Manual Técnico

Realizado por:
Carlos Roberto Cortez Iraheta, 00204119
Diana Sugeyth Umaña Rodriguez, 00143619
Henry Alexander Cortez Amaya, 00095119
Oscar Enrique Orellana Monterrosa, 00258219

Jajant games

Contenido

Aspectos Generales

Objetivo:

- Proporcionar la información necesaria para guiar al lector del desarrollo de la interfaz y el diseño de software implementado, así también conocer sus partes y la manera en la que fue construida, etc.

Descripción general:

- El programa que se presenta se define como un juego ya conocido como **Arkanoid**, el cual se controla una pequeña plataforma apodada como *nave espacial Vaus* , la cual tiene como objetivo impedir que la bola salga de la zona del juego, en la parte superior se cuentan con bloques o ladrillos de diferente dureza, los cuales serán destruidos por la bola. Para un mejor desarrollo y entendimiento de la aplicación, se aplicó el Modelo Vista Controlador (MVC).

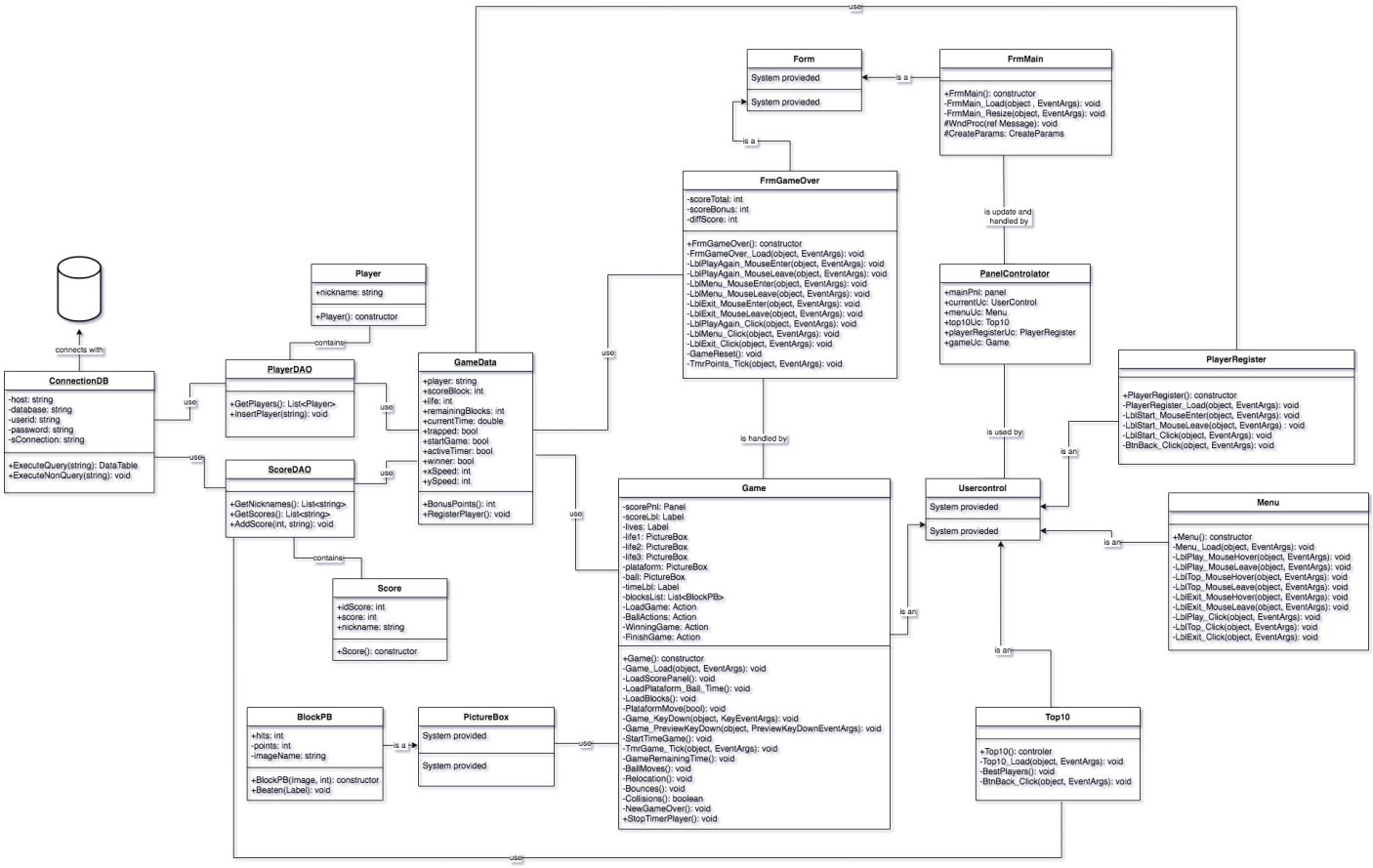
Software utilizado:

