Design document

MDW – Ludo game

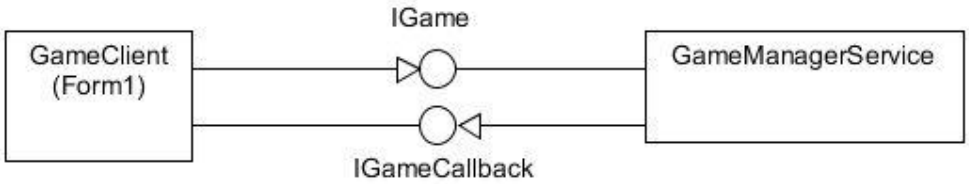
FONTYS UNIVERISTY OF APPLIED SCIEENCES

June 26, 2016

Authored by: Monica Stoica, Dimitar Vikentiev, Rosen Danev

Table of Contents

Architecture diagram



Description of Interfaces

IGame:

All the methods in the class are of type [OperationContract]

```
List<Player> Connect(string username, int nbOfPoints, int Position, int priority);
```

This method will be called each time when a person enters the game and that user will be connected. His/her information, such as username, will be shown in a certain list box in the main form.

```
void Disconnect();
```

This method one will be used when a player exists the game and the he/she will be disconnected. The player will also be removed from the list box and the other players will be informed.

```
void CreatePlayer(string username, int nbOfPoints, int position, int priority);
```

```
List<Player> GetPlayers();
```

It will return a string with all information of a certain player depending on his/her username.

```
string GetPlayerInfo(string playerName);
```

```
int ThrowDice();
```

Will return a random integer from 1 to 6.

```
void MoveForward(Pawn p, int newPosition);
```

When this method is called the pawn will move forward according to the diceresult.

```
void SendMessage(string msg, string playerName);
```

Used for communication between players.

```
void GamePause(bool b);
```

Pauses/Resumes the game for all players.

```
void ResetPawn(Pawn p);
```

Respawns a pawn.

```
bool AllFinish(string playerName);
```

Checks if all the pawns of a player have finished.

```
void PawnHasFinish(Pawn pawn, string playerName);
```

Sets the final position of the pawn.

```
void NotifyForWinner(string playerName);
```

Notifies other players that there is a winner.

```
void setTurn();
```

Sets the turn of the player based on priority.

```
bool getTurn(string player, Pawn pawn);
```

Checks if the current player is allowed to make a move.

```
bool diceTurn(string player);
```

Checks if the current player is allowed to roll the die.

IGameCallback:

```
void NewPlayerConnected(List<Player> players);
```

This method will inform other players when a new player is connected.

```
void MessageRecieved(string msg, string playerName);
```

Sends the message to all the players in the game.

```
void DiceNotify(int result);
```

Notifies other players about the dice result.

```
void PawnNotify(Pawn p, int newPosition);
```

Notifies the other players about the new position of the pawn.

```
void GamePaused(bool b);
```

Notifies the other players that the game has been paused/resumed.

```
voi d NewWinner(string username);
```

The method will inform other players when some of them has finished and will show his/her place (ranking) among other players.

```
void RespawNotify(Pawn p);
```

Notifies the other players that the pawn has been respawned.

IAccount:

```
string LogIn(string username, string password);
```

Checks if the combination of username and password matches.

```
bool createAccount(string name, string username, string password);
```

Checks if there is already a user with that name and if not, it creates a new database entry.

```
List<string> getRanking();
```

Returns a list of the players' username and their points.

