**Documento técnico**

**Objetivo**

El objetivo de este documento es hacer más fácil la comprensión de nuestro proyecto, detallando funcionalidades, aspectos generales, etc.

**Aspectos generales**

El proyecto es una recreación del clásico juego para la NES Arkanoid, pero le hemos añadido nuestro toque personal, el aspecto visual de nuestro proyecto se inspiró en gran parte por los juegos de Mario Bros, esto se hace obvio desde el menú principal, con el tema de super Mario y escenario ambientado por uno de los niveles más icónicos.

Con respecto a la propia jugabilidad, esta es bastante similar al Arkanoid original, una nave, una pelota (ahora Mario) y muchos ladrillos. El objetivo del juego es destruir todos los ladrillos que aparecen en pantalla, utilizando la pelota y la nave.

Si la pelota cae fuera de los límites de la pantalla, pierdes una vida, cuentas con una cantidad de vidas limitadas, si las vidas se te acaban, naturalmente, pierdes el juego.

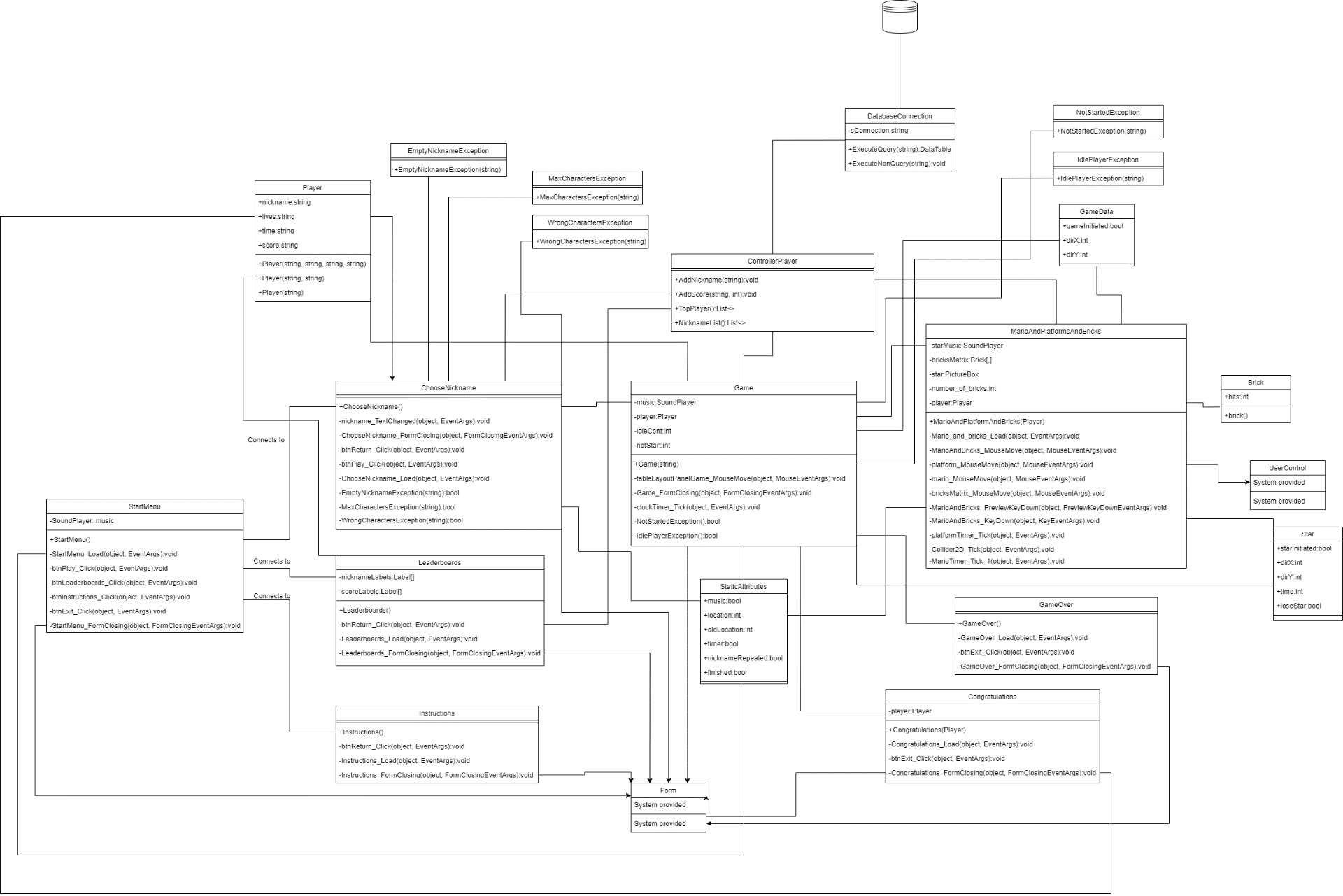
Una vez destruido todos los ladrillos (algunos más resistentes que otros), procedes a una pantalla felicitándote por la hazaña, esta pantalla está acompañada por el nombre del jugador y su respectivo puntaje.