- Para la creación del programa se utiliza el lenguaje JetBrains Rider en su versión 2019.3.4, en conjunto con PostgreSQL 12 para la creación de la base de datos. Para la conexión de la base de datos se complemento con Npgsql.

Modelos Utilizados

Diagrama de clases:

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



- Puede apreciarse en una mejor calidad en el siguiente enlace

- Se tiene un form “FrmMain” que es la ventana principal para el manejo del programa, este a su vez hereda de la clase “Form”.
- Mediante una clase estática “panelControlator” se maneja el intercambio entre cada control de usuario en un panel “pnlBase” de los cuales participan “Menu.cs”, “Top10.cs”, “PlayerRegister.cs” y “Game.cs”, que son heredados de la clase “UserControl”.
- “Menu.cs” se encarga mediante eventos, de usar el “panelControlator” para cambiar el control de usuario actual en el “pnlBase”.
- Al entrar a “Top10.cs” cargan métodos que llenan los Labels que contienen los nombres de los diez mejores jugadores y sus correspondientes puntajes que son extraídos mediante las consultas en “PlayerDAO” y “ScoreDAO”.
- Cuando cambia a “PlayerRegister.cs” se puede agregar el nickname del jugador en un caja de texto que pasa a guardarse a una clase estática llamada “GameData”, esta clase contiene los datos más relevantes para el desarrollo del juego.
- En “Game.cs” se guardan los atributos que en su mayoría son los componentes de este control de usuario, como lo son los bloques que se modelan en una clase llamada “BlockPB” la cual hereda de un “PictureBox” y guarda las propiedades necesarias para el funcionamiento de un bloque del juego, por medio de eventos y métodos se logra la funcionalidad del juego (movimientos, físicas, control de puntaje, sistemas de vida, y manejo del tiempo).

- Cuando el juego llega a concluir (ganar o perder) se muestra una ventana llamada “FrmGameOver” que hereda de ”Form” que contiene un resumen de la partida jugada (score, nombre del jugador) que mediante eventos muestra estos datos y que da tres opciones de las cuales podemos reiniciar la partida, movernos al control de usuario “Menu.cs” o salir de la aplicación.
- Si cuando concluye la partida y el jugador gana se guardan los datos en la base de datos (nickname y/o score), para ello se utilizan las siguientes clases: “ScoreDAO” y “PlayerDAO” que se encargan de enviar o recibir datos, para ello se apoya de “Player” y “Score” donde se crean objetos con los datos correspondientes, esta recepción o emisión de datos son enviadas a “ConnectionDB” que es la clase encargada de comunicarse con la base de datos y enviarle lo que necesitamos hacer y/u obtener.