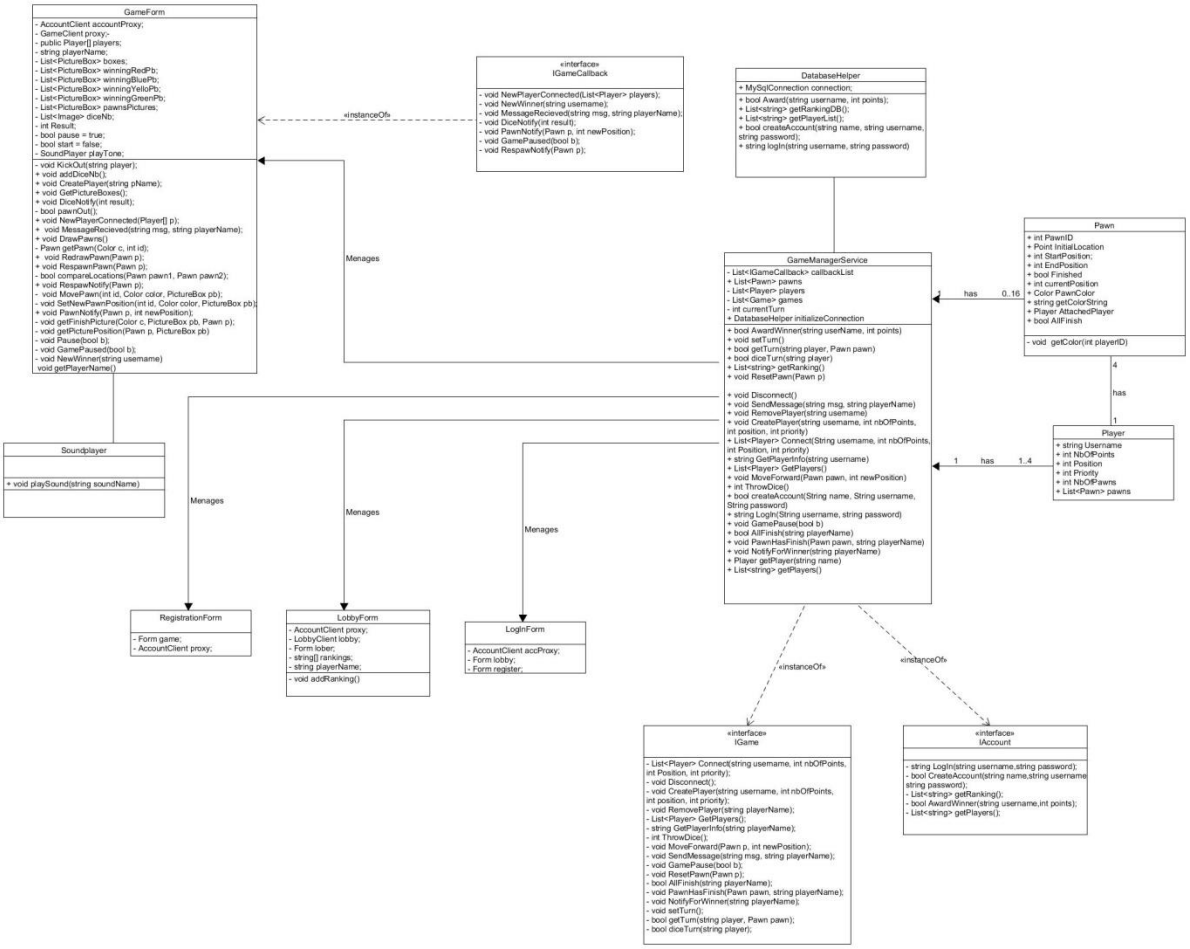
```
bool AwardWinner(string userName, int points);
```

Adds points to a player when winning the game.

```
List<string> getPlayers();
```

Returns a list with the username of the players.

Class Diagram



Below, you can find a description of the fields, properties and methods used in our Class diagram.

Class Player

Properties:

- Username: string – represents the name of the player
- NbOfPoint: int – represents the number of points accumulated by a player
- Position: int – represents the position of the player compared to other players
- Priority: int – represents the priority of the player when starting a new game (e.g. who will roll the die first)
- NbOfPawns: int – represents the number of pawns the player needs to bring to the finish line in order to win
- PawnColor: int – represents the colour of the pawns of a player
- Pawns : List<Pawn> - stores all the pawns of a player

Constructor:

- Player(string username, int points, int position, int priority, Color color)

Class Pawn

Properties:

- PawnID: int – the unique identification of a pawn
- Status: status – the current status of a pawn
- PawnImage: Image – the representative image of a pawn
- StartPosition: int – the starting position of the pawn
- EndPosition: int – the ending position of the pawn
- CurrentPosition: int – the current position of the pawn
- GetColorString: string – the color of the pawn
- AttachedPlayer: Player -the player to whom the pawn belongs to.
-

Constructor:

- Pawn(int ID);

Methods:

- getColor(int playerID): void – sets the color of the player

Class GameManagerService

Properties:

- Players: List<Player> - list of all players
- Callbacks: List<IGameCallBack> - list with all callbacks
- Pawns: List<Pawn> - list with all pawns
- Games: List<Game> – list with all games
- currentTurn: int – stores the current turn

Methods:

- AwardWinner(string userName, int points): bool – awards the winner with points.
- setTurn(): void – Sets the turn for players
- getTurn(string player, Pawn pawn): bool – check if the current player can move bases on priority. It also checks if the pawn is owned by the player.
- diceTurn (string player) :bool – check if the player is allowed to roll the die.
- getRanking(): List<string> - list of all player's ranking from the database.
- ResetPawn (Pawn p): void – resets the pawn to its initial position.
- CreatePlayer (string username, int nbOfPoints,
  int position, int priority): void – add a new player to the list.
- RemovePlayer (string username): void – removes player from the list.
- Connect (String username, int nbOfPoints,
  int Position, int priority): List<Player> - connects a new user.
- Disconnect: void – disconnects a player.
- SendMessage (string msg, string playerName): void – sends a message to other players.
- GetPlayerInfo (string username): string – returns information about a player.
- GetPlayers(): List<Player> - list of all connected players.
- MoveForward (Pawn pawn, int newPosition): void – Notifies other player that a pawn has moved.
- ThrowDice: int – generates a random number between 1 and 6.
- LogIn (String username, string password): string – logs in a user.

- createAccount (String name, String username,
- String password): bool – registers a new user.
- GamePaused (bool b) : void – Pauses the game for all users.
- AllFinish (string playerName): bool – checks if all the pawns of a player have reached the final
- PawnHasFinish (Pawn pawn, string playerName) : void – checks if a pawn has finished.
- NotifyForWinner (string playerName): void – notifies other players that there is a new winner.
- getPlayer (string name): Player – returns a player.
- List<string> getPlayers() – returns a list of players

Class DatabaseHelper

Properties:

- Connection: MySQLConnection – initializes database connection.

Methods:

- Award (string username, int points): bool – gives award points to a given player into the database.
- getRankingDB(): List<string> - gets the ranking of the players and puts them in a list
- getPlayerList(): List<string> - gets the players from the database and puts them in a list.
- createAccount (string name, string username,
  string password): bool – registers a player into the database.
- logIn (string username, string password): string – checks if the given username and password exists in the database.

Class Soundplayer

Methods:

- PlaySound(string SoundName): void – plays a sound

Class GameForm

Fields:

- AccountClient accountProxy
- GameClient proxy
- Player[] players
- string playerName
- List<PictureBox> boxes – list with all pictureboxes that represent the board
- List<PictureBox> winningRedPb
- List<PictureBox> winningBluePb
- List<PictureBox> winningYelloPb
- List<PictureBox> winningGreenPb
- List<PictureBox> pawnsPictures
- List<Image> diceNb
- int Result
- bool pause = true

- bool start = false
- SoundPlayer playTone

Methods:

- void KickOut(string player) – checks if there are already four players in the game. If yes, the current player is not allowed to join.
- void addDiceNb() – adds the pictures for the dice according to the number rolled.
- void CreatePlayer(string pName) – creates a new player.
- void GetPictureBoxes() – adds all the pictureboxes to the lists.
- bool pawnOut() – respawns the pawn
- void DrawPawns() – draws the pictures of the pawns for each player
- Pawn getPawn(Color c, int id) – returns a pawn based on its colour and id
- void RedrawPawn(Pawn p) – redraws the pawn on the screen
- void RespawnPawn(Pawn p) – respawns the pawn
- bool compareLocations(Pawn pawn1, Pawn pawn2);
- void RespawNotify(Pawn p) – respawns the pawn
- void MovePawn(int id, Color color, PictureBox pb) – move the pawn
- void SetNewPawnPosition(int id, Color color, PictureBox pb) – sets the new position of the pawn
- void getFinishPicture(Color c, PictureBox pb, Pawn p);
- void getPicturePosition(Pawn p, PictureBox pb)
- void Pause(bool b) – pauses the game by disabling the functionality.
- void getPlayerName() – returns the name of the player.

