterDesc = "file second name";
221             string aName = "file first name";
222             string aDesc = "this is a description";
223             ctrl.CreateProject("aProjectName", "A Description for my project", ↩
                    DateTime.Now);
224             Project project = ctrl.Projects.Last();
225             ctrl.CreateUserStory("aDescription", null, 2, ctrl.Priorities[0], ↩
                    ctrl.Types[0], project);
226             UserStory userStory = project.AllUserStories[0];
227 
228             //Test Create
229             Assert.IsTrue(ctrl.CreateFile(aName, aDesc, userStory));
230             File file = userStory.Files[0];
231 
232             Assert.AreEqual(aName, file.Name);
233             Assert.AreEqual(aDesc, file.Description);
234             Assert.AreEqual(userStory.Id, file.UserStoryId);
235 
236             //Test Update
237             Assert.IsTrue(ctrl.UpdateFile(afterDesc, file));
238             Assert.AreEqual(afterDesc, file.Description);
239 
240             //Test Remove
241             Assert.IsTrue(ctrl.Delete(file));
242 
243             ctrl.Delete(project);
244             ctrl.Delete(userStory);
245         }
246         [TestMethod]
247         public void CRUDProject()
248         {
249             string firstName = "the project first Name";
250             string firstDesc = "the project first description";
251             DateTime firstDate = DateTime.Now;
252             string secName = "the project sec Name";
253             string secDesc = "the project sec description";
254             DateTime secDate = DateTime.Now + new TimeSpan(7, 0, 0, 0);
255 
256             //Test Create
257             Assert.IsTrue(ctrl.CreateProject(firstName, firstDesc, firstDate));
258             Project project = ctrl.Projects.Last();
259 
260             Assert.AreEqual(firstName, project.Name);
261             Assert.AreEqual(firstDesc, project.Description);
262             Assert.AreEqual(firstDate.ToString(), project.Begin.ToString());
263 
264             //Test Update
265             Assert.IsTrue(ctrl.UpdateProject(secName, secDesc, secDate, project));
266 
267             Assert.AreEqual(secName, project.Name);
268             Assert.AreEqual(secDesc, project.Description);
269             Assert.AreEqual(secDate.ToString(), project.Begin.ToString());
270 
271             //Test Remove
272             Assert.IsTrue(ctrl.Delete(project));
273         }
274         [TestMethod]
275         public void CRUDSprint()
276         {
277             DateTime firstBegin = DateTime.Now;
278             DateTime firstEnd = firstBegin + new TimeSpan(7, 0, 0, 0);
279             ctrl.CreateProject("aProjectName", "A Description for my project", ↩
                    DateTime.Now);
280             Project project = ctrl.Projects.Last();
281             DateTime secBegin = firstEnd;
282             DateTime secEnd = secBegin + new TimeSpan(7, 0, 0, 0);
283 
284             //Test Create
285             Assert.IsTrue(ctrl.CreateSprint(firstBegin, firstEnd, project));
286             Sprint sprint = project.Sprints[0];
287 
288             Assert.AreEqual(firstBegin.ToString(), sprint.Begin.ToString());
289             Assert.AreEqual(firstEnd.ToString(), sprint.End.ToString());
290 
291             //Test Update
292 
293             Assert.IsTrue(ctrl.UpdateSprint(secBegin, secEnd, sprint));
```

```
294
295             Assert.AreEqual(secBegin.ToString(), sprint.Begin.ToString());
296             Assert.AreEqual(secEnd.ToString(), sprint.End.ToString());
297
298             //Test Remove
299             Assert.IsTrue(ctrl.Delete(sprint));
300         }
301         [TestMethod]
302         public void CRUDState()
303         {
304             //Test Create
305             string firstName = "first␣state␣name";
306
307             Assert.IsTrue(ctrl.CreateState(firstName));
308             State state = ctrl.States.Last();
309
310             Assert.AreEqual(firstName, state.Name);
311
312
313             //Test Remove
314             Assert.IsTrue(ctrl.Delete(state));
315         }
316         [TestMethod]
317         public void CRUDUser()
318         {
319             //Test Create
320             string firstName = "first␣user␣name";
321
322             Assert.IsTrue(ctrl.CreateUser(firstName,null));
323             User user = ctrl.Users.Last();
324
325             Assert.AreEqual(firstName, user.Name);
326
327             //Test Remove
328             Assert.IsTrue(ctrl.Delete(user));
329         }
330         [TestMethod]