Diagrama Entidad Relación Extendido:

- El juego necesitaba guardar los nicknames de los jugadores y sus puntajes alcanzados, de esta manera se realizó el esquema correspondiente para la creación de la base de datos del juego y se representa de la siguiente manera:

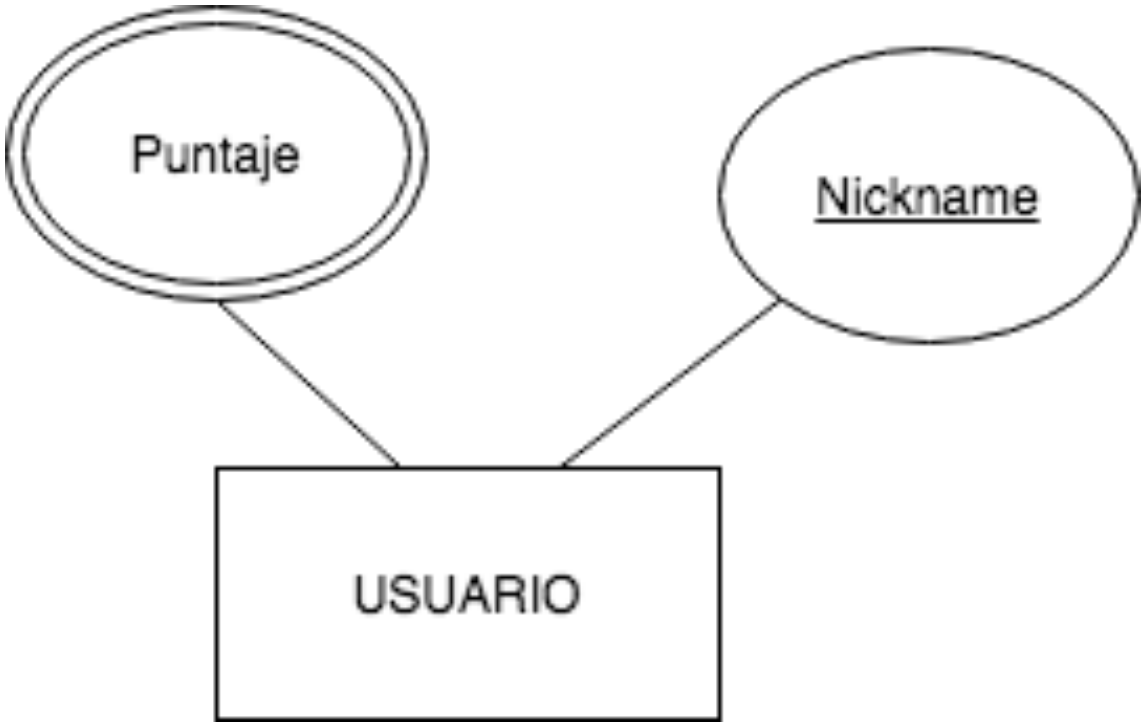
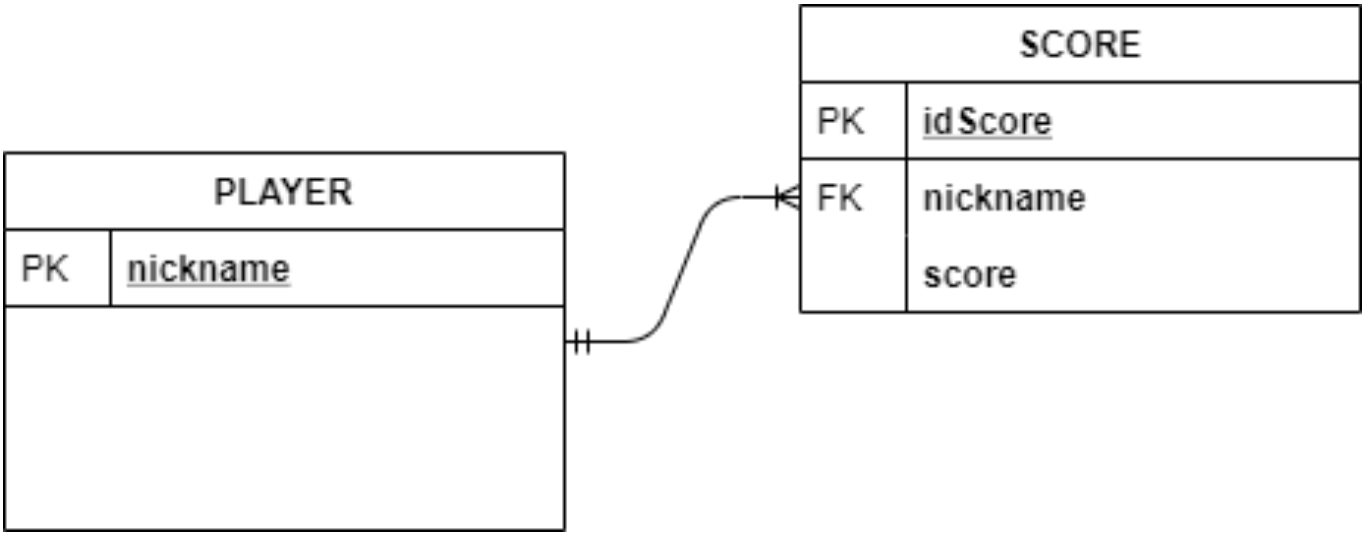