Class LoginForm

Fields:

- AccountClient accProxy;
- Form lobby;
- Form register;

Class LobbyForm:

Fields:

- AccountClient proxy;
- LobbyClient lobby;
- Form lober;
- string[] rankings;
- string playerName;

Methods:

- - void addRanking() - adds the ranking from the database to a local list.

RegistrationForm

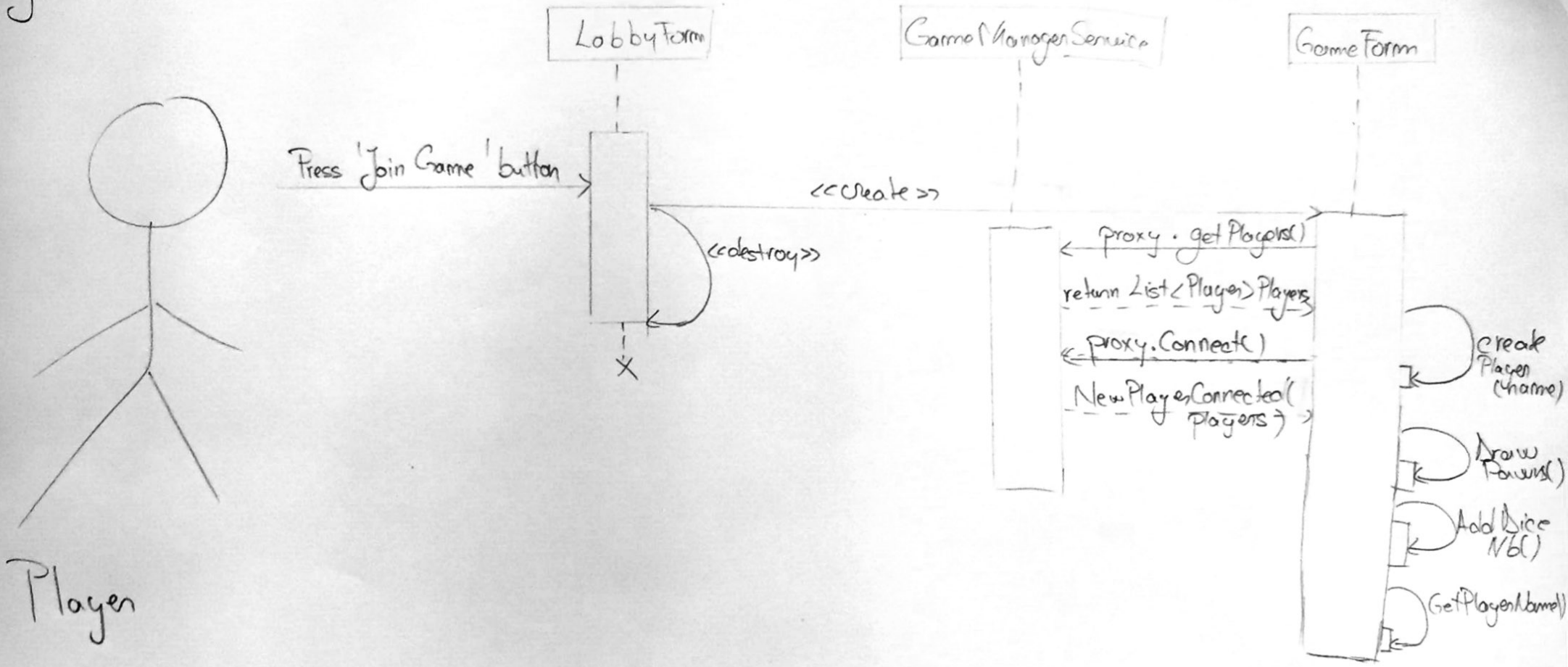
Fields:

