

Manual Técnico

Juego Arkanoid

**Integrantes**:

Ricardo Daniel Peña Rivera 00365918

Paula Daniela Zepeda Barrera 00062019

Fernando Adonay Rosa Cardoza 00022319

Daniel Ernesto Morales Siguenza 00139419

**Contenidos**

Aspectos Generales3

Objetivo del documento3

Descripción del programa3

Software Utilizado3

Modelos Utilizados4

UML Diagrama de casos de uso6

UML Diagrama de clases7

UML Diagrama Entidad Relación8

Conceptos Tecnicos 9

Implementacion de la interfaz gráfica 9

Manejo de clases (modelo) 10

Manejo de clases (controlador) 10

Plataforma Base 11

Nomenclaturas 11

Eventos y Excepciones 12

Eventos 12

Excepciones12

Aspectos Generales

* **Objetivo del documento**

Dar a conocer de una forma detallada todas las herramientas utilizadas para la creación del juego y a su vez poder crear uno en un futuro.

* **Descripción General**

El juego consiste en una nave la cual lanza una pelota, esta pelota irá destruyendo bloques hasta romperlos todos, a su vez existe un sistema de vidas este ira disminuyendo a medida que la pelota toque un área que no sea la nave, ganará al destruir todos los bloques.

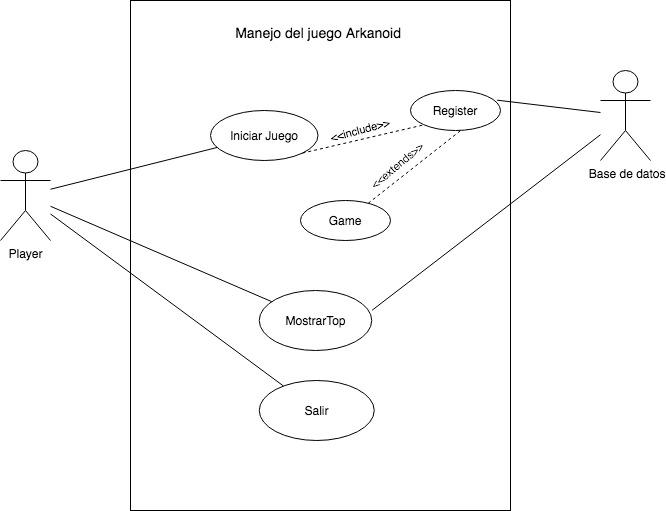
* **Software Utilizado**

Para la creación del juego, se hizo uso de Rider versión 2019.3.4, con .Net Framework 4.8, y para manejo de los datos se utilizo PostgresSQL 12 con PgAdmin4 y adicional Npgsql para la conexión a la base de datos.

Modelos Utilizados

* **UML Diagrama de casos de uso**

El código de nuestro juego, esta basado en el siguiente diagrama de casos de uso para una visualización mas entendible con el usuario:

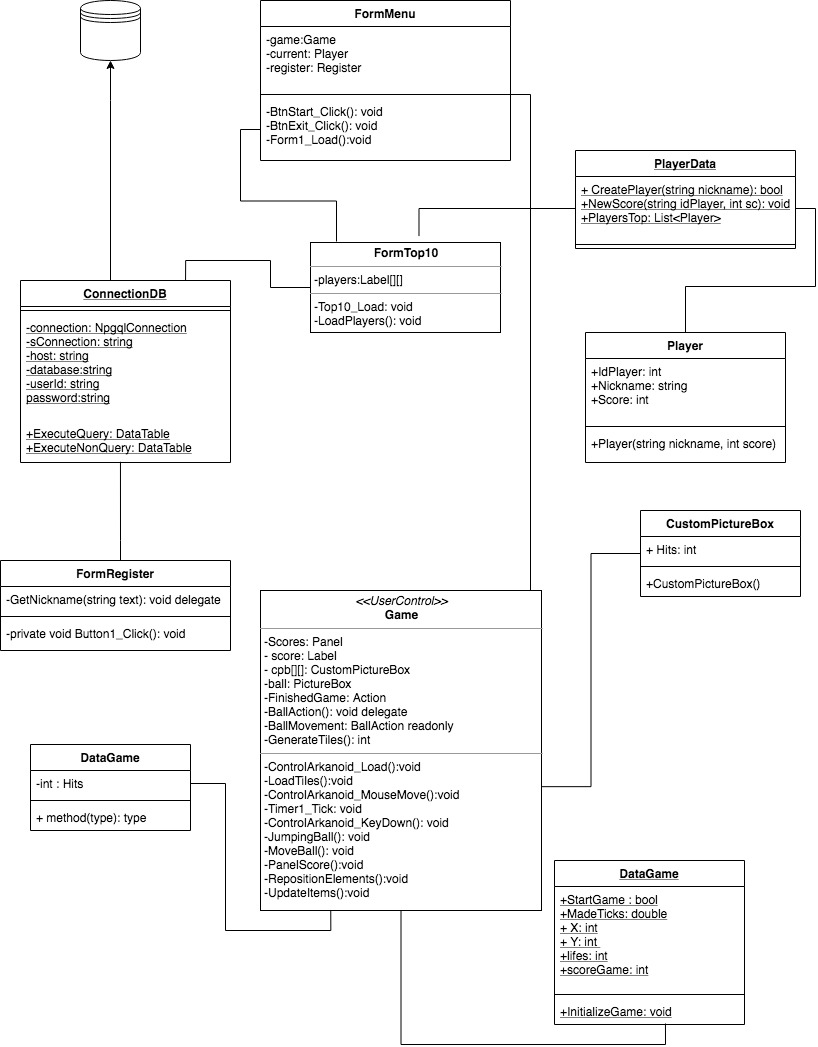


En el diagrama podemos ver como el usuario player quien es el juagador puede interactuar con 3 distintos botones, al dar clic al botón iniciar juego se mandara a un registro en este se validará si esta ya registrado o no, todo esto viene interactuado en conjunto con la base de datos quien nos brinda esa información, validada esta información el jugador puede comenzar el juego.