Diagrama Relacional:

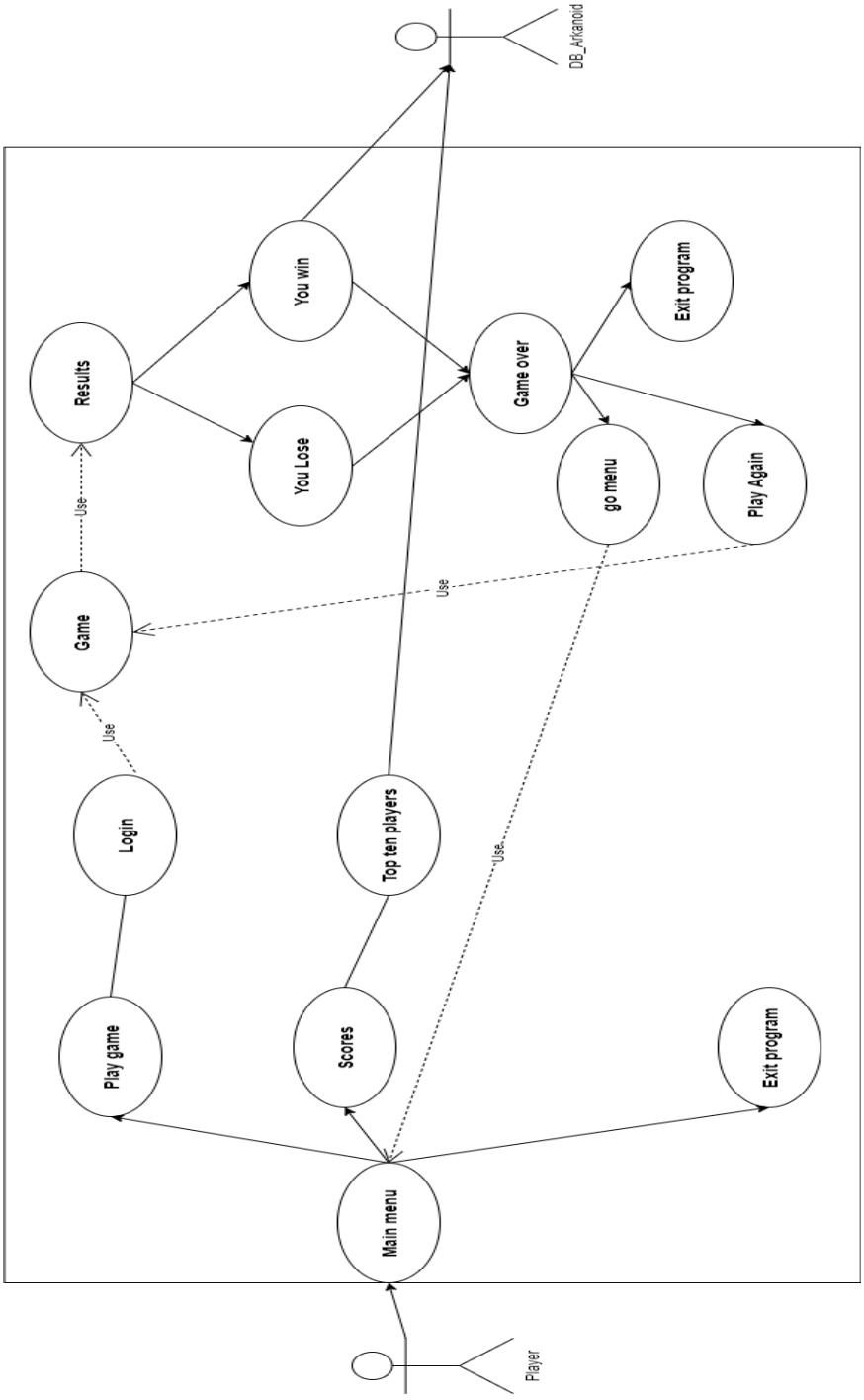
- Se modificó el diagrama Entidad Relación para obtener el diagrama Relacional y puede representarse de la siguiente manera:



- El objetivo de crear la base de datos era precisamente guardar los nickname de los jugadores y su puntaje al ganar una partida. El jugador tiene la opción de volver a jugar, por lo que es posible acumular un nuevo puntaje. Esto es lo que se puede interpretar en el esquema anterior.

Diagrama Casos de Uso:

- El siguiente diagrama representa todo aquello con lo que el usario puede interactuar dentro del programa:



Concéptos Técnicos

Implementación de la Interfaz Gráfica:

La interfaz gráfica del programa consiste en una ventana global o formulario “FrmMain.cs”, compuesta únicamente con un panel principal llamado “pnlBase” que es el encargado de cargar distintos controles de usuario, estos son:

- ***Menu.cs***: contiene tres labels que simulan botones, los cuales nos ayudan a cambiar entre user controls.
- ***Top10.cs***: contiene labels donde se muestra la lista de los nicknames y scores de los mejores 10 jugadores.
- ***PlayerRegister.cs***: contiene un label y textbox que nos ayuda a registrar al jugador para luego pasar al “Game.cs”.
- ***Game.cs***: Dentro de él se encuentran los componentes del juego, 24 picturebox “Bloques” que se muestran mediante una lógica de matriz, los picturebox de la plataforma y la bola, además de un panel que contiene el score del jugador (label) y las vidas restantes (picturebox).

Se implementa también una ventana “FrmGameOver.cs” que es la encargada de manejar el final del juego, contiene labels los cuales muestran el resumen de los scores obtenidos, además de labels, que simulan botones, con las opciones para volver a jugar, volver al “Menu.cs” o salir del juego.

Manejo de Clases en Controlador:

Para controlar el flujo de datos en el programa y a la base de datos se manejan las siguientes clases:

- GameData.cs
- PanelControlator.cs
- PlayerDAO.cs
- ScoreDAO.cs

Manejo de Clases en Modelo:

Para manejar el modelo del programa se utilizan las siguientes clases:

- BlockPB.cs
- Player.cs
- Score.cs

Plataforma Base:

Sistema Operativo	Windows 10
Tecnologías	JetBrains Rider
Lenguaje	C#
Gestor de DB	PostgreSQL

Tipos de Error:

Controlados:

Los diferentes tipos de errores que se consideran y controlan en nuestro programa son:

- Dejar vacío el nickname o solo con espacios (PlayerRegister.cs)
- Presionar ESC cuando el juego este pausado (Game.cs)
- Minimizar la ventana con el juego ejecutándose (Game.cs)
- Error al obtener datos de la base de datos (PlayerDAO.cs, ScoreDAO.cs)
- Error al insertar datos a la base de datos (PlayerDAO.cs, ScoreDAO.cs)
- Presionar una tecla diferente de SPACE para iniciar el juego (Game.cs)
- La plataforma se sale de los límites de Right y Left (Game.cs)

No Controlados:

Este error salta muy pocas veces cuando se le da un uso no promedio:

- Bitmap

Nomenclaturas

Abreviaturas:

Para los componentes del entorno gráfico se implementa el siguiente formato de nombramiento con el estilo *camelCase* (<Abreviatura>Descripción).

Las abreviaturas se presentan a continuación:

Form	frm
UserControl	uc
Timer	tmr
Panel	pnl
TableLayoutPanel	tlp
PictureBox	pb
Label	lbl
Button	btn
TextBox	Txt

Eventos y Excepciones

Eventos:

Para poder realizar una acción en un control de usuario o formulario fue necesario la implementación de eventos, tales como:

Eventos más generales:

Load Event: cargan todos los recursos del controlador.

- FrmMain_Load
- Menu_Load
- Top10_Load
- PlayerRegister_Load
- Game_Load
- FrmGameOver_Load

MouseEnter, MouseLeave, MouseHover**:** eventos que ayudan a simular el funcionamiento de un hover en un Label, además cambian el diseño al pasar con el mouse sobre él y al salir el mouse vuelve a su estado predeterminado.

- LblPlay_MouseHover
- LblPlay_MouseLeave

- LblTop_MouseHover
- LblTop_MouseLeave
- LblExit_MouseHover
- LblExit_MouseLeave
- LblStart_MouseEnter
- LblStart_MouseLeave
- LblPlayAgain_MouseEnter
- LblPlayAgain_MouseLeave
- LblMenu_MouseEnter
- LblMenu_MouseLeave
- LblExit_MouseEnter
- LblExit_MouseLeave

Eventos más específicos:

- ***FrmMain_Resize***: detecta al minimizar la ventana y pausa el juego (mientras se esté jugando), así también maximiza la ventana siempre que esta esté visible.
- ***LblPlay_Click***: cambia a el UserControl PlayerRegister para poder jugar.
- ***LblTop_Click***: cambia a el UserControl Top10 para poder ver los mejores puntajes.
- ***LblExit_Click***: cierra la aplicación(desde Menu).

- ***BtnBack_Click***: cambia a el UserControl Menu para elegir otra opción(desde Top10).
- ***LblStart_Click***: cambia a el UserControl Game y guarda el nombre del jugador.
- ***BtnBack_Click***: cambia a el UserControl Menu para elegir otra opción(desde PlayerRegister).
- ***Game_KeyDown***: realiza acciones en el juego si se apreta una tecla.
- ***Game_PreviewKeyDown***: ayuda a detectar las arrows keys.
- ***TmrGame_Tick***: cada cierto intervalo de milisegundos pasan ciertas acciones en el juego.
- ***LblPlayAgain_Click***: reinicia cada elemento en el juego para volver a jugar al instante.
- ***LblMenu_Click***: cambia a el UserControl Menu cuando se termina de jugar.
- ***LblExit_Click***: cierra la aplicación(desde FrmGameOver).
- ***TmrPoints_Tick***: mediante un intervalo de milisegundos se suma el puntaje bonus al total.

Excepciones:

Las excepciones fueron creadas para evitar posibles errores en ejecución del programa. Son clases que reciben en su constructor un parámetro string con el que se muestra el mensaje de error.
Las excepciones poseen nombres autoexplicativos que explican su razón de uso, estas son los siguientes:

- AddScoreException
- EmptyNickNameException
- GamePausedException
- GetDataException
- MinimizingPauseException
- OutOfBoundsException
- PlayerRegistrationException
- StartOfTheGameException