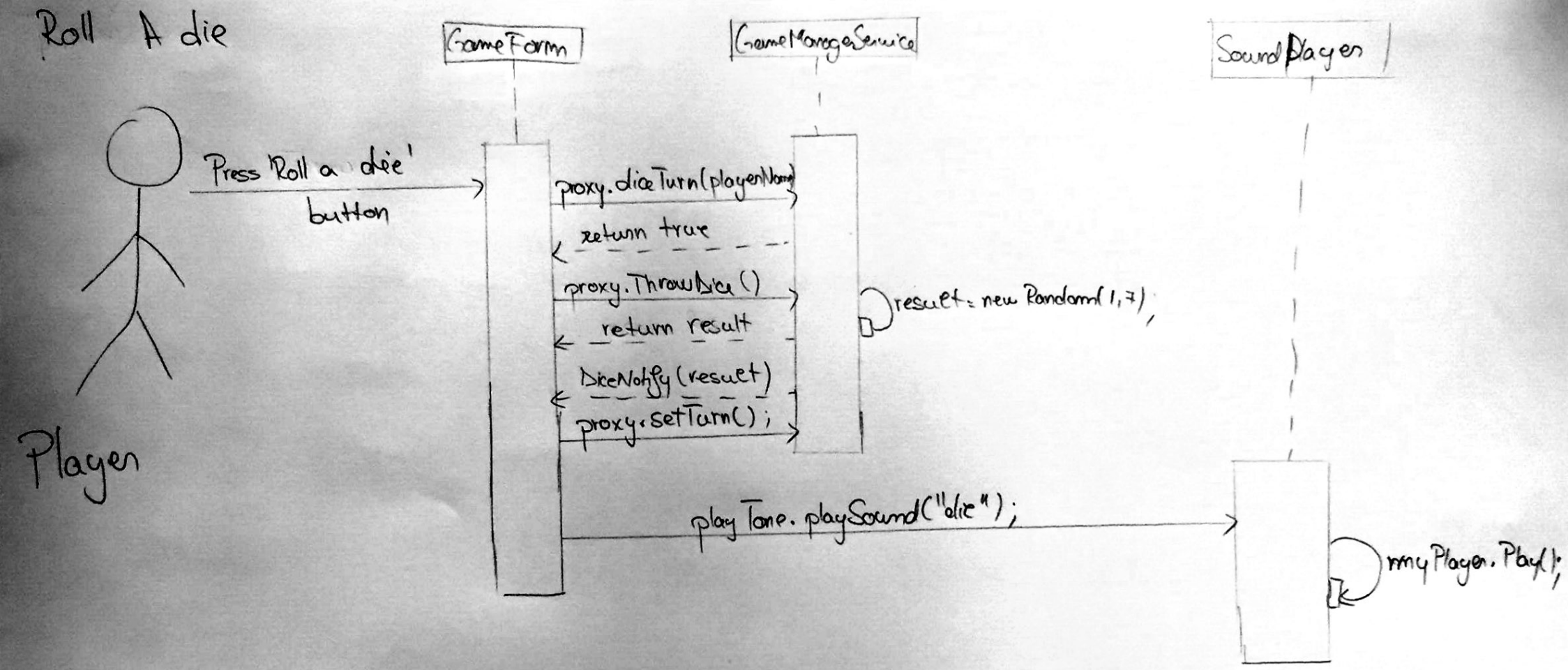
- Form game;
- AccountClient proxy;