331         public void CRUDUserStoryTest()
332         {
333             ctrl.CreateProject("aProjectName", "A␣Description␣for␣my␣project", ↩
                    DateTime.Now);
334             Project project = ctrl.Projects.Last();
335
336             string firstDesc = "aDesc";
337             DateTime? firstDate = DateTime.Now;
338             int firstComplexity = 2;
339             Priority firstPrio = ctrl.Priorities[0];
340             Classes.Type firstType = ctrl.Types[0];
341
342
343             Assert.IsTrue(ctrl.CreateUserStory(firstDesc, firstDate, ↩
                    firstComplexity, firstPrio, firstType, project));
344             UserStory userStory = project.AllUserStories[0];
345
346             Assert.IsNotNull(userStory, "Exception␣in␣userStory␣creation");
347             Assert.AreEqual(firstDesc, userStory.Description);
348             Assert.AreEqual(firstDate, userStory.DateLimit);
349             Assert.AreEqual(firstComplexity, userStory.ComplexityEstimation);
350             Assert.AreEqual(firstPrio, userStory.Priority);
351             Assert.AreEqual(firstType, userStory.Type);
352             Assert.AreEqual(project, userStory.Project);
353             Assert.AreEqual(false, userStory.Blocked);
354             Assert.AreEqual(0, userStory.CompletedComplexity);
355
356
357             string secDesc = "aNewDesc";
358             DateTime? secDate = null;
359             int secComplexity = 3;
360             int secCompleted = 1;
361             bool secBlock = true;
362             Priority secPrio = ctrl.Priorities[1];
363             Classes.Type secType = ctrl.Types[1];
364             while (ctrl.States.Count < 2)
365             {
366                 ctrl.CreateState("An␣additional␣state␣name");
367             }
368             State secState = ctrl.States.Last();
369
```

```
370             Assert.IsTrue(ctrl.UpdateUserStory(secDesc, secDate, secComplexity, ↩
                    secCompleted, secBlock, secPrio, secType, secState, userStory));
371
372             Assert.AreEqual(secDesc, userStory.Description);
373             Assert.AreEqual(secDate, userStory.DateLimit);
374             Assert.AreEqual(secComplexity, userStory.ComplexityEstimation);
375             Assert.AreEqual(secPrio, userStory.Priority);
376             Assert.AreEqual(secType, userStory.Type);
377             Assert.AreEqual(secState, userStory.State);
378             Assert.AreEqual(secBlock, userStory.Blocked);
379             Assert.AreEqual(secCompleted, userStory.CompletedComplexity);
380
381             Assert.IsTrue(ctrl.Delete(userStory));
382
383             ctrl.Delete(project);
384         }
385
386         [TestMethod]
387         public void CRUDMindMap()
388         {
389
390             string firstName = "firstMindMap␣name";
391             string secondName = "secondtMindMap␣name";
392             ctrl.CreateProject("a␣project␣name", "a␣description", DateTime.Now);
393             Project project = ctrl.Projects.Last();
394
395             //Test Create
396
397             Assert.IsTrue(ctrl.CreateMindMap(firstName, project));
398             MindMap mindMap = project.MindMaps.Last();
399
400             Assert.AreEqual(firstName, mindMap.Name);
401
402             //Test Update
403             Assert.IsTrue(ctrl.UpdateMindMap(secondName, mindMap));
404             Assert.AreEqual(secondName, mindMap.Name);
405
406             //Test Remove
407             Assert.IsTrue(ctrl.Delete(mindMap));
408
409             ctrl.Delete(project);
410         }
411         [TestMethod]
412         public void CRUDNode()
413         {
414             ctrl.CreateProject("a␣project␣name", "a␣description", DateTime.Now);
415             Project project = ctrl.Projects.Last();
416             ctrl.CreateMindMap("a␣mind␣map␣name", project);
417             MindMap mindMap = project.MindMaps.Last();
418
419             //Test Create Without Previous
420             string firstName = "first␣node␣name";
421
422             Assert.IsTrue(ctrl.CreateNode(firstName, null, mindMap));
423             Node node = mindMap.Root;
424
425             Assert.AreEqual(firstName, node.Name);
426             Assert.AreEqual(mindMap, node.MindMap);
427             //Test Create With Previous
428             string firstName2 = "first␣node␣name2";
429
430             Assert.IsTrue(ctrl.CreateNode(firstName2, node, mindMap));
431             Node node2 = node.Childrens.Last();
432
433             Assert.AreEqual(firstName2, node2.Name);
434             Assert.AreEqual(node, node2.Previous);
