

Proyecto Final

INTEGRANTES

Raymond Lee Chavez Bravo

AL02952500

Ediel Olivero Gonzalez

AL02931247



CLASE

Base de datos

5/12/24

Proyecto Final: Sistema de Inscripciones para Torneo de Fútbol

Descripción del Proyecto:

El objetivo de este proyecto es diseñar e implementar un sistema de inscripciones para un torneo de fútbol utilizando MongoDB como base de datos y un framework para el frontend, preferiblemente en Python. El sistema debe permitir la inscripción de equipos en diferentes categorías según la edad, sexo y experiencia de los jugadores.

Requisitos Técnicos:

1. Base de Datos:

- Utilizamos MongoDB como la fuente principal para guardar los datos

1. Backend:

- Utilizamos Python para hacer el manejo de registros, consultas, etc.

1. Frontend:

- Utilizamos un import de Python (Tkinter) para poder hacer un interfaz facil y simple para nuestro programa.

Documentación:

EVIDENCIA3

ADMINS

personID
personName
idPassword
personType (admin / user)

PLAYERS

playerID
playerName
playerAge
playerSex
playerLevel
teamName (fk)

TEAMS

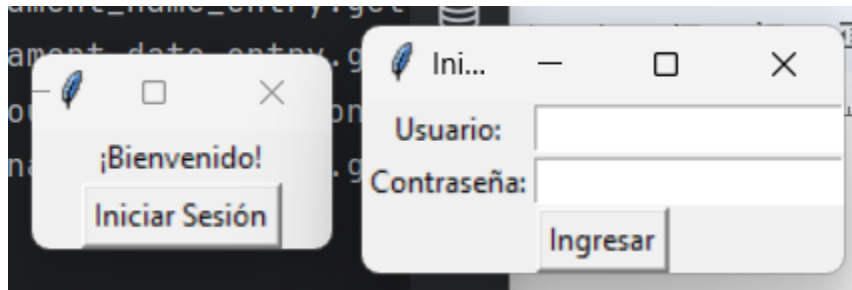
teamName
teamLevel
tournamentName (fk / opcional)

TOURNEY

tournamentName
tournamentDate
tournamentLocation
tournamenetStatus
tournamentLevel
tournamentTeams

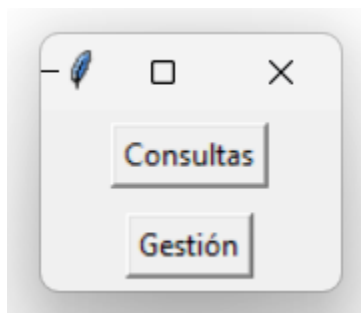


Pantalla Ingresar



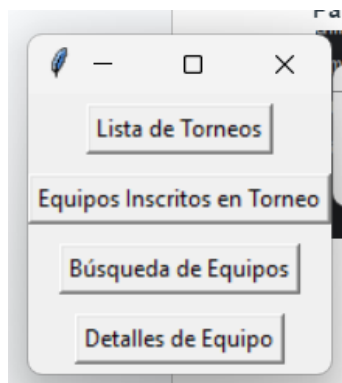
- Introducir el usuario y contraseña
 - *Paco > testing2 (nivel de usuario)*
 - *Raymond > testing (nivel de admin)*

Pantalla Inicial



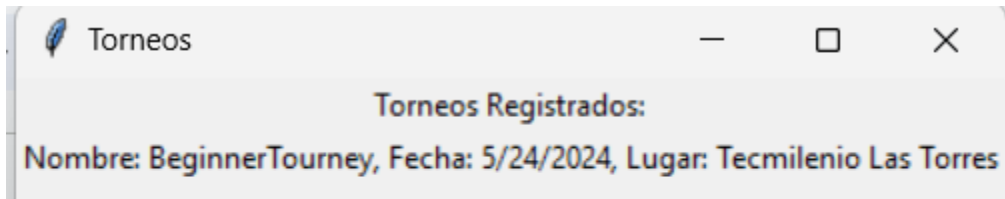
- Decidir entre ir a Consultas o Gestionar datos

