



# TATETI

Documentación

---



UTN FRD - Universidad Tecnológica  
Nacional  
Tecnicatura en Programación  
Laboratorio de Computación IV  
Docente: Castaño Facundo

# DESCRIPCIÓN Y DESARROLLO

Este proyecto consiste en la implementación del clásico juego de tatetí interactivo utilizando la biblioteca tkinter de Python para la interfaz de usuario y mysql.connector para el almacenamiento de datos en una base de datos MySQL. El juego permitirá a dos jugadores competir entre sí, llevando un registro de sus nombres y puntuaciones.



## Requisitos:

⊗ Lenguaje:

Python (Última versión)

⊗ Librerías:

tkinter (ya incluida en la instalación estándar de Python)

pygame para los efectos de sonido (puede instalarse usando pip)

mysql-connector-python (puede instalarse usando pip)

⊗ Database:

MySQL Server (debe estar instalado y en ejecución para la base de datos)

⊗ Archivos:

Archivos de sonido en formato MP3 para los efectos de sonido (opcional).

Nombres de los archivos:

win\_sound.mp3

tie\_sound.mp3

### Procedimiento de Instalación:

- . Descargar Python si aún no se encuentra instalado. (Sitio oficial: [python.org](https://python.org))
- . Instalar las librerías necesarias usando pip:
  - | **pip install pygame mysql-connector-python**
  - | **pip install pygame**
- . Configurar la base de datos MySQL en el servidor local o remoto.
- . Ejecutar el script de creación de la tabla en nuestra base de datos MySQL desde la línea de comandos.

Base de datos -> proyecto

Tabla -> partidas

Columnas -> id, jugador1, jugador2, resultado, puntos\_ganador

| ejemplo:

```
CREATE TABLE partidas (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    jugador1 VARCHAR(255) NOT NULL,  
    jugador2 VARCHAR(255) NOT NULL,  
    resultado VARCHAR(255) NOT NULL,  
    puntos_ganador INT NOT NULL  
);
```

```
mysql> describe partidas;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
jugador1	varchar(255)	NO		NULL	
jugador2	varchar(255)	NO		NULL	
resultado	varchar(255)	YES		NULL	
puntos_ganador	int	YES		NULL	

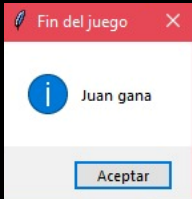
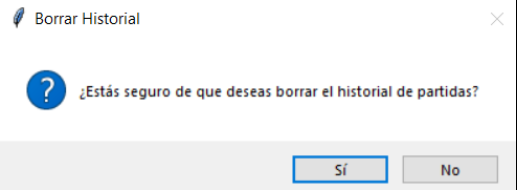
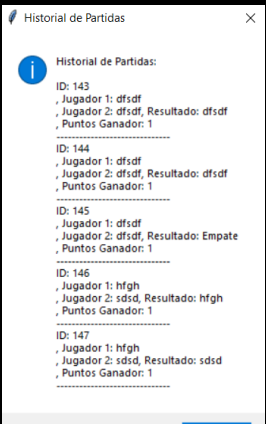
```
5 rows in set (0.01 sec)
```

- Tener los archivos de sonido "win\_sound.mp3" y "tie\_sound.mp3" (opcional para efectos de sonido) y modificar la dirección de carpeta si es necesario para habilitar los sonidos.
 

```
| self.win_sound = pygame.mixer.Sound("win_sound.mp3")
```

```
| self.tie_sound = pygame.mixer.Sound("tie_sound.mp3")
```
- Ejecutar el script de Python para iniciar el juego. El juego se abrirá en una ventana de tkinter.

#### Procedimientos de prueba más importantes:

- El juego en ejecución debería mostrar la ventana principal con la interfaz del juego y todos sus elementos funcionando.
- La lógica del juego funciona según las reglas del tatetí, por lo que la puntuación se actualiza automáticamente a medida que un jugador gana la partida.
 
- La función de borrar el historial de partidas debe eliminar los registros de la base de datos y mostrar un mensaje de confirmación.
 
- Al iniciar un nuevo juego los nombres de los jugadores y las puntuaciones se restablecen.
- El historial de partidas debe mostrar una lista de partidas anteriores almacenadas en la base de datos.
 
- Si los efectos de sonido fueron habilitados, se deben reproducir cuando alguien gane o haya un empate en el juego.

## FLUJO DEL JUEGO



**START**

- > El juego solicita a los jugadores que ingresen sus nombres.
- > Los nombres se almacenan para su uso en el juego.
- > Juegan una partida.
- > El juego lleva un registro de la partida y determina al ganador.
- > Después de cada partida, se suma un punto al jugador ganador.
- > Los puntos se almacenan en la base de datos junto con los nombres de los jugadores y el resultado de la partida.
- > Si la partida termina en empate o los jugadores eligen reiniciar antes de un resultado, el tablero se restablece y se inicia una nueva partida.
- > Se puede continuar jugando varias partidas, acumulando puntos a medida que se avanza.
- > Cuando los jugadores deciden cerrar la ventana del juego o seleccionan "Juego Nuevo," el juego determina al ganador en función de los puntos acumulados.
- > Se muestra un mensaje que anuncia al ganador, junto con la cantidad de puntos que ha acumulado.
- > Los jugadores van sumando puntos a medida que juegan varias partidas
- > Cuando cierran la ventana o ponen "Juego nuevo" tiene que salir un cartel que diga que el ganador fue el que sumó más puntos.
- > El resultado se guarda en el ID mostrando nombre del jugador 1, nombre del jugador 2, y el nombre del ganador con la cantidad de puntos que logró acumular.



**END.**

## ANÁLISIS DEL CÓDIGO