435             Assert.AreEqual(node2, node.Childrens.Last());
436
437             //Test Update
438             string secondName = "second␣node␣name";
439
440             Assert.IsFalse(ctrl.UpdateNode(secondName, node2, node));
441
442             Assert.IsTrue(ctrl.UpdateNode(secondName, null, node));
443             Assert.AreEqual(secondName, node.Name);
444
445             //Test Remove
446             Assert.IsFalse(ctrl.Delete(node));
447             Assert.IsTrue(ctrl.Delete(node2));
```

```
448
449             ctrl.Delete(mindMap);
450             ctrl.Delete(project);
451         }
452
453     }
454 }
```

Listing 85 – ../../Scrum'o'wall/Scrum'o'wallTests/ControllerTests.cs

5.2 DBTests.cs

```
1  /*
2   * Author    :    Ga l Serge Mariot
3   * Project   :    Scrum'o'wall
4   * File      :    DBTests.cs
5   * Desc.     :    This file test the DB class
6   */
7  using Microsoft.VisualStudio.TestTools.UnitTesting;
8  using Scrum_o_wall.Classes;
9  using System;
10 using System.Collections.Generic;
11 using System.Linq;
12
13 namespace Scrum_o_wall.Tests
14 {
15     [TestClass()]
16     public class DBTests
17     {
18         [TestInitialize]
19         public void TestInitialize()
20         {
21             DB.DbFileName = @"C:\Users\redwo\OneDrive\Scrum-o-Wall\TestDB.accdb";
22         }
23
24         [TestMethod()]
25         public void CRDUserStoriesSprintTest()
26         {
27             Project project = DB.CreateProject("aname", "adesc", DateTime.Now);
28
29             //Create UserStory
30             UserStory userStory;
31             List<Priority> priorities = DB.GetPriorities();
32             List<State> states = DB.GetStates();
33             List<Classes.Type> types = DB.GetTypes();
34             if (states.Count == 0)
35             {
36                 DB.CreateState("AStateName");
37                 states = DB.GetStates();
38             }
39             userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                   priorities[0], types[0], states[0], project);
40
41             //Create Sprint
42             Sprint sprint;
43             sprint = DB.CreateSprint(DateTime.Now, DateTime.Now, project);
44             int order = 0;
45
46             //Link,test,unlink
47             Assert.IsTrue(DB.AddUserStoryToSprint(userStory, sprint, order));
48             Assert.IsNotNull(DB.GetUserStoriesSprint());
49             Assert.IsTrue(DB.RemoveUserStoryFromSprint(userStory, sprint, order));
50
51             DB.Delete(project);
52             DB.Delete(userStory);
53             DB.Delete(sprint);
54         }
55         [TestMethod()]
56         public void CRDProjectStatesTest()
57         {
58             Project project = DB.CreateProject("aProjectName", "A␣Description␣↩
                   for␣my␣project", DateTime.Now);
59             State state = DB.CreateState("a␣new␣state");
60             int order = 0;
61
62             Assert.IsTrue(DB.AddStateToProject(state, project, order));
63             Assert.IsNotNull(DB.GetProjectStates());
```

```
64              Assert.IsTrue(DB.RemoveStateFromProject(project, order));
65
66              DB.Delete(project);
67              DB.Delete(state);
68          }
69
70          [TestMethod()]
71          public void CRDUserProjectTest()
72          {
73              Project project = DB.CreateProject("aname", "adesc", DateTime.Now);
74              User user = DB.CreateUser("a user name");
75
76              Assert.IsTrue(DB.AddUserToProject(user, project));
77              Assert.IsNotNull(DB.GetUserProject());
78              Assert.IsTrue(DB.RemoveUserFromProject(user, project));
79
80              DB.Delete(user);
81              DB.Delete(project);
82          }
83
84          [TestMethod()]
85          public void CRDUserChecklistItemTest()
86          {
87              //CreateChecklistItem
88              List<Priority> priorities = DB.GetPriorities();
89              List<State> states = DB.GetStates();
90              List<Classes.Type> types = DB.GetTypes();
91              List<Project> projects = DB.GetProjects();
92              if (states.Count == 0)
93              {
94                  DB.CreateState("AStateName");
95                  states = DB.GetStates();
96              }
97              if (projects.Count == 0)
98              {
99                  DB.CreateProject("aProjectName", "A Description for my ↩
                        project", DateTime.Now);
100                 projects = DB.GetProjects();
101             }