Tambien en el botón de mostrarTop el jugador podra visualizar quienes son los primeros 10 puntajes altos, esto viene dado por la base de datos quien brinda los datos

Y por último tenemos el botón salir el cual no interactua con la base de datos y en este se cierra la aplicación.

* **UML Diagrama de clases**



Como podemos observar en el diagrama tenemos como elemento principal el UserControl de Game.cs, en el cual tenemos los métodos necesarios para el funcionamiento del juego, como por ejemplo LoadTiles() que es un método que nos ayuda a cargar todos los bloques que usaremos.

En la clase ConnectionDB nos conectamos a la base de datos, esto ayuda a que pgAdmin se conecte a Rider.

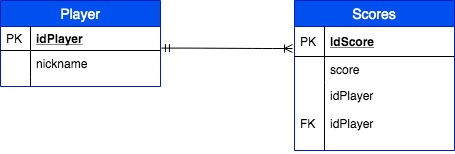
Luego tenemos FormMenu encontramos el Menú principal, en este menú tenemos 3 opciones, iniciar juego, mostrar top y salir, cada uno cambia con los demás formularios y user control, al dar clic a iniciar juego se mostrará el FormRegister allí se pedirá el usuario para empezar a jugar al ingresarlo se guardará en la base de datos y pasara a mostrarse el userControl del juego para empezar a jugar; al dar clic a MostrarTop se cambiara al FormTop10 en el cual se mostrarán los 10 mayores puntajes.

En las clases DataGame y PlayerData se guardará información para el juego y para el jugador, en DataGame tendremos variables las cuales hacen la funcionabilidad del puntaje, sistema de vidas, entre otros y en PlayerData información de cada jugador, su usuario y puntaje.

La clase CustomPictureBox se utiliza para poder realizar cada golpe a un bloque y asi poder cambiar de sprites, entre otras cosas.

Por último la clase Player será utilizada cada vez que se necesite crear un nuevo jugador y a su vez un nuevo puntaje.

* **Diagrama relacional normalizado de la base de datos**

****

El diagrama presentado nos muestra dos tablas, la tabla player para relacionar al jugador mediante su id de usuario y u nombre de usuario, y la tabla score nos permite guardar mas especificamente la informacion del puntaje junto a su nombre de usuario teniendo como llave foranea el id del usuario.

Conceptos técnicos

**Implementación de la interfaz gráfica (vista)**

La interfaz gráfica de nuestro programa consiste en una ventana de formulario en la cual se irá cambiando de user control, esta ventana tiene tres botones de iniciar juego, mostrar el top 10 y salir, un label en el cual se muestra nuestro logo del juego, ademas de un tablelayoutpanel para mantener todo de una manera ordenada.

Tenemos los siguientes user controls y formularios utilizados en nuestro programa:

*Game.UserControl.cs*

*FormRegister.cs*

*FormTop10.cs*

*FormMenu.cs*

Cada uno tiene una funcionalidad, el Game.UserControl es utilizado para cargar todo el juego, el Register.UserControl es utlizado para guardar el nombre del usuario que desea jugar y el Top10.UserControl es para mostrar los primeros 10 jugadores que tienen el mayor puntaje.

**Manejo de clases en modelo vista controlador**

En nuestro programa utilizamos las siguientes clases para manejar de una mejor manera la funcionalidad del juego:

*CustomPictureBox.cs*

*Player.cs*

Asi mismo utlizamos clases de manera controlador para un mayor entendimiento del programa:

*DataGame.cs*

*PlayerData.cs*

*ConnectionDB.cs*

Plataforma Base

|  |  |
| --- | --- |
| Sistema Operativo | Multiplataforma |
| Tecnologías | Rider 2019.3.4 |
| Lenguaje | C# |
| Gestor de base de datos | PostgreSQL12 |
|  |  |

Nomenclaturas

Para un mejor entendimiento de nuestro programa y una mejor visualización del código se utilizaron diversas nomenclaturas las cuales se abrevian según su tipo , las utlizadas en el código son:

|  |  |
| --- | --- |
| TextBox | Txt |
| Button | Btn |
| Label | Lbl |

Eventos y Excepciones

En nuestro programa utlizamos eventos los cuales son necesarios para que el juego tenga total funcionalidad con el usuario, estos eventos son:

*FormLoadEventArgs.cs*

*FormMouseMoveEventArgs.cs*

*ButtonClickEventArgs.cs*

Las excepciones que utilizamos para evitar que nuestro programa genere problemas son:

*EmptyNicknameException.cs*

*ExceededMaxCharactersException.cs*

*NotRemainingLifesException.cs*

*OutOfBoundsException.cs*

*WrongKeyPressedException.cs*