MANUAL TÉCNICO ARKANOID

**Realizado por:**

* Axel Javier Alvarez Medina 00088119
* Joaquin Ernesto Santos Acosta 00159619
* Wilfredo Josue Morales Alfaro 00025719

**Contenido**

Manual Técnico……………………………………1

Aspectos generales……………………………….3

Objetivos del documento…………………………………….3

Descripción general…………………………………………..3

Software utilizado……………………………………………..3

Modelos utilizados…………………………………4

UML diagrama de clase………………………………………4

Diagrama relacional normalizado……………………………7

Modelo entidad-relación………………………………………7

Conceptos técnicos………………………………..8

Implementación de interfaz gráfica………………………….8

Manejo de clases………………………………………………9

Plataforma base………………………………………………..9

Nomenclaturas……………………………………...9

Eventos y excepciones…………………………...10

Eventos…………………………………………………………..10

Excepciones………………………………………………………12

**ASPECTOS GENERALES**

Objetivo del documento

Pretende explicar de manera clara la funcionalidad en cuanto al software que se maneja en el presente proyecto, explicando el proceso y herramientas utilizadas hasta llegar a la versión final

Descripción general

Para el manejo de software se hizo uso del modelo-vista-relación(MVR).la finalidad del programa es entretenimiento basado en un juego adaptado para el mejor entendimiento para el usuario

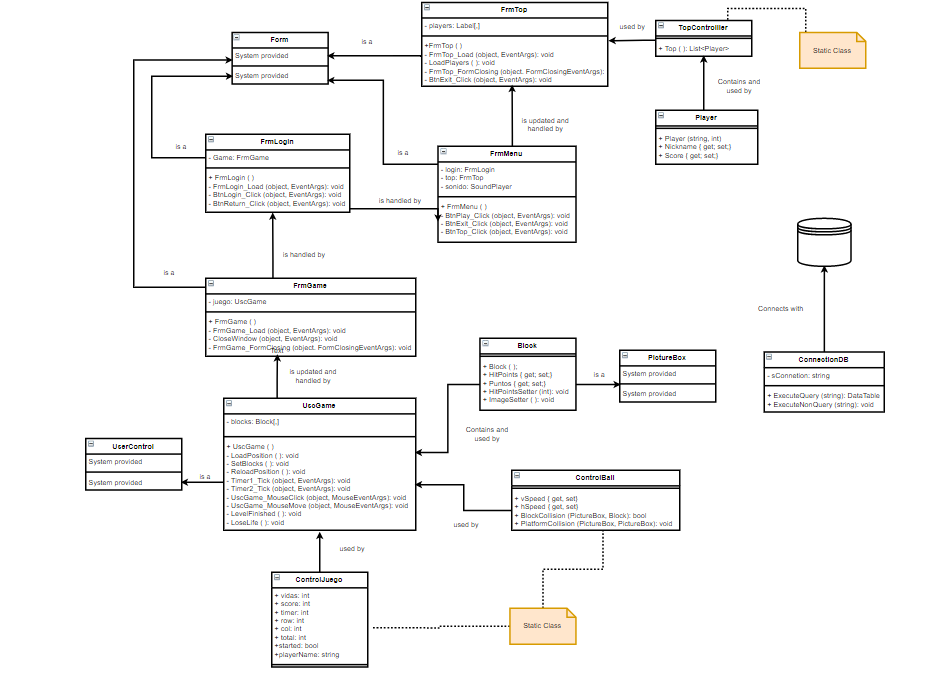
Software utilizado

Para la creación del programa se utilizó Visual Studio Community 2019 y JetBrains Rider 2019, postgreSQL para la creación de la base de datos; y Npgsql como complemento adicional para la conexión con la base de datos.

**MODELOS UTILIZADOS**

UML diagrama de clases

El diseño arquitectónico del software está basado en el siguiente diagrama de clases:



Para una vista más detallada puede visitar este link:

<https://drive.google.com/file/d/1buuRkaBABDPNzGxTwvJ-ozH4S7Hm5y-k/view?usp=sharing>

Explicando el diagrama de clases, se tiene un usuario por la ejecución de todo el programa.

FormMenu, hija de Form, es la primera clase que la usuaria vera (es decir la ventana principal); por medio de esta se tiene acceso al resto de ventanas, más en específico a FormLogin y FormTop (ambas hijas de Form). La funcionalidad de FormMenu no es otra más que la de redirigir al usuario a cualquiera de las otras dos Forms anteriormente mencionadas.