Cabe destacar, que el puntaje se calcula con respecto al tiempo, la suma de puntos al romper bloques (100 por cada golpe) y la cantidad de vidas restantes.

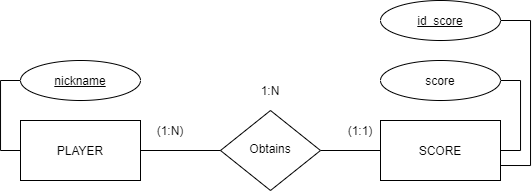
**Software utilizado**

El software utilizado para realizar este proyecto, fue Rider, utilizando Microsoft .NET Framework 4.8, en conjunto de PostgreSQL mediante la aplicación de pgAdmin 4 (para tener una conexión local), complemento extra para Rider fue Npgsql, desarrollado con utilización de formularios, métodos, interfaz gráfica, eventos y UserControls, a su vez se han tomado las excepciones para tomar en cuenta errores, que sin la implementación de ellos no se podrían resolver.

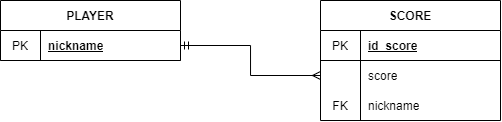
**UML:**

****

**Diagrama Entidad Relación**

****

**Diagrama relacional**

****

**Conceptos técnicos y distintos tipos de error**

**Windows Forms**

1. **Congratulations**Un form dedicado a la pantalla de victoria.
2. **Game**Form dedicado a la funcionalidad del propio juego (aunque necesita del user control MarioAndPlayformsAndBricks para poder funcionar).
3. **GameOver**Form utilizado para mostrar la pantalla de derrota al perder todas las vidas.
4. **Leaderboard**

Un form dedicado para mostrar el top 10 de mejores puntajes.

1. **Instructions**Form dedicado a informar al jugador sobre cómo jugar y cómo funciona el sistema de puntos.
2. **ChooseNickname**

Un form utilizado para escribir el alias de tu personaje.

1. **StartMenu**

Form dedicado al mostrar el menú y poder acceder a todas las funciones pertinentes del programa.

**UserControl**

1. **MarioAndPlatformsAndBricks**

User control dedicado al funcionamiento del juego.

**Modelos**

1. **GameData**Clase estática encargada de la velocidad a la que se mueve mario y verificar si el juego está iniciado.
2. **Brick**Clase encargada de almacenar la “vida” que tiene cada uno de los ladrillos.
3. **Player**Clase estática encargada de almacenar todos los datos importantes del jugador (nickname, vidas, tiempo y puntaje)
4. **Star**

Clase estática con los mismos atributos de game data, encargada de dar una bonificación de velocidad al jugador una vez obtenida.

**Controladores**

1. **ControllerNickname**Clase utilizada para almacenar el nickname del jugador (sino es repetido).
2. **DatabaseConnection**Código para unir la base de datos con el programa de c# y almacenar los puntajes
3. **StaticAttributes**Sirve para identificar la localización del mouse y para poder utilizar la música del menú principal

