

Manual técnico

Arkanoïd

Realizado por:

Oscar Alejandro Rodríguez Abrego, 00206019

Marco Vinicio Zelidon Molina, 00216219

Ivette Carolina Pinto León, 00186319

Contenidos

Aspectos generales	2
Objetivo del documento	2
Descripción general	2
Software utilizado	2
Modelos utilizados	3
UML Diagrama de clases	3
Diagrama entidad relación extendido	4
Diagrama relacional	5
Conceptos técnicos	6
Implementación de interfaz gráfica	6
Clases implementadas	6
Plataforma base	7
Nomenclaturas	8
Abreviaturas	8
Eventos y excepciones	9
Eventos	9
Excepciones	10

Aspectos generales

Objetivo del documento

El objetivo del documento presente es explicar el funcionamiento de un juego que ha sido recreado, así como también las herramientas utilizadas para su desarrollo de modo que los usuarios obtengan una buena experiencia en su uso.

Descripción general

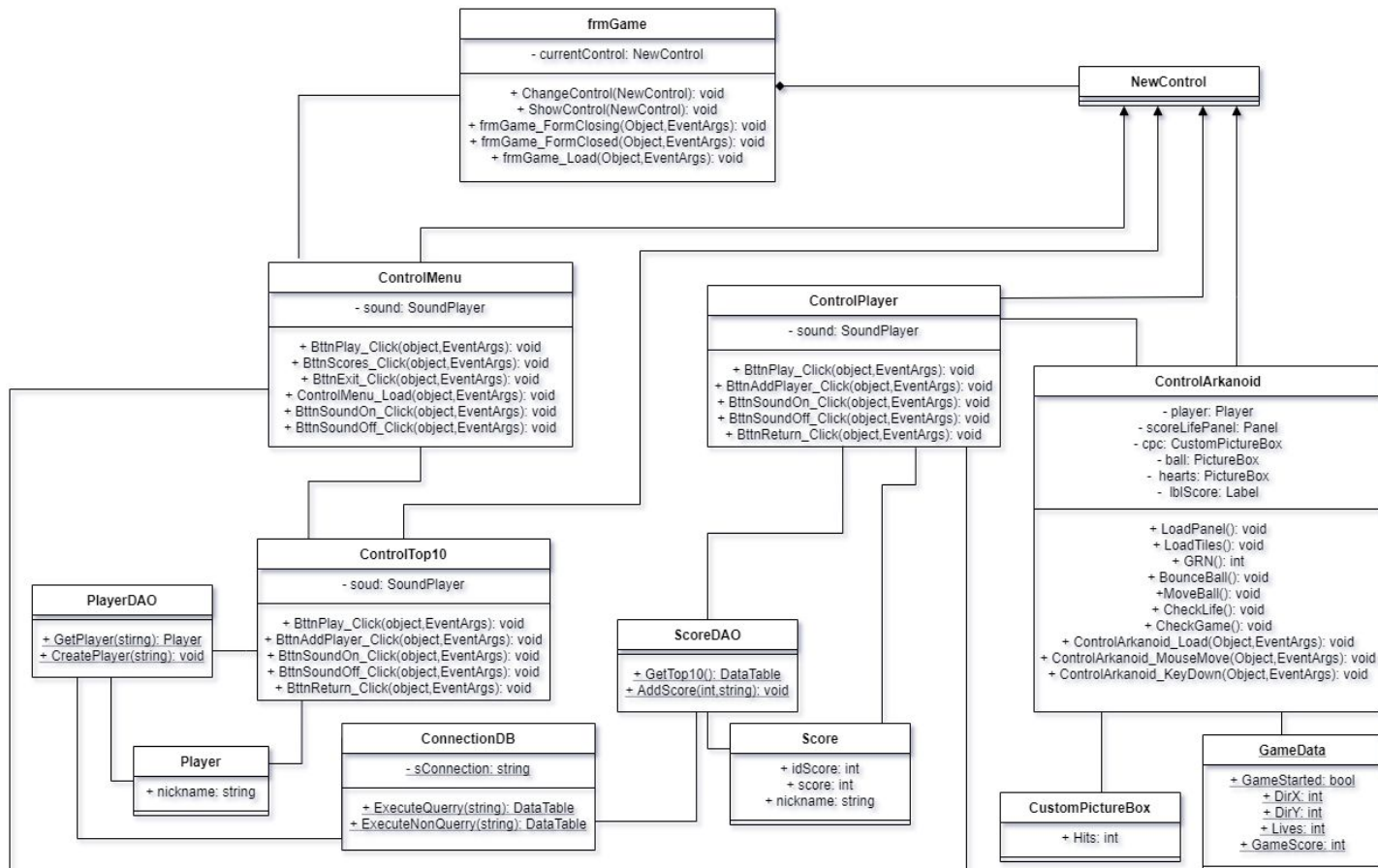
El programa está basado en un estilo de arquitectura de software que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes, dicho estilo es conocido como MVC (modelo - vista - controlador). El programa consiste en un juego, en la parte inferior de la pantalla se encuentra un estilo de nave con una pelota, el objetivo es que la pelota rebote en la nave para que destruya unos bloques encontrados en la parte superior de la pantalla.