En FormLogin se obtiene el nickname del usuario (jugador) y se procede a guardar temporalmente en la clase estática ControlJuego (se mantendrá guardada hasta que se coloque otro nickname), por medio de la clase estática ConnectionDB se conecta a la base de datos y se verifica si el nickname ya existe dentro de la misma, en caso de no hacerlo se agregara, y se procede a abrir un FormGame, hija de Form, siempre y cuando se cumpla que el nickname no este vacío o supere el máximo de caracteres (en este caso 15).

FormGame es el encargado de cargar y actualizar a UserControlGame, hija de UserControl.

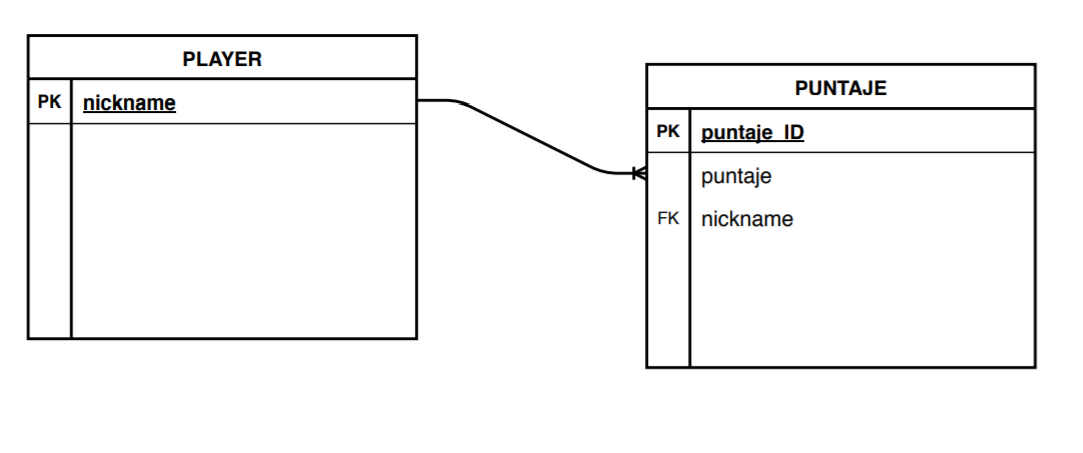
Dentro de UserControlGame se cargará el juego; principalmente los bloques, jugador y bola; al igual que una barra de estadísticas donde

el usuario podrá visualizar su puntaje, vidas restantes, tiempo restante (el tiempo solo será usado para bonificaciones) y su nickname. Sin embargo, la velocidad y colisiones con los bloques y el jugador de la pelota son controladas por la clase estática ControlBall (eliminando los bloques cuando sus hitPoints llegan a 0). Todo otro dato no visual es guardado en ControlJuego.

Para los bloques con los que interactúa el jugador se creó una clase especial llamada Block la cual hereda de PictureBox y lleva control de las veces que la bola ha interactuado con los bloques (hitPoints) al igual que el puntaje que el jugador obtendrá cuando destruya el bloque.

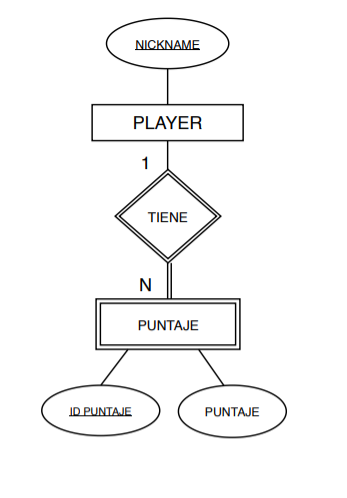
En FormTop se muestra un listado de los diez mejores puntajes, este listado es brindado por la clase estática TopController, el cual por medio de una de sus funciones crea un listado de jugadores haciendo uso de la clase modelo Player, a cada jugador en la lista se le guarda el puntaje y nickname de los diez usuarios con puntaje más alto en la base de datos. Los datos son obtenidos por medio de una consulta realizada por ConnectionDB.

DIAGRAMA RELACIONAL NORMALIZADO



Gracias a este esquema nos permite relacionar de buena manera los atributos utilizados para nuestro programa en nuestra base de datos, permitiéndonos guardar su puntaje por su id y su nickname.

