Design document

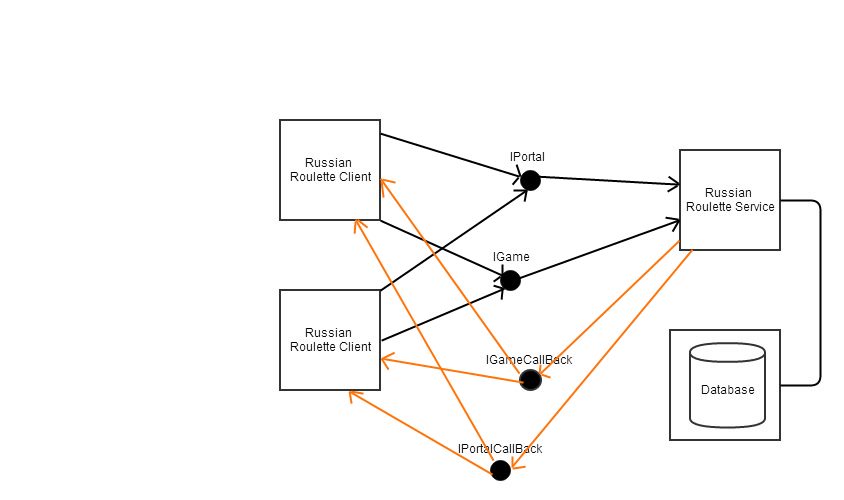
Russian Roulette

Zigmas Slusnys, Preslav Petkov, Aleksandar Asenov

Service and Client application design document

|  |
| --- |
| Contents Architecture diagram……………………………………………………………………………………………………………………………….2  Description of Interface…………………………………………………………………………………………………………………………..2  IPortal……………………………………………………………………………………………………………………………………………………2  Methods …..……………………………………………………………………………………………………………………………………….2  SignUp().………………………………………………………………………………………………………………………………………….2  SignIn().…….……………………………………………………………………………………………………………………………………..3  SendPortalMessage()……………………………………………………………………………………………………………………….3  InviteToPlay()…………………………………………………………………………………………………………………………………..3  AgreeToPlay()………………………………………………………………………………………………………………………………….3  EventsAndCallbacks()………………………………………………………………………………………………………………………….3  OnUserSignIn()…………………………………………………………………………………………………………………………………3  OnUserSignOut()……………………………………………………………………………………………………………………………..3  OnMessageSent()…………………………………………………………………………………………………………………………….4  IGame……………………………………………………………………………………………………………………………………………………4  Methods…………………………………………………………………………………………………………………………….………………4  Play()……………………………………………………………………………………………………………………………………………….4  PlaceBullet()…………………………………………………………………………………………………………………………………….4  SpinCylinder()………………………………………………………………………………………………………………………………….4  Shoot()…………………………………………………………………………………………………………………………………………….5  SendMessage…………………………………………………………………………………………………………………………………..5  Rematch………………………………………………………………………………………………………………………………………….5  EventsAndCallbacks……………………………………………………………………………………………………………………………6  PlayerSentMessage………………………………………………………………………………………………………………………….6  PlayerDisconnected…………………………………………………………………………………………………………………………6  PlayerReady…………………………………………………………………………………………………………………………………….6  PlayerLost………………………………………………………………………………………………………………………………………..6  Class Diagram for Clients…………………………………………………………………………………………………………………………7  Class Diagram for Service…………………………………………………………………………………………………………………………7  Sequence diagrams for MUST use-cases………………………………………………………………………………………………….7 |

# Architecture diagram

This diagram shows two instances of client applications and one instance of service application of the Russian roulette game. Both clients will communicate with both interfaces from the server side.

# Description of Interfaces

## IPortal

Portal interface is used at the very client application beginning when the user is requested to sign up or sign in to gain further functionality of the application. The main idea to have a portal is that the application would have possibility to have different users, give them the option to chat in the public or private chats and invite each other for a friendly match of a Russian roulette game.

