GameLobby service

We definiëren in onze service 2 datacontracten om onze GameLobby en de daarbijhorende Players voor te stellen:

```
[DataContract]
public class GameLobby
{
    public GameLobby(string plname, string lobbyname)
    {
        HostPlayer = new Player() { Naam = plname };
        Players= new List<Player>();
        Players.Add(HostPlayer);
        this.LobbyName = lobbyname;
        IsWaitingForPlayers = true;

    }
    [DataMember]
    public string LobbyName { get; set; }
    [DataMember]
    public List<Player> Players { get; set; }
    [DataMember]
    public Player HostPlayer { get; private set; }
    [DataMember]
    public bool IsWaitingForPlayers { get; set; }
}

[DataContract]
public class Player
{
    [DataMember]
    public string Naam { get; set; }
}
```

OperationContracts

In de service definieren we vervolgens enkele methoden om de lobbies te kunnen aanmaken, tonen en joinen.

Eerst maken we een Lijst aan waarin we alle lobbies in zullen bewaren. We maken deze static zodat alle clients die met onze service verbinden dezelfde lobbies te zien krijgen:

```
private static List<GameLobby> gameLobbies= new List<GameLobby>();
```

Vervolgens maken we een method om alle lobbies te weten te komen. Ter illustratie voegen we een filter toe waarbij we enkel de lobbies teruggeven die ffectief nog op spelers wachten (denk bijvoorbeeld aan lobbies die reeds volzet zijn maar nog niet aan hun spel zijn begonnen:

```
[OperationContract]
 public IEnumerable<GameLobby> GetLobbies()
{
    var waiting = from p in gameLobbies where p.IsWaitingForPlayers select p;
     return waiting;
 }
```

Dan maken we een method die spelers toelaat hun eigen lobby aan te maken.Hierbij dient de speler z'n eigen naam, alsook die van de lobby mee te geven (extra code zou dan kunnen voorzien dat er geen dubbele lobbynamen ontstaan en ook dat de speler z'n id meegeeft ipv username)

```
[OperationContract]
public bool CreateLobby(string playername, string lobbyname)
{
    GameLobby lob= new GameLobby(playername,lobbyname);
    gameLobbies.Add(lob);

    return true;
}
```

Indien een speler een lobby wil joinen dan kan dat door de naam naar de juiste lobby op te geven (zie client zijde straks). We zoeken vervolgens in de lobbies of deze bestaat, zo ja dan voegen we de speler toe aan deze lobby (uitbreiding kan zijn dat er gecontroleerd wordt of de lobby niet reeds aan een spel is begonnen)

```
[OperationContract]
public bool EnterLobby(string playername, string gamelobbytoJoin)
{
    if (gamelobbytoJoin != null)
    {
      var lob =from p in gameLobbies where gamelobbytoJoin == p.LobbyName select p).FirstOrDefault();
        if (lob != null)
        {
            lob.Players.Add(new Player() {Naam = playername});
            return true;
        }
    }

    return false;
}
```

GameState

Opmerking: de GameLobby kan je vervolgens uitbreiden met ook een GameState object wanneer het spel gaan beginnen. Als volgt. We maken een GameState klasse , die bijvoorbeeld nu enkel bijhoudt in welke ronde we zijn en welke speler aan de beurt is:

```
public class GameState
{
    public int CurrentRound { get; set; }
    public Player ActivePlayer { get; set; }
}
```

Vervolgens voegen we deze informatie toe aan de GameLobby klasse, bv (merk op dat we de gamestate niet van buiten de klasse GameLobby willen kunnen aanpassen en we deze dus readonly maken):

```
[DataMember]
public GameState GameState { get; private set; }
[DataMember]
public bool GameIsRunning { get; set; }
```