102             UserStory userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                    priorities[0], types[0], states[0], projects[0]);
103             Checklist checklist = DB.CreateCheckList("a bioutifoul name", ↩
                    userStory);
104             ChecklistItem checklistItem = DB.CreateCheckListItem("a better ↩
                    name", checklist);
105
106             //CreateUser
107             User user = DB.CreateUser("a user name");
108
109             Assert.IsTrue(DB.AddUserToChecklistItem(user, checklistItem));
110             Assert.IsNotNull(DB.GetUserChecklistItem());
111             Assert.IsTrue(DB.RemoveUserFromChecklistItem(user, checklistItem));
112
113             DB.Delete(user);
114             DB.Delete(userStory);
115             DB.Delete(checklistItem);
116             DB.Delete(checklist);
117         }
118         [TestMethod()]
119         public void CRDUserUserStoryTest()
120         {
121             //Create User Story
122             List<Priority> priorities = DB.GetPriorities();
123             List<State> states = DB.GetStates();
124             List<Classes.Type> types = DB.GetTypes();
125             List<Project> projects = DB.GetProjects();
126             if (states.Count == 0)
127             {
128                 DB.CreateState("AStateName");
129                 states = DB.GetStates();
130             }
131             if (projects.Count == 0)
132             {
133                 DB.CreateProject("aProjectName", "A Description for my ↩
                        project", DateTime.Now);
134                 projects = DB.GetProjects();
135             }
136             UserStory userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                    priorities[0], types[0], states[0], projects[0]);
```

```
137
138             //Create User
139             User user = DB.CreateUser("a user name");
140
141             Assert.IsTrue(DB.AddUserToUserStory(user, userStory));
142             Assert.IsNotNull(DB.GetUserUserStory());
143             Assert.IsTrue(DB.RemoveUserFromUserStory(user, userStory));
144
145             DB.Delete(userStory);
146             DB.Delete(user);
147         }
148         [TestMethod()]
149         public void GetTypesTest()
150         {
151             Assert.IsNotNull(DB.GetTypes());
152         }
153         [TestMethod()]
154         public void GetPrioritiesTest()
155         {
156             Assert.IsNotNull(DB.GetPriorities());
157         }
158         [TestMethod]
159         public void CRDActivity()
160         {
161             string aDesc = "activity test";
162             DateTime eventTime = DateTime.Now;
163             List<UserStory> userStories = DB.GetUserStories();
164             UserStory userStory;
165             if (userStories.Count == 0)
166             {
167                 List<Priority> priorities = DB.GetPriorities();
168                 List<State> states = DB.GetStates();
169                 List<Classes.Type> types = DB.GetTypes();
170                 List<Project> projects = DB.GetProjects();
171                 if (states.Count == 0)
172                 {
173                     DB.CreateState("AStateName");
174                     states = DB.GetStates();
175                 }
176                 if (projects.Count == 0)
177                 {
178                     DB.CreateProject("aProjectName", "A Description for my ↩
                            project", DateTime.Now);
179                     projects = DB.GetProjects();
180                 }
181
182
183                 userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                        priorities[0], types[0], states[0], projects[0]);
184             }
185             else
186             {
187                 userStory = userStories.Last();
188             }
189
190
191             Activity a = DB.CreateActivity(aDesc, eventTime, userStory);
192
193             Assert.AreEqual(aDesc, a.Description);
194             Assert.AreEqual(eventTime, a.DateTime);
195             Assert.AreEqual(userStory.Id, a.UserStoryId);
196
197             Assert.IsNotNull(DB.GetActivities());
198
199             Assert.IsTrue(DB.Delete(a));
200         }
201         [TestMethod]
202         public void CRUDChecklist()
203         {
204             //Test Create
205             string aName = "checlist first name";
206             List<UserStory> userStories = DB.GetUserStories();
207             UserStory userStory;
208             if (userStories.Count == 0)
209             {