### Methods

#### SignUp()

This method will allow new users to sign up to the portal in order to sign in to the portal later on.

The method will use User class as a parameter in order to sign the user in with the data the user entered himself in the registration form.

The method returns a string value noticing the user if he has successfully registered or an error has interrupted the registration.

#### SignIn()

The method will allow already existing user in the database to sign in to the portal and use the functionality and GUI that only signed up users are provided with.

As parameter the method uses User class as it provides with all the data needed to sign in.

Method returns Boolean value of true if the user has entered the correct data and vice versa.

#### SendPortalMessage()

This method performs communication functions, it executes when user sends a message to the public portal channel or a private message to another user.

Parameters that the method uses differ by what user intentions are.  
If the user intention is to send a message to the public chat then the method uses Message class.  
If the user intention is to send a private message to the user, then the method uses User and Message classes.

Method does not return anything as it is of type void.

#### InviteToPlay()

The method is used by a user to send a request of a gameplay to another user.

The parameters are User class, as it will use the Nickname of a user to send a request.

Returns nothing as the method is of type void.

#### AgreeToPlay()

The method is used to agree to a received request to play of another user.

Method does not use any parameters.

Method returns nothing because of type void.

### Events and Callbacks

#### OnUserSignIn()

This callback fires when user signs in, it tells all the online users that a new user has signed in and updates the user list with a new nickname and a chat list saying that a new user signed in.

Use User class as a parameter to get the Nickname of a signed in user.

Returns nothing as it is a type of void.

#### OnUserSignOut()

Every online user gets a callback when an existing user signs out of the portal, the nickname of leaving user is removed from the list of online users and the chat list is updated with a certain user left message.

Use User class as a parameter.

Returns nothing as the type of the callback is void.

#### OnPublicMessageSent() & OnPrivateMessageSent()

This callback is executed when a user sends a message to a public portal chat or a private message to another user.

The parameters that the method use depends on what kind of message the user wants to send.  
If user wants to send a public chat message, then it uses only Message class.  
If user wants to send a private message, it uses Message and User classes.

Returns nothing as the type of method is Void.

#### OnInvitedToPlay()

**Meaning:** When a user sends an invitation to a player, the callback goes through all the users finding the one that has a certain nickname and performs an invitation on his side.

**Trigger:** Invitation to play a game.

**Parameters:** User class nickname.

## IGame

### Methods

#### Play()

**Description:**  
This method is called when a user is ready to play the game. This method calls a callback event that results in telling other player that the user is ready.

**Parameters:**   
The method requires no parameters as it only signals other player about being ready.

**Return value:**Returns nothing because of type void.

#### SpinCylinder()

**Description:**Pre-Conditions: The bullet has to be placed in the cylinder.

This method is called by the player that has been choosing the place of bullet – it shuffles the array of Boolean values of the cylinder and makes a random placement of the bullet that the user doesn’t know about.

The method could send a callback event to the players that the cylinder has been spun.

**Returns:**  
This method returns string value that the method succeeded spinning the bullet.

#### Shoot()

**Description:**Pre-Conditions:   
The bullet has to be placed and cylinder has to be spun.

When user triggers a gun to fire, the method uses incremental value of bullet holes to check if the cylinder hole has a bullet or is empty. If the hole has been empty, the turn to shoot goes to other user, else the gun shoots with a bullet, kills the player and send a callback to other player that he has won.

**Parameters:**None.

**Returns:**  
The method returns Boolean value true if the player has dies or false if the cylinder hole was empty.

#### SendMessage()

**Description:**

Pre-Conditions: None.

This method sends a message to a public game chat between two players.

**Parameters:**  
Message class object.

**Returns:**  
Type void.

### Events and Callbacks

#### PlayerSentMessage()

**Meaning:**The callback signals that the user has sent a message and updates the chatbox lists with the sent message data.