MODELO ENTIDAD-RELACION



**CONCEPTOS TECNICOS**

IMPLEMENTACION DE INTERFAZ GRAFICA

La interfaz gráfica del programa consiste en 3 formularios siendo su principal el FrmMenu, este consiste en 3 botones, los primeros dos siendo su funcionalidad el acceder a los siguientes formularios para registrarte(FrmLogin) y el formulario para ver el top de puntajes con respecto a cada jugador(FrmTop), en el orden expuesto, y el ultimo botón con la única finalidad de salir y terminar el programa.

Más detalladamente, el FrmLogin contiene un textbox, en el cual el usuario digita su “nickname” y gracia al BttnLogin almacena cada uno en la base de datos; y contiene por ultimo un botón para regresar al menú(bttnReturn).

Así mismo el FrmTop contiene un listado de labels que es el encargado de mostrar los mejores 10 puntajes que se encuentren almacenados en la base de datos y un botón con la finalidad de salir de este form.

Luego, al pasar de la ventana “FrmLogin” accedemos a un formulario llamado “FrmGame” que contiene un único control de usuario(UserControl) encargado de manejar todos los aspectos de jugabilidad en nuestro proyecto; la movilidad de la barra, física de la pelota, destrucción de bloques, entre otras cosas.

MANEJO DE CLASES

Para la parte fundamental del modelo nos encontramos con las siguientes clases:

* ConectionDB.cs
* ControlBall.cs
* ControlJuego.cs
* TopController.cs
* Block.cs
* Player.cs

PLATAFORMA BASE

|  |  |
| --- | --- |
| Sistema Operativo | Multiplataforma |
| Tecnologías | Visual Studio Community 2019  JetBrains Rider 2019 |
| Lenguaje | C# |
| Gestor de DB | Postgre SQL |

**NOMENCLATURAS**

Para los elementos del entorno grafico se implementa la siguiente normativa de nombramiento:

<Abreviatura de tipo>\_descripción\_<id correlativo>

Las abreviaturas son las siguientes:

|  |  |
| --- | --- |
| PicturBox | Pic |
| Label | Lbl |
| Button | Btn |
| textBox | Txt |

**Eventos y excepciones**

**EVENTOS**

**FrmMenu**

* BtnPlay\_Click

Registra un click en el botón y es el encargado de pasar al formulario “FrmLogin.cs”

* BtnExit\_Click

Registra un click y es el encargado de finalizar el programa por completo

* BtnTop\_Click

Registra un click en el botón y es el encargado de pasar al formulario “FrmTop.cs”

**FrmLogin**

* FrmLogin\_Load

Es el encargado de cargar la posición de los objetos visuales en “FrmLogin”

* BtnLogin\_Click

Registra un clic y es el encargado de guardar el nickname del textbox en la base de datos y luego proceder al “FrmGame”

* BtnReturn\_Click

Registra un click y es el encargado de regresar al “FrmMenu” en caso de no querer continuar

**FrmTop**

* FrmTop\_Load

Es el encargado de cargar los labels con el puntaje de todos los registrados en la base

* FrmTop\_FormClosing

Es el encargado de borrar los labels ya puestos y colocar los nuevos, actualizando así los datos

* BtnExit\_Click

Registra un click y es el encargado de regresar al “FrmMenu”.

**FrmGame**

* FrmGame\_Login

Es el encargado de cargar “UscGame” para el juego.

* FrmGame\_FrmClosing

Es el encargado de remover el “UscGame” al momento de cerrar la ventana.

**UscGame**

* Timer1\_Tick

Es el encargado de manejar los aspectos en cuanto a jugabilidad se refiere, por ejemplo, las coordenadas que recorre la pelota, llama a las funciones encargadas de las colisiones, entre otras cosas.

* Timer2­\_Tick

Es el encargado de manejar el tiempo desde el inicio del juego hasta que lo finalice o en su defecto, pierda todas las vidas.

* UscGame\_MouseClick

Es el encargado de inicializar las mecánicas de jugabilidad para el usuario.

* UscGame\_MouseMove

Es el encargado de guardar los movimientos del mouse y reflejarlos en la barrita en la que colisiona la pelota.

**Excepciones**

Las excepciones, debido a sus nombres, son auto explicativas.

Constan de un constructor que recibe un string con el mensaje

de error. Se cuentan con las siguientes:

* EmptyNicknameException.cs
* ExcededMaxCharacterException.cs
* NoRemainingLivesException.cs
* OutOfBoundsException.cs
* WrongClickException.cs