```
import tkinter as tk
```

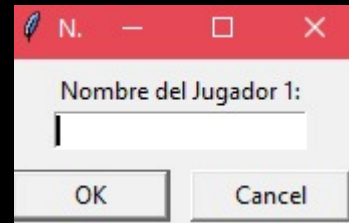
Utilizamos tkinter para crear la ventana principal y los elementos de la interfaz gráfica.

```
class Tateti:
```

Creamos la base del juego definiendo la clase principal que se llamará Tateti.

```
self.player1_name =  
self.get_player_name("Nombre del Jugador  
1:")
```

```
self.player2_name =  
self.get_player_name("Nombre del Jugador  
2:")
```



Los nombres de los jugadores se obtienen mediante ventanas emergentes utilizando `simpledialog`.

```
self.board = [" " for _ in range(9)]
```

El tablero de juego se representa como una lista de 9 elementos (`self.board`) que comienza con espacios en blanco para cada casilla.

```
button = tk.Button(self.window, text="", font=("Helvetica",  
20, "bold"), width=8, height=3, bg="blue", fg="yellow",  
command=lambda row=i, col=j: self.on_click(row, col))
```

Cada casilla del tablero se representa como un botón en la interfaz gráfica.

```
self.board[index] = self.current_player
```

Cuando un jugador hace clic en una casilla vacía, se coloca su ficha en la casilla.

```
self.check_winner()
```

Se verifica si hay un ganador comprobando todas las combinaciones ganadoras posibles en el tablero.

```
if " " not in self.board:
```

Se detecta si hay un empate si no quedan más movimientos posibles.

```
self.current_player = "O" if self.current_player == "X" else  
"X"
```

La lógica se encarga de alternar entre los jugadores "X" y "O" en cada turno.

```
self.player1_score = 0
```

```
self.player2_score = 0
```

```
self.ins_res = "INSERT INTO partidas (jugador1, jugador2,  
resultado, puntos_ganador) VALUES (%s, %s, %s, %s)"
```

```
self.cursor.execute(self.ins_res, self.res)
```

```
self.conn.commit()
```

Se lleva un registro de la puntuación de los jugadores y almacena los resultados en la base de datos.

```
pygame.mixer.init()
```

```
self.win_sound = pygame.mixer.Sound("win_sound.mp3")
```

```
self.tie_sound = pygame.mixer.Sound("tie_sound.mp3")
```

Se reproducen sonidos cuando un jugador gana o hay un empate.

```
clear_history_button = tk.Button(self.window, text="Borrar  
Historial", font=("Helvetica", 15),  
command=self.clear_history, bg="red")
```

```
restart_button = tk.Button(self.window, text="Reiniciar",  
font=("Helvetica", 15), command=self.reset_game, bg="yellow")
```

```
new_game_button = tk.Button(self.window, text="Nuevo Juego",  
font=("Helvetica", 15), command=self.new_game, bg="yellow")
```

Se incluyen botones para borrar el historial de partidas, reiniciar el juego actual y comenzar un nuevo juego.

```
self.show_history()
```

Se puede ver el historial de partidas que aparecerá en una ventana emergente.

```
self.new_game()
```

Cuando un juego termina, se muestra un mensaje con el resultado y se ofrece la opción de ingresar el nombre de los nuevos jugadores.

| Integrantes del proyecto:

Correa, Danilo Nehuen

Lopez Arsich, Maitena

Ramirez, Juan Cruz

Terenzi, Magali Alejandra