Manual técnico Arkanoid

Realizado por:

René Armando Flores Cortez 00118719

Dylan Moisés Meléndez Linares 00087018

René Guillermo Canales Clímaco 00001318

Alberto Enrique de León Quiñonez 00087919

GrupoLarry_Arkanoid

Contenido

Aspectos generales

Objetivo del documento

El objetivo de este documento pretende orientar y explicar el diseño del software creado, explicando las herramientas utilizadas y a utilizar para futuras versiones.

Descripción general

Para la creación del software se hizo uso del modelo – vista – controlador. El programa está inspirado en la versión original de Arkanoid del desarrollador Taito en 1986, presentando asi una remasterización de este juego, en el cual el usuario puede disfrutar de un nivel prediseñado.

Software utilizado

Para la creación del programa se utilizó JetBrains Rider 2019, en conjunto con PostgreSQL 11 en el cual se creó la base de datos. Complemento adicionale para JetBrains Rider utilizado ha sido Npgsql.

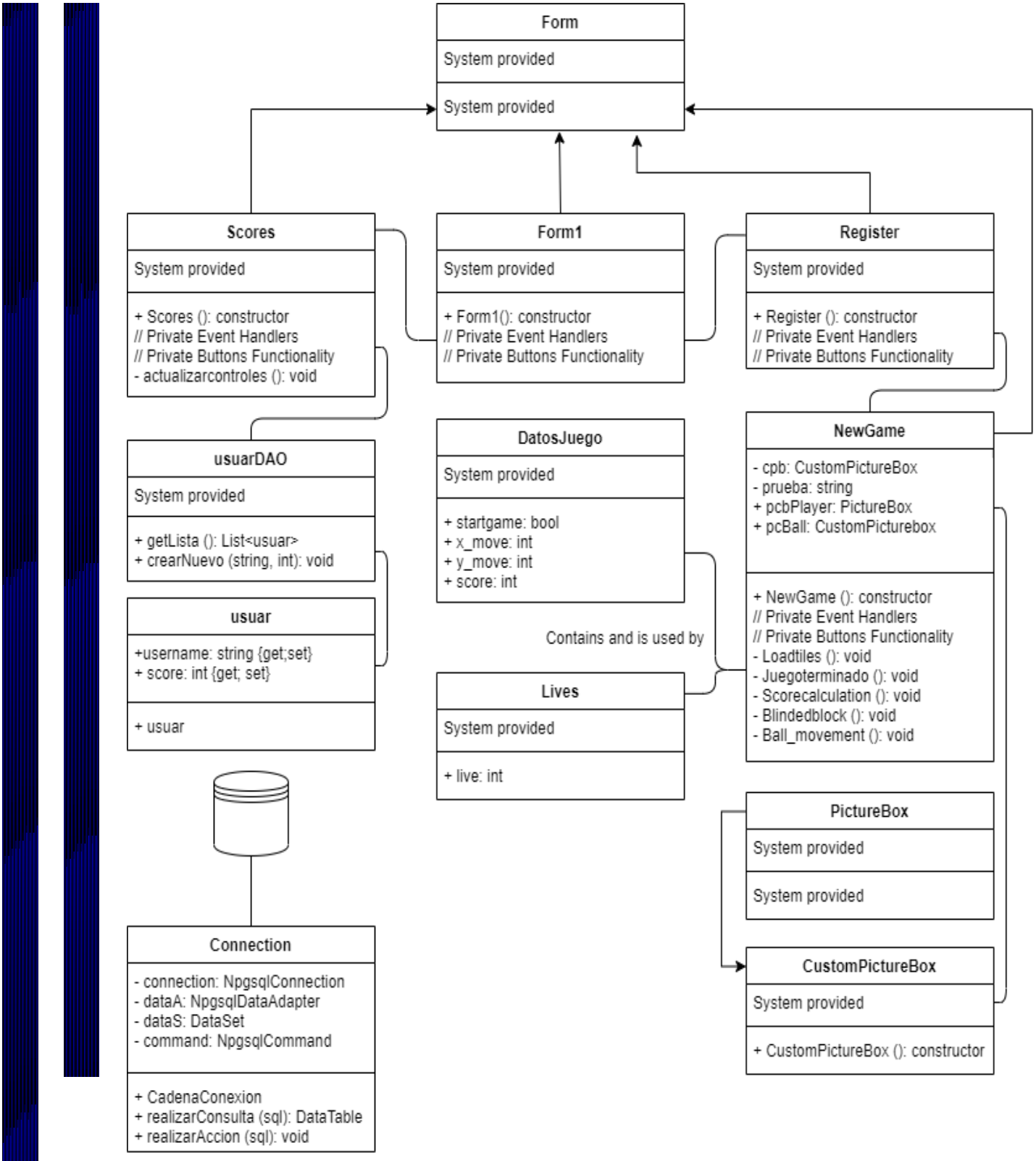
Modelos utilizados

UML Diagrama de clases

El diseño arquitectónico del código está basado en el diagrama de clases siguientes (ver en siguiente página):