CONSULTAS



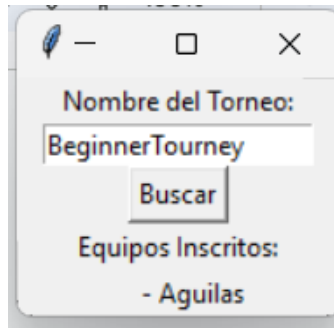
- **Lista de Torneos**

- Ver la lista completa de torneos inactivos / activos



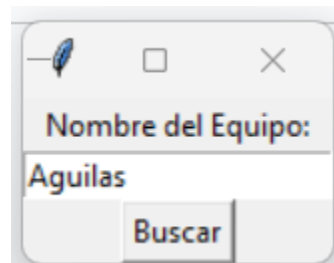
- **Equipos Inscritos en Torneo**

- Checar equipos inscritos dentro de un torneo (base al nombre del torneo)



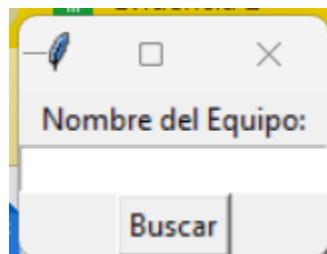
- **Busqueda de Equipos**

- Ver equipos y sus integrantes (base al nombre del equipo)



- **Detalles de Equipo**

- Ver los equipos con mas detalle



GESTION DE DATOS

A screenshot of a software window titled 'Me...' with standard Windows window controls (minimize, maximize, close). The window contains three sections of buttons:

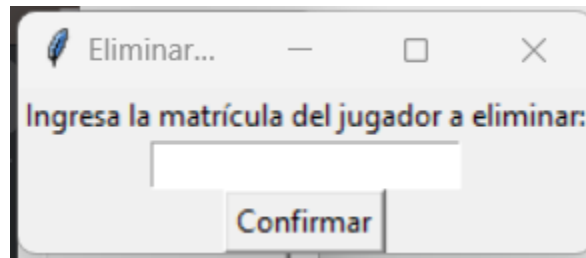
- Jugadores:**
 - Crear Jugador
 - Eliminar Jugador
 - Actualizar Jugador
- Equipos:**
 - Crear Equipo
 - Eliminar Equipo
 - Actualizar Equipos
- Torneos:**
 - Crear Torneo
 - Inscribir al Torneo
 - Eliminar Torneo

- Crear Jugador

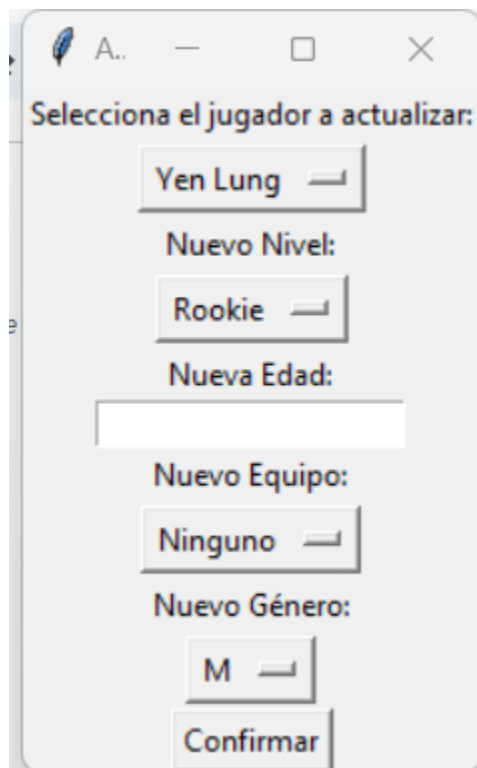
A screenshot of a software window titled 'Crear Pe...' with standard Windows window controls. It contains a form for creating a new player with the following fields and options:

- Nombre:** A text input field.
- Edad:** A text input field.
- Sexo:** A dropdown menu with 'M' selected.
- Nivel:** A dropdown menu with 'Rookie' selected.
- Equipo (opcional):** A dropdown menu with 'Ninguno' selected.
- Crear:** A button at the bottom right to submit the form.