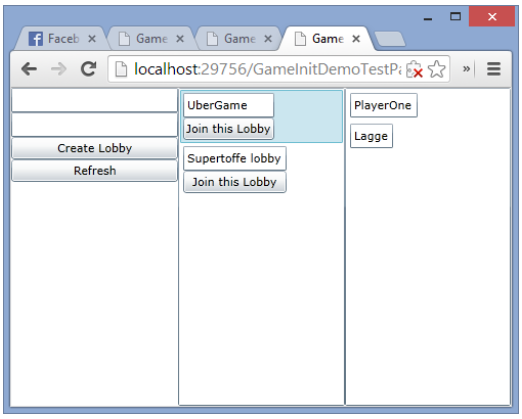
Binnen de GameLobby klasse willen we dan ook een method die het spel zal initialiseren vanaf de lobbyspelers besluiten dat ze willen beginnen:

```
public bool StartTheGame()
{
    GameIsRunning = true;
    IsWaitingForPlayers = false;
    GameState= new GameState();
    GameState.CurrentRound = 1;
    GameState.ActivePlayer = Players.First();
    return true;
}
```

We zouden dan aan onze service deze methode kunnen toevoegen:

```
[OperationContract]
public bool StartGame(string gameLobby)
{
    var lob =
       (from p in gameLobbies where gameLobby == p.LobbyName select p).FirstOrDefault();
    if (lob != null)
       return  lob.StartTheGame();
    return false;
}
```

Client



We tonen enkel hoe een potentiele client er zou kunnen uitzien.

XAML:

```
<Grid x:Name="LayoutRoot" Background="White" >
    <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
   <TextBlock Visibility="Collapsed" Grid.Row="1" Name="wait">Please wait</TextBlock>
    <StackPanel Grid.Column="0">
        <TextBox Name="playerNametb"></TextBox>
        <TextBox Name="lobbyNametb"></TextBox>
        <Button Name="createLobbyBTN" Click="CreateLobbyBTN_OnClick">Create Lobby</Button>
        <Button Name="refreshBtn" Click="RefreshBtn_OnClick">Refresh</Button>
    </StackPanel>
    <ListBox Grid.Column="1" Name="lobbieslb" SelectionChanged="Lobbieslb_OnSelectionChanged">
        <ListBox.ItemTemplate>
            <DataTemplate>
                <StackPanel>
                <TextBox Text="{Binding LobbyName}"></TextBox>
                    <Button Name="JoinLobbyBtb" Click="JoinLobbyBtb_OnClick">Join this Lobby</Button>
                </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
    <ListBox Grid.Column="2" Name="playerinlobbylb">
        <ListBox.ItemTemplate>
            <DataTemplate><TextBox Text="{Binding Naam}"></TextBox></DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
</Grid>
```

Codebehind:

```
        private GameServiceClient client;
        public MainPage()
        {
            InitializeComponent();
            client = new GameServiceClient();
            UpdateLobbies();
        }

        private void CreateLobbyBTN_OnClick(object sender, RoutedEventArgs e)
        {
            wait.Visibility = Visibility.Visible;
            client.CreateLobbyCompleted += (se, ea) =>
            {
                wait.Visibility = Visibility.Collapsed;
                UpdateLobbies();
            };
            client.CreateLobbyAsync(playerNametb.Text, lobbyNametb.Text);
        }

        private void UpdateLobbies()
        {
            wait.Visibility = Visibility.Visible;
            client.GetLobbiesCompleted += (se, ea) =>
                {
                    wait.Visibility = Visibility.Collapsed;
                    lobbieslb.ItemsSource = ea.Result;
                };
            client.GetLobbiesAsync();
        }

        private void RefreshBtn_OnClick(object sender, RoutedEventArgs e)
        {
            UpdateLobbies();
        }

        private void Lobbieslb_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (lobbieslb.SelectedItem != null)
                playerinlobbylb.ItemsSource = (lobbieslb.SelectedItem as GameLobby).Players;
        }

        private void JoinLobbyBtb_OnClick(object sender, RoutedEventArgs e)
        {
            if (playerNametb.Text == string.Empty)
                MessageBox.Show("Gelieve een geldige naam in te geven");
            else
            {
                client.EnterLobbyCompleted += (es, ea) => { UpdateLobbies(); };
                client.EnterLobbyAsync(playerNametb.Text, ((sender as FrameworkElement).DataContext as
GameLobby).LobbyName);
            }
        }
```