Hipervínculo para una vista más detallada:

https://drive.google.com/file/d/15r6kEQjb3KBBZkO5oWlXQOuMzB1Ug9FN/view?usp=sharing


Form
System provided
System provided
Scores
System provided
+ Scores (): constructor
// Private Event Handlers
// Private Buttons Functionality
- actualizarcontroles (): void
Form1
System provided
+ Form1(): constructor
// Private Event Handlers
// Private Buttons Functionality
Register
System provided
+ Register (): constructor
// Private Event Handlers
// Private Buttons Functionality
usuarDAO
System provided
+ getLista (): List<usuar>
+ crearNuevo (string, int): void
usuar
+username: string {get;set}
+ score: int {get; set}
+ usuar
DatosJuego
System provided
+ startgame: bool
+ x_move: int
+ y_move: int
+ score: int
Contains and is used by
Lives
System provided
+ live: int
NewGame
- cpb: CustomPictureBox
- prueba: string
+ pcbPlayer: PictureBox
+ pcBall: CustomPicturebox
+ NewGame (): constructor
// Private Event Handlers
// Private Buttons Functionality
- Loadtiles (): void
- Juegoterminado (): void
- Scorecalculation (): void
- Blindedblock (): void
- Ball_movement (): void
PictureBox
System provided
System provided
CustomPictureBox
System provided
+ CustomPictureBox (): constructor
Connection
- connection: NpgsqlConnection
- dataA: NpgsqlDataAdapter
- dataS: DataSet
- command: NpgsqlCommand
+ CadenaConexion
+ realizarConsulta (sql): DataTable
+ realizarAccion (sql): void

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

Form1 cuenta con 3 botones (Nuevo juego, Puntuaciones, Salir los cuales haran que el usuario cambie de ventana, sus metodos son solo las acciones de presionar los botones, esta es la primer ventana que se le muestra al usuario.

Register esta ventana se abre al pulsar un boton, esta cuenta con una verificación de existencia de usuario y cuenta con un boton, este es el unico camino para abrir la siguiente ventana.

NewGame esta es la ventana en la que se desarrolla el juego, cuenta con constructores para armar el juego y colocar los bloques, cuenta con comprobadores que verifican cuantas **lives** le quedan, cuenta con un contador de puntos llamado **ScoreCalculation,** al finalizar el juego esta manda la puntuación registrada al respectivo **usuar**.

Scores es una ventana en la cual se pueden visualizar las puntuaciones de cada usuario registrado en orden de mayor puntuación a menor puntuación, cuenta con una función *actualizarcontroles* que actualiza la tabla.

usuarDAO es una clase que nos ayuda a almacenar a los **usuar** en una lista.

Usuar es una clase que nos ayuda a obtener los datos de los usuarios, este cuanta con *username* y *score*, cada uno con su debido getter y setter.

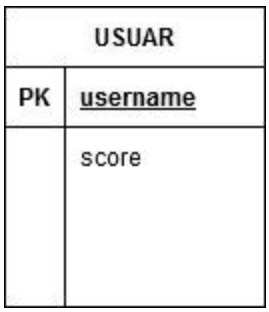
DatosJuego es una clase que cuenta con diferentes variables, las cuales nos ayudan en la ventana de **NewGame** para darle movilidad a la barra del jugador y a la pelota.

Lives es una clase que nos ayuda a contabilizar la cantidad de vidas del jugador.

Connection es una clase que se conecta a la base de datos almacenada en SQL.