Database

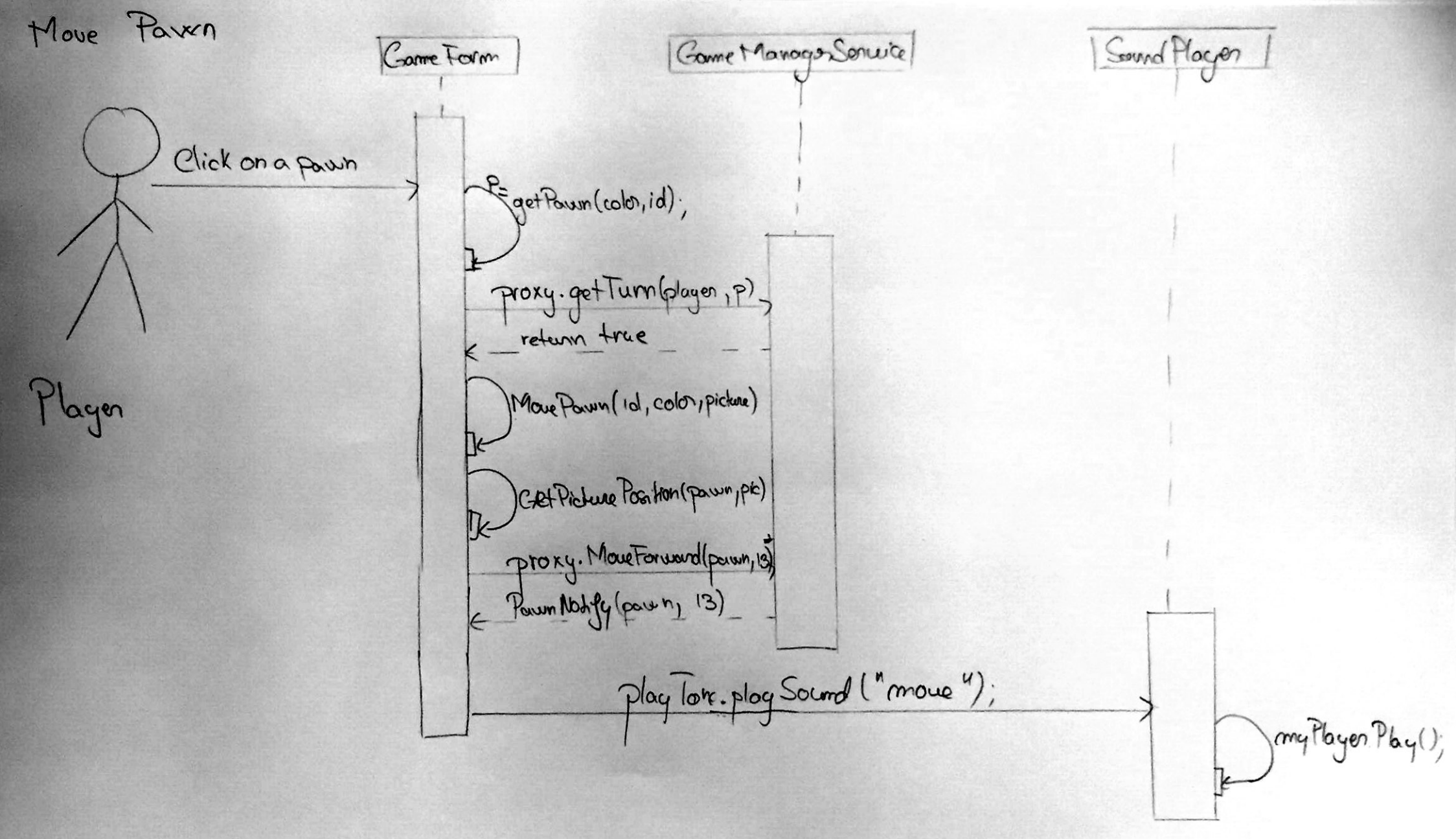
The diagram represents the database design which will be used in order to implement the Sign in, Log in and Ranking functionalities. The entity, stores the information about a user such as username(string), password (string), Name (string) and the points (int). The primary key is username since there cannot be two players with the same username.