Software utilizado

Para la creación del programa se utilizó JetBrains Rider en su versión 2020.1.2 (con el lenguaje de programación C# en su versión 7.3), en conjunto con PostgreSQL 12 para la creación de la base de datos. En cuanto a los complementos utilizados en JetBrains Rider, se encuentra Npgsql (versión 4.1.3.1) el cual nos permite realizar la conexión entre la base de datos y nuestro programa.

Modelos utilizados

UML Diagrama de clases



Para una mejor vista, puede encontrar esta imagen haciendo click en el siguiente link:

[UML Diagrama de clases](#)

Detallando el diagrama de clases, si un jugador gana o pierde, puede volver a jugar ya sea él mismo u otro usuario.:

Player, cuenta con el nickname único del usuario para iniciar el juego.

Score, cuenta con un id del Score, el score del usuario y el nickname del usuario.

PlayerDAO, contiene dos métodos de consulta en la base de datos, uno que devuelve un usuario que se usa para verificar si el usuario ya existe en la base de datos en el momento de querer iniciar el juego, y otro para crear un nuevo usuario en caso de que no exista en la base de datos.

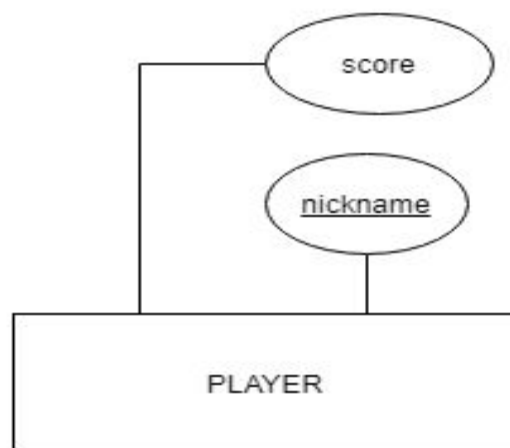
ScoreDAO, contiene dos métodos de consulta en la base de datos, uno que devuelve el top 10 de los mejores puntajes con su respectivo jugador, y otro, que agrega un score a un usuario en caso de que gane el juego.

La interfaz gráfica consta de:

ControlMenu: es el menú principal del programa el cual hereda de NewControl, el cual contiene los botones de jugar, top 10 y salir. Al hacer click sobre el botón jugar se muestra el ControlPlayer el cual también hereda de NewControl, y solicita un nickname, posteriormente verifica si el nickname ingresado existe en la base de datos, en caso que no exista, el usuario debe registrar dicho nickname previamente a iniciar el juego. El juego se inicia de acuerdo con el nickname que ha sido ingresado, mediante un objeto de tipo Player, si el usuario gana se le asigna un puntaje equivalente a 180, este puntaje se multiplica por el número de vidas que obtuvo al final de la partida para obtener el puntaje final.