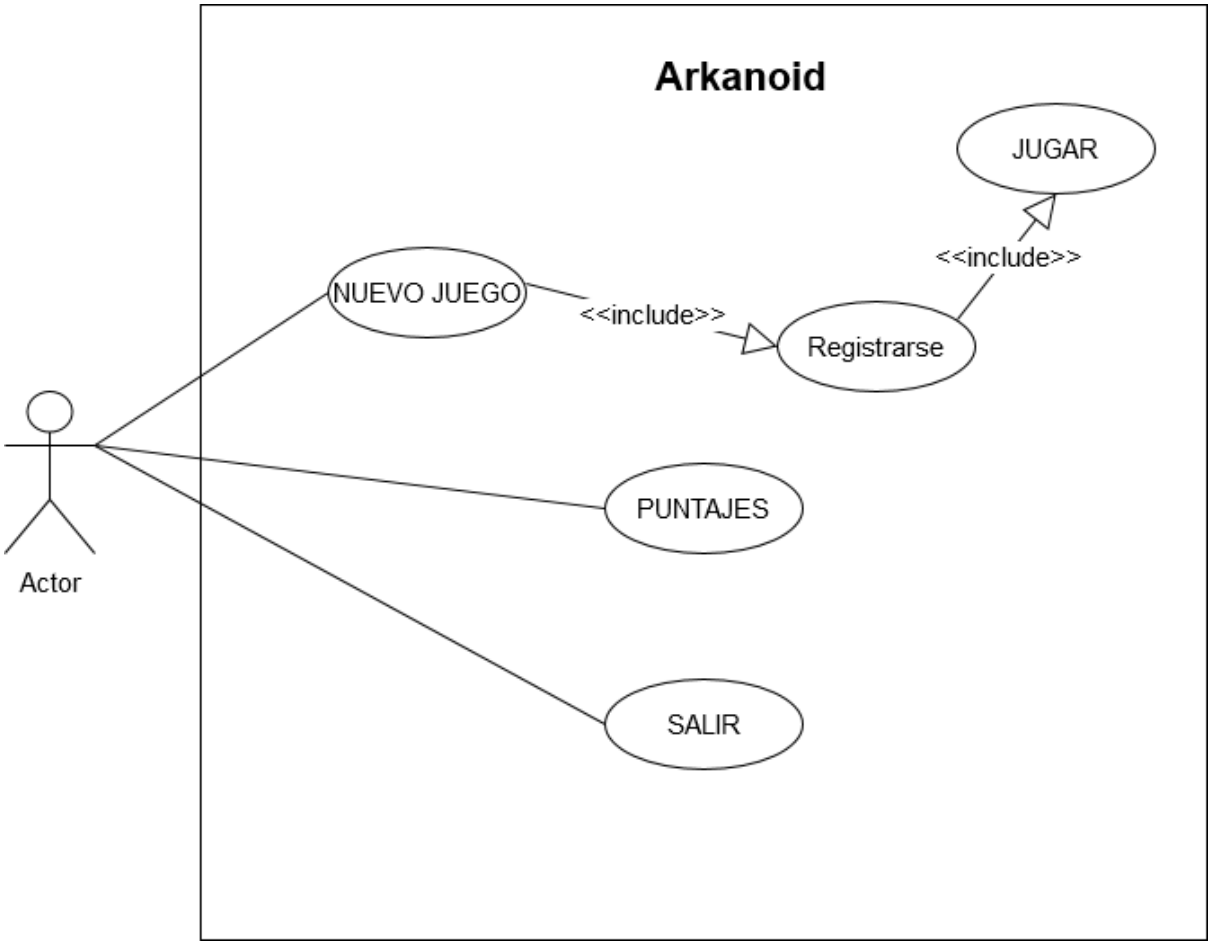
Diagrama relacional normalizado de base de datos utilizada

Complementariamente, el esquema brindado de la base de datos se segmenta de la siguiente manera:



Este es un esquema bastante básico, debido a que en el juego solo se registra el nombre del usuario (username) y se registra la puntuación que obtuvo en la partida (score)

UML de diagrama de casos de usos



Conceptos técnicos

Implementación de interfaz gráfica

La interfaz gráfica del programa consiste de 4 ventanas o formularios, la primera de ellas es llamada **Form1.cs,** la cual está compuesta de 3 botones que son **btnPuntajes, btnClose, btnNewGame,** los cuales redirigen respectivamente a las otras 2 ventanas.

El **btnNewGame** abre la ventana **Register.cs** la cual cuenta con un **textBox** y un **button** el cual redirige a la ventana **NewGame.cs** en el cual se desarrolla el juego, ventana la cual cuenta con **hearts, timer y labels.**

El **btnPuntajes** lleva a la ventana **Scores.cs,** ventana la cual cuenta con un **tableLayoutPanel,** el **btnClose** cierra completamente el programa.

Manejo de clases en modelo

Para manejar la parte fundamental del modelo del programa, se cuenta con las siguientes clases:

- Connection.cs
- DatosJuego.cs
- Lives.cs
- Usuar.cs
- usuarDAO.cs

Plataforma base

SISTEMA OPERATIVO	MULTIPLATFORMA
TECNOLOGIAS	JetBrains Rider 2019
LENGUAJE	C#
GESTOR DE DB	PostgreSQL

Nomenclaturas

Abreviaturas

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

<Abreviatura de tipo>_descripción_<id correlativo>

Las abreviaturas se presentan a continuación:

LABEL	lbl
TEXTBOX	txt
PICTURE	pic
BUTTON	btn
DATA GRID VIEW	dgv
TIMER	tmr

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:

- EmptyTextBoxException.cs
- HasNotPlayesGameException.cs
- NoRemainingLivesException.cs
- OtherKeyException.cs
- TooManyCharactersException.cs