Join Game

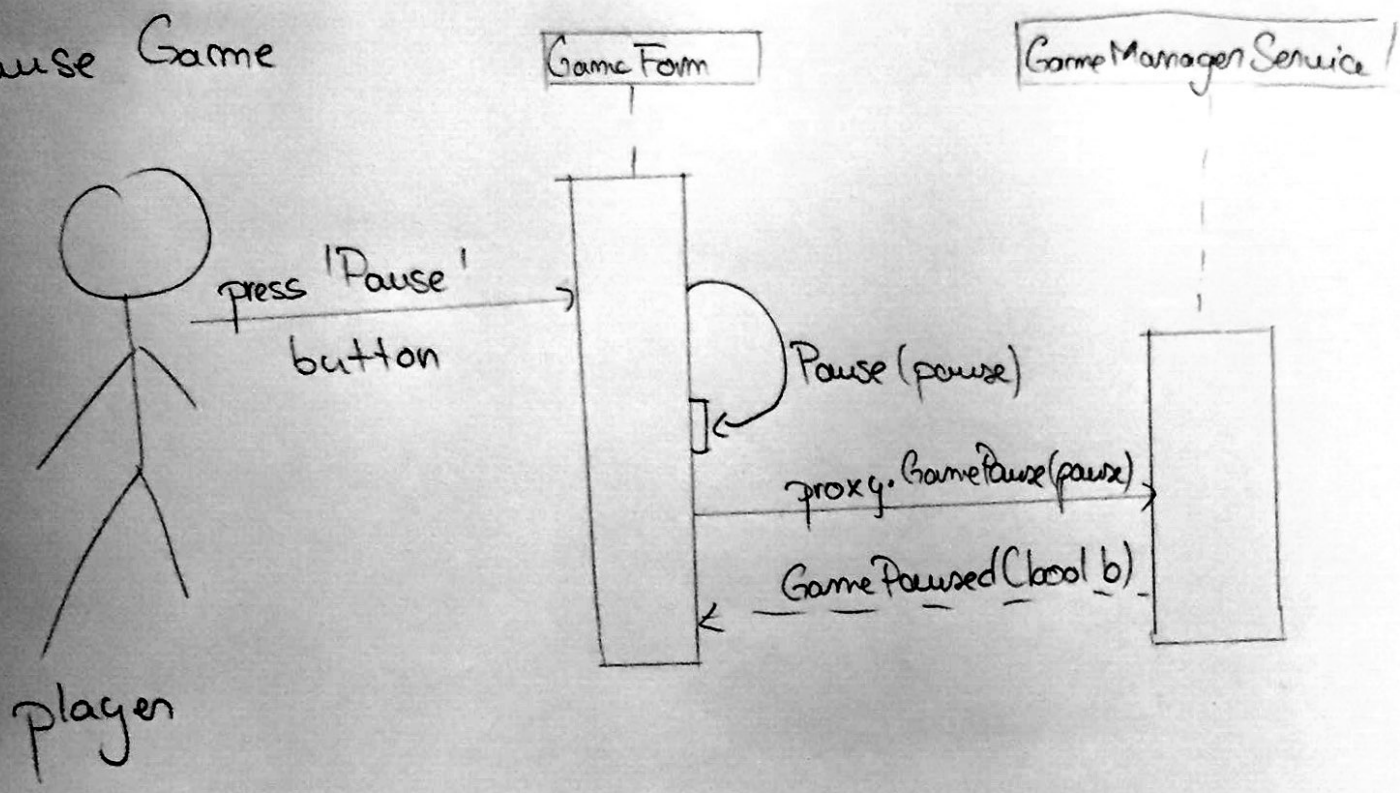

LobbyForm
GameManagerService
GameForm
Player
Press 'Join Game' button
<<create>>
<<destroy>>
proxy.getPlayers()
return List<Player> Players
proxy.Connect()
NewPlayerConnected(players)
Create Player (name)
Draw Players()
Add Dice NO()
GetPlayerName()