Diagrama entidad relación extendido

La base de datos del programa fue creada con la ayuda de PostgreSQL y se basa en el siguiente diagrama:

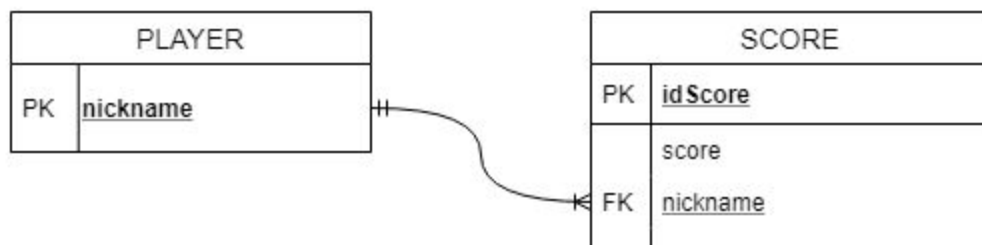


También puede encontrar esta imagen haciendo click en el siguiente link:

[Diagrama ER](#)

Diagrama relacional

A partir del diagrama entidad relación mostrado previamente, se obtiene un diagrama relacional de dicha base de datos, el cual se presenta a continuación:



También puede encontrar esta imagen haciendo click en el siguiente link:

[Diagrama relacional](#)

Conceptos técnicos

Implementación de interfaz gráfica

Para el programa la interfaz gráfica es almacenada en la carpeta “*Vista*”. y se compone de un formulario y cuatro controles de usuario, estos últimos poseen cada uno la interfaz para cada parte del programa y conviven en el formulario alternando su aparición y lanzamiento conforme el usuario interactúa con cada una de ellas. El formulario y los controles de usuario son:

- frmGame.cs
- ControlArkanoid.cs
- ControlMenu.cs
- ControlPlayer.cs
- ControlTop10.cs

ControlArkanoid.cs es el control de usuario encargado de llevar a cabo dentro de sí el juego Arkanoid.

ControlMenu.cs es el primer control de usuario desplegado en frmGame.cs y posee las opciones y los medios para lanzar los todos los demás controles de usuario.


ControlPlayer.cs tiene la tarea de llevar a cabo los procesos necesarios para que el usuario, pueda jugar.

ControlTop10.cs posee la funcionalidad que permite al usuario ver a los 10 jugadores con los puntajes más altos.

Clases implementadas

En el apartado no gráfico del juego encontraremos todo lo que permite que el programa se ejecute correctamente, se dividen y están almacenadas en dos carpetas “*Modelo*” y “*Controlador*” las clases de este apartado interactúan con el usuario mediante la interfaz gráfica y son las siguientes:

- Controlador:



ConexionDB.cs es una clase que contiene los métodos necesarios para comunicar a aquellas clases que lo requieran, con una base de datos.

PlayerDAO.cs es el Objeto de Acceso de Datos (DAO por sus cifras en inglés) de la clase Player, y sirve como una interfaz abstracta entre el programa y una base de datos.

ScoreDAO.cs es el Objeto de Acceso de Datos (DAO por sus cifras en inglés) de la clase Score, y sirve como una interfaz abstracta entre el programa y una base de datos.

- Modelo:

CustomPictureBox.cs es una clase hija de la clase PictureBox y añade a esta la propiedad “Hits”, necesaria para el juego, esta clase es utilizada para construir los bloques del juego.

GameData.cs es una clase estática que almacena información relevante para el juego.

NewControl.cs es una clase hija de UserControl y se encarga de implementar una característica necesaria para la interfaz gráfica del programa.

Player.cs es una clase vital para el programa ya que es la clase que representa y maneja todo lo relacionado al jugador.