210                 List<Priority> priorities = DB.GetPriorities();
211                 List<State> states = DB.GetStates();
212                 List<Classes.Type> types = DB.GetTypes();
213                 List<Project> projects = DB.GetProjects();
```

```
214                 if (states.Count == 0)
215                 {
216                     DB.CreateState("AStateName");
217                     states = DB.GetStates();
218                 }
219                 if (projects.Count == 0)
220                 {
221                     DB.CreateProject("aProjectName", "A␣Description␣for␣my␣project", DateTime.Now);
222                     projects = DB.GetProjects();
223                 }
224 
225 
226                 userStory = DB.CreateUserStory("aDescription", null, 2, priorities[0], types[0], states[0], projects[0]);
227             }
228             else
229             {
230                 userStory = userStories.Last();
231             }
232 
233 
234             Checklist checklist = DB.CreateCheckList(aName, userStory);
235 
236             Assert.AreEqual(aName, checklist.Name);
237             Assert.AreEqual(userStory.Id, checklist.UserStoryId);
238 
239             //Test Update
240             string afterName = "checklist␣second␣name";
241 
242             DB.UpdateCheckList(afterName, checklist);
243             checklist = DB.GetChecklists().First(c => c.Id == checklist.Id);
244 
245             Assert.AreEqual(afterName, checklist.Name);
246 
247             //Test Remove
248             Assert.IsTrue(DB.Delete(checklist));
249 
250         }
251         [TestMethod]
252         public void CRUDChecklistItem()
253         {
254             //Test Create
255             string aName = "checlistItem␣first␣name";
256             List<Checklist> checklists = DB.GetChecklists();
257             Checklist checklist;
258             if (checklists.Count == 0)
259             {
260                 List<UserStory> userStories = DB.GetUserStories();
261                 UserStory userStory;
262                 if (userStories.Count == 0)
263                 {
264                     List<Priority> priorities = DB.GetPriorities();
265                     List<State> states = DB.GetStates();
266                     List<Classes.Type> types = DB.GetTypes();
267                     List<Project> projects = DB.GetProjects();
268                     if (states.Count == 0)
269                     {
270                         DB.CreateState("AStateName");
271                         states = DB.GetStates();
272                     }
273                     if (projects.Count == 0)
274                     {
275                         DB.CreateProject("aProjectName", "A␣Description␣for␣my␣project", DateTime.Now);
276                         projects = DB.GetProjects();
277                     }
278 
279 
280                     userStory = DB.CreateUserStory("aDescription", null, 2, priorities[0], types[0], states[0], projects[0]);
281                 }
282                 else
283                 {
284                     userStory = userStories.Last();
285                 }
286 
287                 checklist = DB.CreateCheckList("the␣name␣of␣a␣checklist", userStory);
```

```
288
289             }
290             else
291             {
292                 checklist = checklists[0];
293             }
294             ChecklistItem checklistItem = DB.CreateCheckListItem(aName, ↩
                    checklist);
295
296             Assert.AreEqual(aName, checklistItem.NameItem);
297             Assert.IsFalse(checklistItem.Done);
298             Assert.AreEqual(checklist.Id, checklistItem.ChecklistId);
299
300             //Test Update
301             string afterName = "checklistItem␣second␣name";
302
303             DB.UpdateCheckListItem(afterName, true, checklistItem);
304             checklistItem = DB.GetChecklistItems().First(c => c.Id == ↩
                    checklistItem.Id);
305
306             Assert.AreEqual(afterName, checklistItem.NameItem);
307             Assert.IsTrue(checklistItem.Done);
308
309             //Test Remove
310             Assert.IsTrue(DB.Delete(checklistItem));
311         }
312         [TestMethod]
313         public void CRDComment()
314         {
315             //Test Create
316             string aName = "comment␣first␣name";
317             List<UserStory> userStories = DB.GetUserStories();
318             UserStory userStory;
319             if (userStories.Count == 0)
320             {