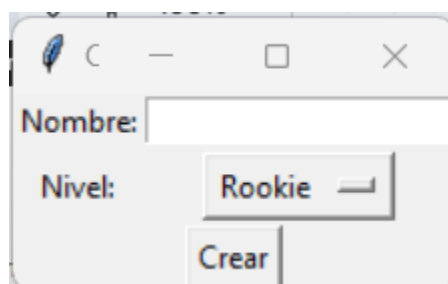
- **Eliminar Jugador**



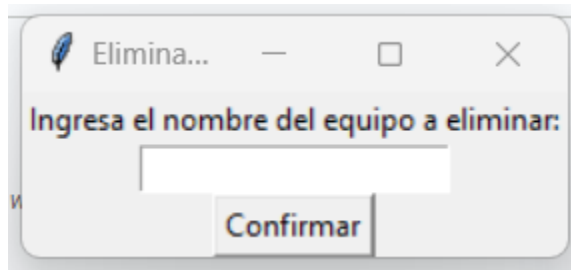
- Se usa matricula, ya que pueden existir jugadores con mismos datos como nombre, edad, sexo, etc.
- **Actualizar Jugador** (base a Jugadores ya existentes)



- **Crear Equipo**

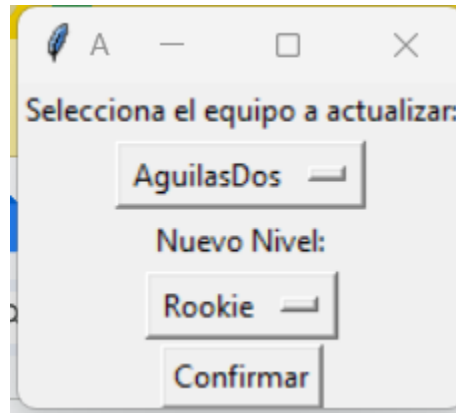


- **Eliminar Equipo**



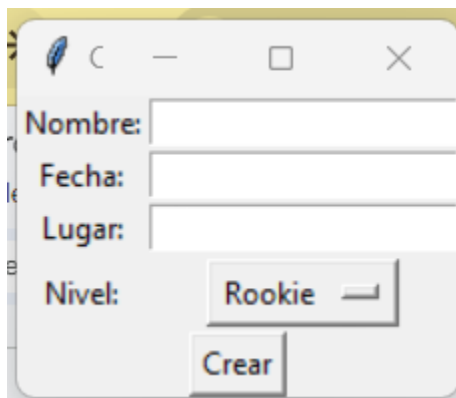
A screenshot of a Windows-style dialog box titled "Elimina...". It contains the text "Ingresa el nombre del equipo a eliminar:" followed by a text input field. Below the input field is a button labeled "Confirmar".

- **Actualizar Equipo** *(base a equipos ya existentes)*



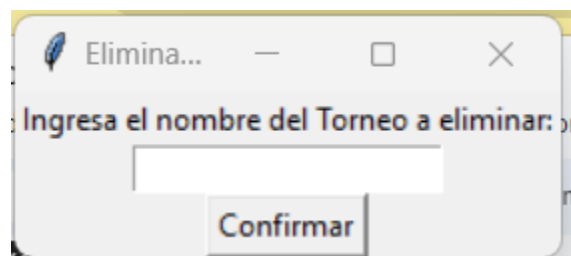
A screenshot of a Windows-style dialog box titled "A". It contains the text "Selecciona el equipo a actualizar:" followed by a dropdown menu showing "AguilasDos". Below this is the text "Nuevo Nivel:" followed by another dropdown menu showing "Rookie". At the bottom is a button labeled "Confirmar".

- **Crear Torneo**



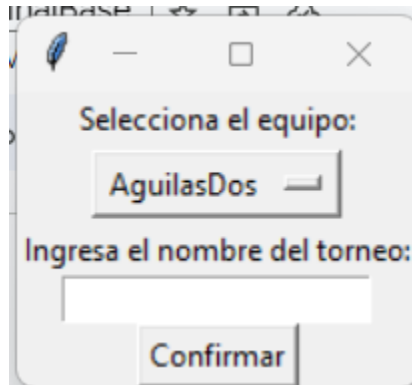
A screenshot of a Windows-style dialog box titled "C". It contains four labels with corresponding input fields: "Nombre:", "Fecha:", "Lugar:", and "Nivel:". The "Nivel:" dropdown menu is open, showing "Rookie". At the bottom is a button labeled "Crear".

- **Eliminar Torneo**



A screenshot of a Windows-style dialog box titled "Elimina...". It contains the text "Ingresa el nombre del Torneo a eliminar:" followed by a text input field. Below the input field is a button labeled "Confirmar".