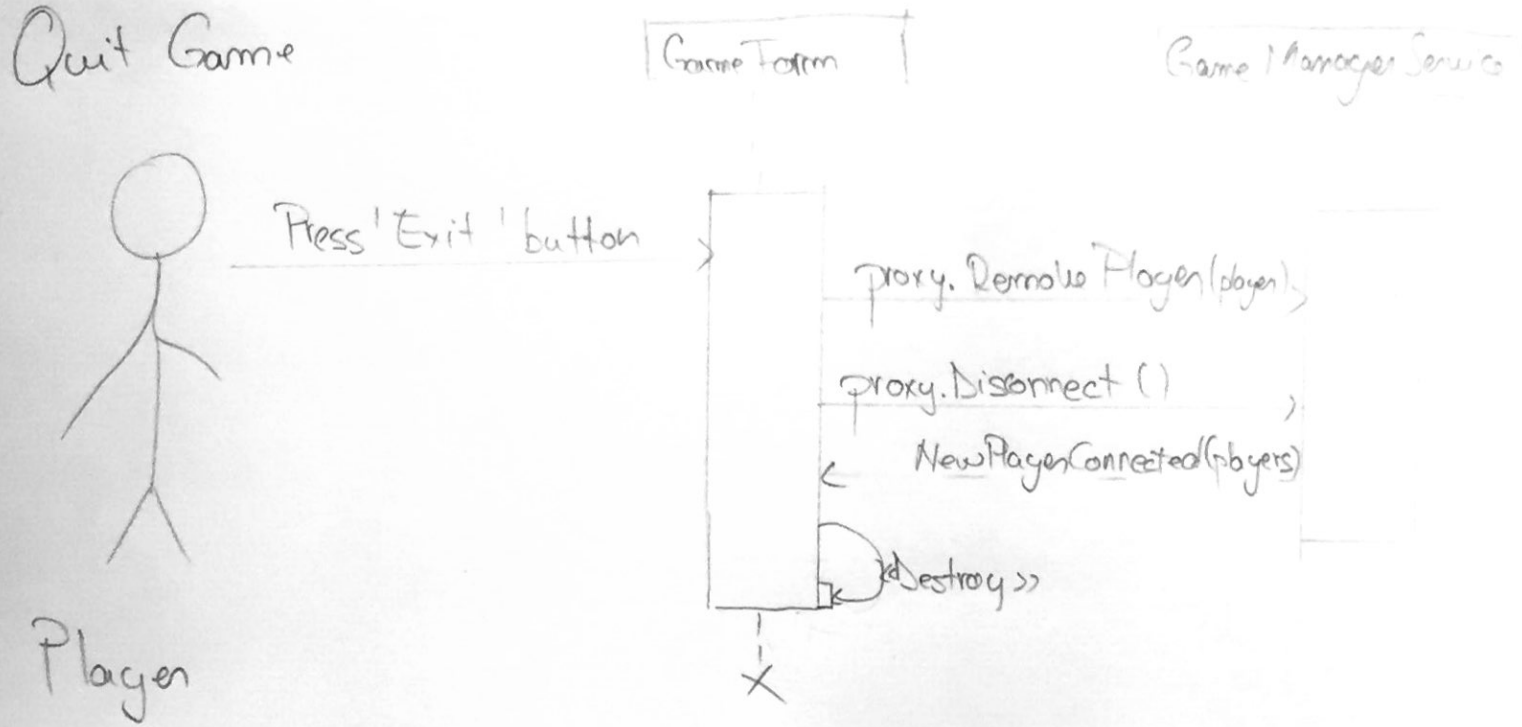

Roll A die
GameForm
GameManagerService
SoundPlayer
Player
Press 'Roll a die' button
proxy.diceTurn(playerName)
return true
proxy.ThrowDice()
return result
result = new Random(1,7);
DiceNotify(result)
proxy.setTurn();
playTone.playSound("die");
myPlayer.Play();

Move Pawn



* 13 is the new Position


Pause Game
GameForm
GameManagerService
press 'Pause'
button
Pause(pause)
proxy.GamePause(paus)
GamePaused(bool b)
player


Quit Game
Game Form
Game Manager Service
Press 'Exit' button
proxy.RemovePlayer(player)
proxy.Disconnect()
NewPlayerConnected(players)
Destroy
Player