321                 List<Priority> priorities = DB.GetPriorities();
322                 List<State> states = DB.GetStates();
323                 List<Classes.Type> types = DB.GetTypes();
324                 List<Project> projects = DB.GetProjects();
325                 if (states.Count == 0)
326                 {
327                     DB.CreateState("AStateName");
328                     states = DB.GetStates();
329                 }
330                 if (projects.Count == 0)
331                 {
332                     DB.CreateProject("aProjectName", "A␣Description␣for␣my␣↩
                            project", DateTime.Now);
333                     projects = DB.GetProjects();
334                 }
335
336
337                 userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                        priorities[0], types[0], states[0], projects[0]);
338             }
339             else
340             {
341                 userStory = userStories.Last();
342             }
343
344             User user = DB.GetUsers().Last();
345
346             Comment comment = DB.CreateComment(aName, userStory, user);
347
348             Assert.AreEqual(aName, comment.Description);
349             Assert.AreEqual(userStory.Id, comment.UserStoryId);
350             Assert.AreEqual(user.Id, comment.UserId);
351
352             Assert.IsNotNull(DB.GetComments());
353
354             //Test Remove
355             Assert.IsTrue(DB.Delete(comment));
356         }
357         [TestMethod]
358         public void CRUDFile()
359         {
360             //Test Create
361             string aName = "file␣first␣name";
362             string aDesc = "this␣is␣a␣description";
```

```
363             List<UserStory> userStories = DB.GetUserStories();
364             UserStory userStory;
365             if (userStories.Count == 0)
366             {
367                 List<Priority> priorities = DB.GetPriorities();
368                 List<State> states = DB.GetStates();
369                 List<Classes.Type> types = DB.GetTypes();
370                 List<Project> projects = DB.GetProjects();
371                 if (states.Count == 0)
372                 {
373                     DB.CreateState("AStateName");
374                     states = DB.GetStates();
375                 }
376                 if (projects.Count == 0)
377                 {
378                     DB.CreateProject("aProjectName", "A␣Description␣for␣my␣↩
                        project", DateTime.Now);
379                     projects = DB.GetProjects();
380                 }
381 
382 
383                 userStory = DB.CreateUserStory("aDescription", null, 2, ↩
                    priorities[0], types[0], states[0], projects[0]);
384             }
385             else
386             {
387                 userStory = userStories.Last();
388             }
389 
390             File file = DB.CreateFile(aName, aDesc, userStory);
391 
392             Assert.AreEqual(aName, file.Name);
393             Assert.AreEqual(aDesc, file.Description);
394             Assert.AreEqual(userStory.Id, file.UserStoryId);
395 
396             //Test Update
397             string afterDesc = "file␣second␣name";
398 
399             DB.UpdateFile(afterDesc, file);
400             file = DB.GetFiles().First(f => f.Id == file.Id);
401 
402             Assert.AreEqual(afterDesc, file.Description);
403 
404             //Test Remove
405             Assert.IsTrue(DB.Delete(file));
406         }
407         [TestMethod]
408         public void CRUDProject()
409         {
410             //Test Create
411             string firstName = "the␣project␣first␣Name";
412             string firstDesc = "the␣project␣first␣description";
413             DateTime firstDate = DateTime.Now;
414 
415             Project project = DB.CreateProject(firstName, firstDesc, firstDate);
416 
417             Assert.AreEqual(firstName, project.Name);
418             Assert.AreEqual(firstDesc, project.Description);
419             Assert.AreEqual(firstDate.ToString(), project.Begin.ToString());
420 
421             //Test Update
422             string secName = "the␣project␣sec␣Name";
423             string secDesc = "the␣project␣sec␣description";
424             DateTime secDate = DateTime.Now + new TimeSpan(7, 0, 0, 0);
425 
426             DB.UpdateProject(secName, secDesc, secDate, project);
427             project = DB.GetProjects().First(p => p.Id == project.Id);
428 
429             Assert.AreEqual(secName, project.Name);
430             Assert.AreEqual(secDesc, project.Description);