- **Inscribir Torneo** (Solo se puede inscribir a un torneo si el equipo es de igual nivel que el torneo, se puede checar el nivel de un torneo via consultas)



A screenshot of a Java Swing dialog box titled "Inscribir Torneo". The dialog box has a standard window title bar with a blue icon, a minus sign, a square icon, and a close button (X). The main content area contains the text "Selecciona el equipo:" followed by a text field containing "AguilasDos". Below this is the text "Ingresa el nombre del torneo:" followed by an empty text field. At the bottom right of the dialog box is a button labeled "Confirmar".

Requisitos Funcionales:

1. Autenticación y Autorización:

- Pantalla de inicio de sesión para los usuarios.
- Validación de credenciales y asignación de roles (administrador, usuario regular).

Dentro de nuestra base, esta información no será editable por el UI (Frontend) sino, tendrá que ser administrador del base de datos y insertar los datos desde ahí.

Estos datos NUNCA serán editables por el programa.

```
_id: ObjectId('663ff071ded71fdbb94edfb8')
personID : 1
personName : "Raymond"
idPassword : "testing"
personType : "admin"
```

```
_id: ObjectId('663ff071ded71fdbb94edfb9')
personID : 2
personName : "Ediel"
idPassword : "hello"
personType : "admin"
```

```
_id: ObjectId('663ff071ded71fdbb94edfba')
personID : 3
personName : "Paco"
idPassword : "testing2"
personType : "normal"
```

2. Gestión de Equipos:

- Creación, edición y eliminación de equipos.
- Asignación de categoría por edad, sexo y experiencia a los equipos (Amateur, semi-pro, profesional y Mater).
- Consulta de equipos por categoría.

Clasificamos con 4 niveles, se pueden ver mas arriba de este reporte, usamos insertOne, dropOne, etc. comandos de Mongo en Python usando Pymongo

UI:

```
def create_team(self):
    self.team_window = tk.Toplevel(self.master)
    self.team_window.title("Crear Equipo")

    tk.Label(self.team_window, text="Nombre:").grid(row=0, column=0)
    self.team_name_entry = tk.Entry(self.team_window)
    self.team_name_entry.grid(row=0, column=1)

    tk.Label(self.team_window, text="Nivel:").grid(row=1, column=0)
    self.team_level_var = tk.StringVar(self.team_window)
    self.team_level_var.set("Rookie")
    self.team_level_menu = tk.OptionMenu(self.team_window, self.team_level_var, value="Rookie", *values: "Beginner", "Intermediate", "Expert")
    self.team_level_menu.grid(row=1, column=1)

    create_button = tk.Button(self.team_window, text="Crear", command=self.save_team)
    create_button.grid(row=2, columnspan=2)
```

Pymongo:

```
1 usage
def save_team(self):
    team_name = self.team_name_entry.get()
    team_level = self.team_level_var.get()

    new_team = {
        "teamName": team_name,
        "teamLevel": team_level
    }

    teams_collection.insert_one(new_team)

    messagebox.showinfo(title="Éxito", message=f"Equipo {team_name} creado exitosamente.")
```

Drop y Update:

```
class TorneoApp:
    def drop_team(self):
        self.drop_team_window = tk.Toplevel(self.master)
        self.drop_team_window.title("Eliminar Equipo")

        tk.Label(self.drop_team_window, text="Ingresa el nombre del equipo a eliminar:").pack()
        self.team_name_entry = tk.Entry(self.drop_team_window)
        self.team_name_entry.pack()

        confirm_button = tk.Button(self.drop_team_window, text="Confirmar", command=self.confirm_team_drop)
        confirm_button.pack()

    1 usage
    def confirm_team_drop(self):
        team_name = self.team_name_entry.get()

        result = players_collection.update_many(filter={"teamName": team_name}, update={"$set": {"teamName": "Ninguno"}})
        if result.modified_count > 0:
            messagebox.showinfo(title="Éxito", message=f"Se han actualizado {result.modified_count} jugadores con el equipo eliminado.")
        else:
            messagebox.showinfo(title="Info", message="No se encontraron jugadores asociados al equipo eliminado.")

        result = teams_collection.delete_one({"teamName": team_name})
        if result.deleted_count > 0:
            messagebox.showinfo(title="Éxito", message=f"Equipo {team_name} eliminado exitosamente.")
        else:
            messagebox.showerror(title="Error", message=f"No se encontró al equipo {team_name}.")

        self.drop_team_window.destroy()
```

```
def update_teams(self):
    self.update_teams_window = tk.Toplevel(self.master)
    self.update_teams_window.title("Actualizar Equipos")

    tk.Label(self.update_teams_window, text="Selecciona el equipo a actualizar:").pack()

    team_names = [team['teamName'] for team in teams_collection.find()]

    self.team_var = tk.StringVar(self.update_teams_window)
    self.team_var.set(team_names[0]) # Setear el primer equipo por defecto
    self.team_menu = tk.OptionMenu(self.update_teams_window, self.team_var, *values: *team_names)
    self.team_menu.pack()

    tk.Label(self.update_teams_window, text="Nuevo Nivel:").pack()
    self.new_team_level_var = tk.StringVar(self.update_teams_window)
    self.new_team_level_var.set("Rookie")
    self.new_team_level_menu = tk.OptionMenu(self.update_teams_window, self.new_team_level_var, value="Rookie",
                                              *values: "Beginner", "Intermediate", "Expert")
    self.new_team_level_menu.pack()

    confirm_button = tk.Button(self.update_teams_window, text="Confirmar", command=self.confirm_team_update)
    confirm_button.pack()
```