**Trigger:**User has to send a message.

**Parameters:**Message class object.

#### PlayerDisconnected()

**Meaning:**Player gets a notice that the opponent player has left the game.

**Trigger:**Left player from the game.

**Parameters:**Could be User class object.

#### PlayerReady()

**Meaning:** Calls back when opponent player is ready to play the game.

**Trigger:** Opponent player has to click a button ready.

**Parameters:**

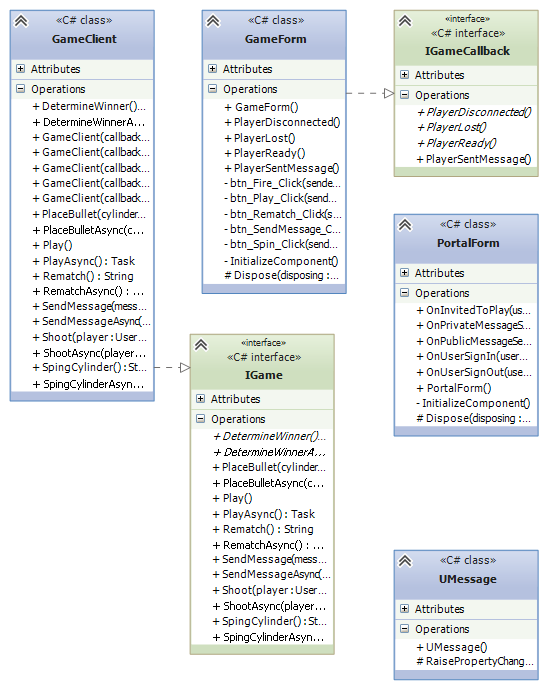
#### PlayerLost()

**Meaning:**Sends an notice that the opponent player has lost.

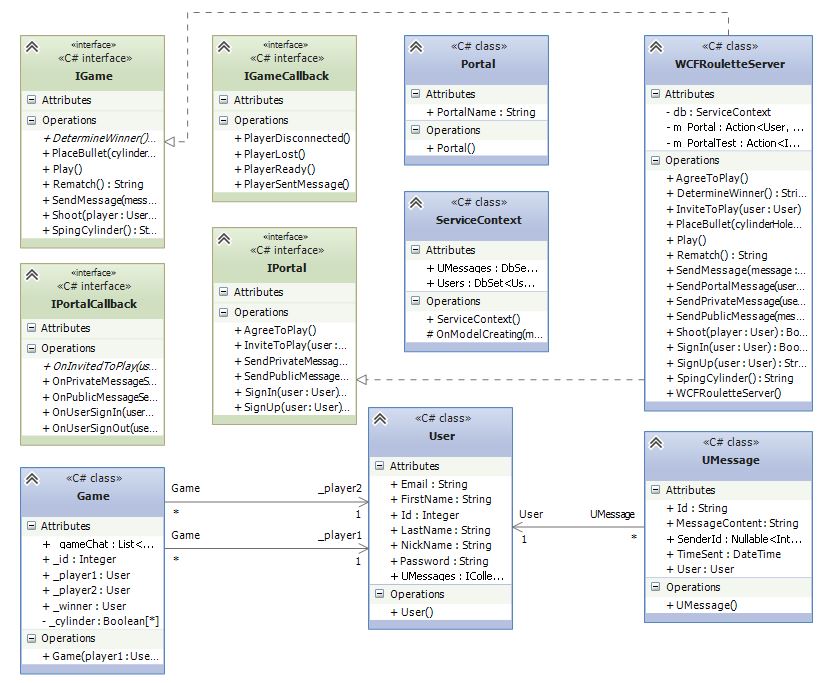
**Trigger:**Player shots himself with a bullet from the cylinder.

**Parameters:**Could be User class object containing Nickname of the user.

# Class diagram for client



# Class diagram for service



# Sequence diagrams for MUST use-cases