**Nomenclaturas**

|  |  |
| --- | --- |
| Label | lbl |
| Button | btn |
| TextBox | txt |

**Eventos y excepciones**

**Eventos**

**StartMenu**1. StartMenu ()

1.1. Inicializar componentes

2. StartMenu\_Load (object sender, EventArgs e): void

2.1. Maximizar Ventana

2.2. ♪ Iniciar música ♪

2.3. Cargar Estilos

3. btnPlay\_Click (object sender, EventArgs e): void

3.1. Ir a Choose Name Form

4. btnLeaderboards\_Click (object sender, EventArgs e): void

4.1. Ir a Leaderboards Form

5. btnInstructions\_Click (object sender, EventArgs e): void

5.1. Ir a Instructions Form

6. btnExit\_Click (object sender, EventArgs e): void

6.1. Salir de la aplicación correctamente

7. StartMenu\_FormClosing (object sender, FormClosingEventArgs e): void

7.1. Salir de la aplicación abruptamente

**Instructions**1. Instructions ()

1.1. Inicializar componentes

2. Instructions\_Load (object sender, EventArgs e): void

2.1. Maximizar ventana

2.2. Cargar Estilos

3. btnReturn\_Click (object sender, EventArgs e): void

3.1. Ir a Start Menu Form

4. Instructions\_FormClosing (object sender, FormClosingEventArgs e): void

4.1. Salir de la aplicación abruptamente

**Leaderboards**1. Leaderboards ()

1.1. Inicializar componentes

2. btnReturn\_Click (object sender, EventArgs e): void

2.1. Ir a Start Menu Form

3. Leaderboards\_Load (object sender, EventArgs e): void

3.1. Maximizar ventana

3.2. Cargar estilos

3.3. Cargar puntajes

4. Leaderboards\_FormClosing (object sender, FormClosingEventArgs e): void

4.1 Salida de la aplicación abruptamente

**GameOver**

1. GameOver()

1.1. Inicializar componentes

2. GameOver\_Load (object sender, EventArgs e): void

2.1. Maximizar ventana

2.2. Cargar estilos

3. btnExit\_Click (object sender, EventArgs e): void

3.1. Salir de la aplicación correctamente

4. GameOver\_FormClosing (object sender, FormClosingEventArgs e): void

4.1 Salir de la aplicación abruptamente

**Congratulations**

1. Congratulations (Player \_player)

1.1. Inicializar componentes

2. Congratulations\_Load(object sender, EventArgs e):void

2.1. Maximizar ventana

2.2. Cargar estilos

3. btnExit\_Click(object sender, EventArgs e):void

3.1. Salir de la aplicación correctamente

4. Congratulations\_FormClosing (object sender, FormClosingEventArgs e): void

4.1 Salir de la aplicación abruptamente

**ChooseNickname**

1. ChooseNickname ()

1.1. Inicializar componentes

2. nickname\_TextChanged (object sender, EventArgs e): void

2.1. Cambia el label cuando el texto cambia

3. ChooseNickname\_FormClosing (object sender, FormClosingEventArgs e): void

3.1. Salir de la aplicación abruptamente

4. btnReturn\_Click (object sender, EventArgs e): void

4.1. Ir a Start Menu Form

5. btnPlay\_Click (object sender, EventArgs e): void

5.1. Ir a Game Form

6. ChooseNickname\_Load (object sender, EventArgs e): void

6.1. Maximizar ventana

6.2. Cargar estilos

7. EmptyNicknameException (string a): bool

7.1. Evaluar si existe una excepción de este tipo

8. MaxCharactersException (string a): bool

8.1. Evaluar si existe una excepción de este tipo

9. WrongCharactersException (string a): bool

9.1 Evaluar si existe una excepción de este tipo

**Game**

1. Game (string nickname)

1.1. ♪ Iniciar música ♪

1.2. Inicializar componentes

1.3. Maximizar Ventana

1.4. Cargar estilos

2. tableLayoutPanelGame\_MouseMove(object sender, MouseEventArgs e):void

2.1 Obtener las coordenadas del mouse en X cuando esta sobre cierta parte del tableLayout

3. Game\_FormClosing (object sender, FormClosingEventArgs e): void

3.1. Salir de la aplicación abruptamente

4. clockTimer\_Tick (object sender, EventArgs e): void

4.1. Ejecuta una línea de acciones cada segundo

5. NotStartedException (): bool

5.1. Evaluar si existe una excepción de este tipo

6. IdlePlayerException (): bool

6.1. Evaluar si existe una excepción de este tipo

**MarioAndPlatformAndBricks**

1. MarioAndPlatformAndBricks (Player \_player)

1.1. Inicializar componentes

2. Mario\_and\_bricks\_Load (object sender, EventArgs e): void

2.1. Cargar bloques

2.2. Cargar estrella

2.3. Cargar estilos

3. MarioAndBricks\_MouseMove (object sender, MouseEventArgs e): void

3.1. Obtener las coordenadas en X cuando se encuentra sobre el UserControl

4. platform\_MouseMove (object sender, MouseEventArgs e): void

4.1. Obtener las coordenadas en X cuando se encuentra sobre la plataforma

5. mario\_MouseMove (object sender, MouseEventArgs e): void

5.1 Obtener las coordenadas en X cuando se encuentra sobre mario

6. bricksMatrix\_MouseMove (object sender, MouseEventArgs e): void

7. MarioAndBricks\_PreviewKeyDown (object sender, PreviewKeyDownEventArgs e): void

7.1. Controlar plataforma mediante flechas

8. MarioAndBricks\_KeyDown (object sender, KeyEventArgs e): void

8.1. Controlar plataforma mediante A y D

8.2. Iniciar juego con espacio

9. platformTimer\_Tick (object sender, EventArgs e): void

9.1. Timer de la plataforma

10. Collider2D\_Tick (object sender, EventArgs e): void

10.1. Timer para controlar las colisiones con los ladrillos

11. MarioTimer\_Tick\_1(object sender, EventArgs e): void

11.1 Timer para controlar los movimientos de mario

**Exception**

**WrongCharactersException**Esta excepción sirve para identificar si pone un símbolo, el cual no es válido para su Nickname.

**MaxCharactersException**Esta excepción sirve para identificar si el usuario ha utilizado más caracteres de los permitidos (máximo 15) en el Nickname.

**EmptyNicknameException**Esta excepción sirve para identificar si el usuario no ha introducido ningún carácter en su Nickname.

**IdlePlayerException**Esta excepción sirve para saber si el usuario tiene inactividad.

**NotStartedException**Esta excepción sirve para recordar al usuario (si se le ha olvidado) qué botón utilizar para iniciar el juego, la excepción se activará, si pasan 15 segundos y el usuario no ha iniciado.