Score.cs es la clase que encargada de manejar los puntajes de los jugadores en tiempo de ejecución.

Plataforma base

Sistema operativo	Windows
Tecnologías	.NET Framework 4.8
Lenguaje	C# 7.3
IDE	JetBrains Rider 2020.1.2
Gestión de bases de datos	PostgreSQL 12

Nomenclaturas

Abreviaturas

Las clases pertenecientes a la carpeta “Vista” contienen elementos de interfaz gráfica que siguen las siguientes normas de abreviatura:

Button	bbtn
Table Layout Panel	tlp
Group Box	gpb
Label	lbl
Text Box	txt
Picture Box	pcb
Custom Picture Box	cpb

Eventos y excepciones

Eventos

- frmGame.cs

frmGame_FormClosing: Un mensaje de verificación para confirmar que el usuario desea cerrar el programa.

frmGame_FormClosed: El último evento realizado, se encarga de cerrar y terminar todos los procesos abiertos por la aplicación.

frmGame_Load: Prepara los elementos gráficos para el programa como el primer control de usuario.

- ControlArkanoid.cs

ControlArkanoid_Load: Inicializa y establece la posición inicial de los elementos y prepara al control de usuario.

ControlArkanoid_MouseMove: Registra los movimientos del mouse permitiendo la interacción del usuario con el juego.

ControlArkanoid_KeyDown: Registra las pulsaciones del teclado, en especial la de la tecla “espacio”, la cual da inicio al juego.

- ControlMenu.cs

BttnPlay_Click: Registra la pulsación del botón que despliega el control responsable de lanzar el juego.

BttnScores_Click: Registra la pulsación del botón que lanza el control que muestra el top de jugadores.

BttnExit_Click: Registra la pulsación del botón que termina y cierra el programa.

ControlMenu_Load: Se encarga de desplegar el estado inicial del control del menú.

BttnSoundOn_Click: Registra la pulsación del botón que reproduce la música del juego.

BttnSoundOff_Click: Registra la pulsación del botón que detiene la reproducción de música del juego.

- ControlPlayer.cs

BttnPlay_Click: Registra las interacciones, entre el usuario y el programa, necesarias para que el primero acceda al juego como tal.

BttnAddPlayer_Click: Registra la pulsación del botón que agrega a un usuario al grupo de jugadores.

BttnSoundOn_Click: Registra la pulsación del botón que reproduce la música del juego.

BttnSoundOff_Click: Registra la pulsación del botón que detiene la reproducción de música del juego.

BttnReturn_Click: Registra la pulsación del botón encargado de retroceder al menú.

- ControlTop10.cs

ControlTop10_Load: Su función es preparar el estado inicial del control de usuario.

BttnSoundOn_Click: Registra la pulsación del botón que reproduce la música del juego.

BttnSoundOff_Click: Registra la pulsación del botón que detiene la reproducción de música del juego.

BttnReturn_Click: Registra la pulsación del botón encargado de retroceder al menú.


Excepciones

El programa cuenta con cinco excepciones personalizadas, todas ellas son necesarias para proteger al programa de acciones inesperadas por parte del usuario, y son:

EmptyNameException.cs es la excepción lanzada cuando el usuario intenta agregar un jugador o comenzar a jugar, sin especificar el nombre de un jugador.

ExceededMaxCharactersException.cs es la excepción lanzada cuando el usuario intenta registrar a un jugador cuyo nombre excede el número máximo de caracteres permitidos.

InvalidNicknameFormatException.cs es la excepción lanzada cuando el usuario intenta registrar a un jugador cuyo nombre posee menos de tres caracteres.



UserNotFoundException.cs es la excepción lanzada cuando el usuario presiona el botón de jugar, y el nombre del jugador introducido no fue encontrado en la base de datos.

WrongKeyPressedException.cs es la excepción lanzada cuando el usuario, dentro del juego intenta iniciar a jugar presionando una tecla diferente de “espacio”.