3. Gestión de Jugadores:

- Registro de jugadores con información personal (nombre, edad, sexo, experiencia, etc.).
- Asignación de jugadores a equipos existentes.
- Verificación de la edad mínima y máxima para cada categoría.

No quisimos limitar las categorías, ya que puede haber personas de 18~ que apenas empiezan.

Igual, confirmamos los otros detalles

Pedir los valores del jugador en Tkinter:

```
def create_person(self):
    self.person_window = tk.Toplevel(self.master)
    self.person_window.title("Crear Persona")

    tk.Label(self.person_window, text="Nombre:").grid(row=0, column=0)
    self.player_name_entry = tk.Entry(self.person_window)
    self.player_name_entry.grid(row=0, column=1)

    tk.Label(self.person_window, text="Edad:").grid(row=1, column=0)
    self.player_age_entry = tk.Entry(self.person_window)
    self.player_age_entry.grid(row=1, column=1)

    tk.Label(self.person_window, text="Sexo:").grid(row=2, column=0)
    self.player_sex_var = tk.StringVar(self.person_window)
    self.player_sex_var.set("M")
    self.player_sex_menu = tk.OptionMenu(self.person_window, self.player_sex_var, value="M", *values: "F", "Other")
    self.player_sex_menu.grid(row=2, column=1)

    tk.Label(self.person_window, text="Nivel:").grid(row=3, column=0)
    self.player_level_var = tk.StringVar(self.person_window)
    self.player_level_var.set("Rookie")
    self.player_level_menu = tk.OptionMenu(self.person_window, self.player_level_var, value="Rookie", *values: "Beginner", "Intermediate", "Expert")
    self.player_level_menu.grid(row=3, column=1)

    tk.Label(self.person_window, text="Equipo (opcional):").grid(row=4, column=0)
    team_names = ["Ninguno"] + [team['teamName'] for team in teams_collection.find()]
```

Guardar el jugador en el Base:

```
def save_person(self):
    player_id = random.randint(a=10000, b=25000)
    player_name = self.player_name_entry.get()
    player_age = int(self.player_age_entry.get())
    player_sex = self.player_sex_var.get()
    player_team = self.player_team_var.get()
    player_level = self.player_level_var.get()

    if player_team != "Ninguno":
        team = teams_collection.find_one({"teamName": player_team})
        if team:
            if player_level != team["teamLevel"]:
                messagebox.showerror(title="Error", message="El nivel del Equipo es mayor o menor que el del Jugador!")
                return

    new_player = {
        "playerID": player_id,
        "playerName": player_name,
        "playerAge": player_age,
        "playerSex": player_sex,
        "playerLevel": player_level,
        "teamName": player_team
    }

    players_collection.insert_one(new_player)

    messagebox.showinfo(title="Exito", message=f"Persona creada exitosamente. Su matricula es: {player_id}")
```