431             Assert.AreEqual(secDate.ToString(), project.Begin.ToString());
432 
433             //Test Remove
434             Assert.IsTrue(DB.Delete(project));
435         }
436         [TestMethod]
437         public void CRUDSprint()
438         {
439             //Test Create
```

```
440             DateTime firstBegin = DateTime.Now;
441             DateTime firstEnd = firstBegin + new TimeSpan(7, 0, 0, 0);
442             Project project = DB.GetProjects().Last();
443 
444             Sprint sprint = DB.CreateSprint(firstBegin, firstEnd, project);
445 
446             Assert.AreEqual(firstBegin.ToString(), sprint.Begin.ToString());
447             Assert.AreEqual(firstEnd.ToString(), sprint.End.ToString());
448 
449             //Test Update
450             DateTime secBegin = firstEnd;
451             DateTime secEnd = secBegin + new TimeSpan(7, 0, 0, 0);
452 
453             DB.UpdateSprint(secBegin, secEnd, sprint);
454             sprint = DB.GetSprints().First(s => s.Id == sprint.Id);
455 
456             Assert.AreEqual(secBegin.ToString(), sprint.Begin.ToString());
457             Assert.AreEqual(secEnd.ToString(), sprint.End.ToString());
458 
459             //Test Remove
460             Assert.IsTrue(DB.Delete(sprint));
461         }
462         [TestMethod]
463         public void CRUDState()
464         {
465             //Test Create
466             string firstName = "first␣state␣name";
467 
468             State state = DB.CreateState(firstName);
469 
470             Assert.AreEqual(firstName, state.Name);
471 
472 
473             //Test Remove
474             Assert.IsTrue(DB.Delete(state));
475         }
476         [TestMethod]
477         public void CRUDUser()
478         {
479             //Test Create
480             string firstName = "first␣user␣name";
481 
482             User user = DB.CreateUser(firstName);
483 
484             Assert.AreEqual(firstName, user.Name);
485 
486             //Test Remove
487             Assert.IsTrue(DB.Delete(user));
488         }
489         [TestMethod]
490         public void CRUDUserStoryTest()
491         {
492             List<Priority> priorities = DB.GetPriorities();
493             List<State> states = DB.GetStates();
494             List<Classes.Type> types = DB.GetTypes();
495             List<Project> projects = DB.GetProjects();
496             if (states.Count == 0)
497             {
498                 DB.CreateState("AStateName");
499                 states = DB.GetStates();
500             }
501             if (projects.Count == 0)
502             {
503                 DB.CreateProject("aProjectName", "A␣Description␣for␣my␣↩
                        project", DateTime.Now);
504                 projects = DB.GetProjects();
505             }
506 
507             string firstDesc = "aDesc";
508             DateTime? firstDate = DateTime.Now;
509             int firstComplexity = 2;
510             Priority firstPrio = priorities[0];
511             Classes.Type firstType = types[0];
512             State firstState = states[0];
513             Project firstProj = projects[0];
514 
515 
516             UserStory userStory = DB.CreateUserStory(firstDesc, firstDate, ↩
                    firstComplexity, firstPrio, firstType, firstState, firstProj);
```

```
517
518             Assert.IsNotNull(userStory, "Exception␣in␣userStory␣creation");
519             Assert.AreEqual(firstDesc, userStory.Description);
520             Assert.AreEqual(firstDate, userStory.DateLimit);
521             Assert.AreEqual(firstComplexity, userStory.ComplexityEstimation);
522             Assert.AreEqual(firstPrio.Id, userStory.PriorityId);
523             Assert.AreEqual(firstType.Id, userStory.TypeId);
524             Assert.AreEqual(firstState.Id, userStory.StateId);
525             Assert.AreEqual(firstProj.Id, userStory.ProjectId);
526             Assert.AreEqual(false, userStory.Blocked);
527             Assert.AreEqual(0, userStory.CompletedComplexity);
528
529
530             string secDesc = "aNewDesc";
531             DateTime? secDate = null;
532             int secComplexity = 3;
533             int secCompleted = 1;
534             bool secBlock = true;
535             Priority secPrio = priorities[1];
536             Classes.Type secType = types[1];
537             while (states.Count < 2)
538             {
539                 DB.CreateState("An␣additional␣state␣name");
540                 states = DB.GetStates();
541             }
542             State secState = states[1];
543