Drop:

```
def drop_person(self):
    self.drop_person_window.title("Eliminar Persona")

    tk.Label(self.drop_person_window, text="Ingresa la matrícula del jugador a eliminar:").pack()
    self.player_id_entry = tk.Entry(self.drop_person_window)
    self.player_id_entry.pack()

    confirm_button = tk.Button(self.drop_person_window, text="Confirmar", command=self.confirm_player_drop)
    confirm_button.pack()

1 usage
def confirm_player_drop(self):
    player_id = self.player_id_entry.get()

    # Validate if the player ID is numeric
    if not player_id.isdigit():
        messagebox.showerror( title: "Error", message: "La matrícula del jugador debe ser un número.")
        return

    result = players_collection.delete_one({"playerID": int(player_id)})
    if result.deleted_count > 0:
        messagebox.showinfo( title: "Éxito", message: f"Jugador #{player_id} eliminado exitosamente.")
    else:
        messagebox.showerror( title: "Error", message: f"No se encontró al jugador #{player_id}.")

    self.drop_person_window.destroy()
```

4. Inscripción en el Torneo:

- Proceso de inscripción de equipos en el torneo.
- Validación de requisitos de elegibilidad (edad, sexo, experiencia).
- Asignación automática de categoría según los jugadores inscritos en el equipo.

Lo unico que limitamos es si el nivel del torneo es Rookie, solo equipos Rookie pueden inscribirse.

Primero hacemos funciones para poder CREAR un torneo:

```
def create_tournament(self):
    self.tournament_window = tk.Toplevel(self.master)
    self.tournament_window.title("Crear Torneo")

    tk.Label(self.tournament_window, text="Nombre:").grid(row=0, column=0)
    self.tournament_name_entry = tk.Entry(self.tournament_window)
    self.tournament_name_entry.grid(row=0, column=1)

    tk.Label(self.tournament_window, text="Fecha:").grid(row=1, column=0)
    self.tournament_date_entry = tk.Entry(self.tournament_window)
    self.tournament_date_entry.grid(row=1, column=1)

    tk.Label(self.tournament_window, text="Lugar:").grid(row=2, column=0)
    self.tournament_location_entry = tk.Entry(self.tournament_window)
    self.tournament_location_entry.grid(row=2, column=1)

    tk.Label(self.tournament_window, text="Nivel:").grid(row=4, column=0)
    self.tournament_level_var = tk.StringVar(self.tournament_window)
    self.tournament_level_var.set("Rookie")
    self.tournament_level_menu = tk.OptionMenu(self.tournament_window, self.tournament_level_var, value="Rookie", *values="Beginner", "Intermediate",
    self.tournament_level_menu.grid(row=4, column=1)

    create_button = tk.Button(self.tournament_window, text="Crear", command=self.save_tournament)
    create_button.grid(row=5, columnspan=2)
```

```
def save_tournament(self):
    tournament_name = self.tournament_name_entry.get()
    tournament_date = self.tournament_date_entry.get()
    tournament_location = self.tournament_location_entry.get()
    tournament_level = self.tournament_level_var.get()

    new_tournament = {
        "tournamentName": tournament_name,
        "tournamentDate": tournament_date,
        "tournamentLocation": tournament_location,
        "tournamentStatus": "inactive",
        "tournamentLevel": tournament_level,
        "tournamentTeams": []
    }

    tourney_collection.insert_one(new_tournament)

    messagebox.showinfo(title="Éxito", message=f"Torneo {tournament_name} creado exitosamente.")
```

Inscribir a un Torneo:

```
def enter_tournament(self):
    self.enter_tournament_window = tk.Toplevel(self.master)
    self.enter_tournament_window.title("Inscribir Equipo al Torneo")

    tk.Label(self.enter_tournament_window, text="Selecciona el equipo:").pack()

    team_names = [team['teamName'] for team in teams_collection.find()]

    self.team_var = tk.StringVar(self.enter_tournament_window)
    self.team_var.set(team_names[0]) # Setear el primer equipo por defecto
    self.team_menu = tk.OptionMenu(self.enter_tournament_window, self.team_var, *team_names)
    self.team_menu.pack()

    tk.Label(self.enter_tournament_window, text="Ingresa el nombre del torneo:").pack()
    self.tournament_name_entry = tk.Entry(self.enter_tournament_window)
    self.tournament_name_entry.pack()

    confirm_button = tk.Button(self.enter_tournament_window, text="Confirmar",
                               command=self.confirm_entry_tournament)
    confirm_button.pack()
```

```
def confirm_entry_tournament(self):
    team_name = self.team_var.get()
    tournament_name = self.tournament_name_entry.get()

    team = teams_collection.find_one({"teamName": team_name})
    tournament = tourney_collection.find_one({"tournamentName": tournament_name})

    if team and tournament:
        if team["teamLevel"] != tournament["tournamentLevel"]:
            messagebox.showerror(title="Error", message="El nivel del equipo no coincide con el nivel del torneo.")
            return
        confirm = messagebox.askyesno(title="Confirmar Inscripción",
                                     message=f"¿Seguro que deseas inscribir al equipo {team_name} al torneo {tournament_name}?")
        if confirm:
            tourney_collection.update_one(filter={"tournamentName": tournament_name},
                                         update={"$addToSet": {"tournamentTeams": team_name}})
            messagebox.showinfo(title="Éxito", message=f"Equipo {team_name} inscrito al torneo {tournament_name} exitosamente.")
    else:
        messagebox.showerror(title="Error", message="Equipo o Torneo no encontrado.")

    self.enter_tournament_window.destroy()
```