544             DB.UpdateUserStory(secDesc, secDate, secComplexity, secCompleted, ↩
                    secBlock, secPrio, secState, secType, userStory);
545
546             userStory = DB.GetUserStories().First(u => u.Id == userStory.Id);
547
548             Assert.AreEqual(secDesc, userStory.Description);
549             Assert.AreEqual(secDate, userStory.DateLimit);
550             Assert.AreEqual(secComplexity, userStory.ComplexityEstimation);
551             Assert.AreEqual(secPrio.Id, userStory.PriorityId);
552             Assert.AreEqual(secType.Id, userStory.TypeId);
553             Assert.AreEqual(secState.Id, userStory.StateId);
554             Assert.AreEqual(secBlock, userStory.Blocked);
555             Assert.AreEqual(secCompleted, userStory.CompletedComplexity);
556
557             Assert.IsTrue(DB.Delete(userStory));
558         }
559
560         [TestMethod]
561         public void CRUDMindMap()
562         {
563             Project project;
564
565             List<Project> projects = DB.GetProjects();
566             if (projects.Count == 0)
567             {
568                 project = DB.CreateProject("a␣Project", "A␣desc␣of␣a␣project", ↩
                        DateTime.Now);
569             }
570             else
571             {
572                 project = projects[0];
573             }
574
575             string firstName = "a␣mindmap␣first␣name";
576             string secondName = "a␣mindmap␣second␣name";
577
578             //Verify creation
579             MindMap mindMap = DB.CreateMindmap(firstName, project);
580             Assert.IsNotNull(mindMap);
581             Assert.AreEqual(firstName, mindMap.Name);
582             Assert.AreEqual(project.Id, mindMap.ProjectId);
583
584             //Verify update
585             Assert.IsTrue(DB.UpdateMindMap(secondName, mindMap));
586
587             //Verify delete
588             Assert.IsTrue(DB.Delete(mindMap));
589
590             DB.Delete(project);
591         }
592         [TestMethod]
593         public void CRUDNode()
```

```
594         {
595             Project project;
596             MindMap mindmap;
597 
598             List<Project> projects = DB.GetProjects();
599             if (projects.Count == 0)
600             {
601                 project = DB.CreateProject("a␣Project", "A␣desc␣of␣a␣project", ↩
                        DateTime.Now);
602             }
603             else
604             {
605                 project = projects[0];
606             }
607             List<MindMap> mindMaps = DB.GetMindMaps().Where(m => m.ProjectId == ↩
                    project.Id).ToList();
608             if (mindMaps.Count == 0)
609             {
610                 mindmap = DB.CreateMindmap("a␣mindmap␣name", project);
611             }
612             else
613             {
614                 mindmap = mindMaps[0];
615             }
616 
617             string firstName = "a␣node␣first␣name";
618             string firstName2 = "a␣second␣node␣first␣name";
619             string secondName = "a␣node␣second␣name";
620             string secondName2 = "a␣second␣node␣second␣name";
621 
622             //Verify creation with and without previous
623             Node node = DB.CreateNode(firstName, null, mindmap);
624 
625             Assert.IsNotNull(node);
626             Assert.IsNull(node.PreviousId);
627             Assert.AreEqual(firstName, node.Name);
628             Assert.AreEqual(mindmap.Id, node.MindmapId);
629 
630             Node node2 = DB.CreateNode(firstName2, node, mindmap);
631 
632             Assert.IsNotNull(node2);
633             Assert.AreEqual(node.Id, node2.PreviousId);
634             Assert.AreEqual(firstName2, node2.Name);
635             Assert.AreEqual(mindmap.Id, node2.MindmapId);
636 
637             //Verify updates with and without previous
638             Assert.IsFalse(DB.UpdateNode(secondName, node2, node), "cannot␣↩
                    change␣previous␣to␣root");
639             Assert.IsTrue(DB.UpdateNode(secondName, null, node));
640             Assert.IsFalse(DB.UpdateNode(secondName2, null, node2), "cannot␣↩
                    change␣node␣to␣root");
641             Assert.IsTrue(DB.UpdateNode(secondName2, node, node2));
642 
643 
644             //Verify delete
645             Assert.IsFalse(DB.Delete(node), "cannot␣delete␣root");
646             Assert.IsTrue(DB.Delete(node2));
647 
648             DB.Delete(mindmap);
649             DB.Delete(project);
650         }
651     }
652 }
```

Listing 86 – ../../Scrum'o'wall/Scrum'o'wallTests/DBTests.cs