Drop Torneo:

```
def drop_tournament(self):
    self.drop_tournament_window = tk.Toplevel(self.master)
    self.drop_tournament_window.title("Eliminar Torneo")

    tk.Label(self.drop_tournament_window, text="Ingresa el nombre del Torneo a eliminar:").pack()
    self.tournament_name_entry = tk.Entry(self.drop_tournament_window)
    self.tournament_name_entry.pack()

    confirm_button = tk.Button(self.drop_tournament_window, text="Confirmar", command=self.confirm_tournament_drop)
    confirm_button.pack()

1 usage
def confirm_tournament_drop(self):
    tournament_name = self.tournament_name_entry.get()

    result = tourney_collection.delete_one({"tournamentName": tournament_name})
    if result.deleted_count > 0:
        messagebox.showinfo(title="Éxito", message=f"Torneo {tournament_name} eliminado exitosamente.")
    else:
        messagebox.showerror(title="Error", message=f"No se encontró el torneo {tournament_name}.")

    self.drop_tournament_window.destroy()
```

5. Consultas:

- Consulta de equipos inscritos en el torneo.
- Búsqueda de equipos por nombre, categoría, sexo, etc.
- Visualización de detalles de cada equipo (jugadores, categoría, etc.).

Usamos funciones basicas del MongoDB para poder realizar las consultas que pide:

1 - Mostrar Equipos y sus Integrantes

```
1 usage
def show_tournaments(self):
    self.tournaments_window = tk.Toplevel(self.master)
    self.tournaments_window.title("Torneos")

    tournaments = tourney_collection.find()

    tk.Label(self.tournaments_window, text="Torneos Registrados:").pack()
    for tournament in tournaments:
        tk.Label(self.tournaments_window,
            text=f"Nombre: {tournament['tournamentName']}, Fecha: {tournament['tournamentDate']}, Lugar: {tournament['tournamentLocation']}").pack
```

2 - Equipos Inscritos

```
1 usage
def show_tournament_teams(self):
    self.tournament_teams_window = tk.Toplevel(self.master)
    self.tournament_teams_window.title("Equipos Inscritos en Torneo")

    tk.Label(self.tournament_teams_window, text="Nombre del Torneo:").pack()
    self.tournament_name_entry = tk.Entry(self.tournament_teams_window)
    self.tournament_name_entry.pack()

    confirm_button = tk.Button(self.tournament_teams_window, text="Buscar",
        command=self.show_tournament_teams_result)
    confirm_button.pack()
```

3 - Detalles del Equipo

```
1 usage
def show_tournament_teams_result(self):
    tournament_name = self.tournament_name_entry.get()

    tournament = tourney_collection.find_one({"tournamentName": tournament_name})
    if tournament:
        tk.Label(self.tournament_teams_window, text="Equipos Inscritos:").pack()
        for team in tournament['tournamentTeams']:
            tk.Label(self.tournament_teams_window, text=f"    - {team}").pack()
    else:
        messagebox.showerror(title="Error", message="Torneo no encontrado.")
```

4 - Mostrar detalles del Equipo

```
def show_team_details(self):
    confirm_button = tk.Button(self.team_details_window, text="Buscar",
                                command=self.show_team_details_result)
    confirm_button.pack()

1 usage
def show_team_details_result(self):
    team_name = self.team_name_entry.get()

    team = teams_collection.find_one({"teamName": team_name})
    if team:
        tk.Label(self.team_details_window, text=f"Nombre: {team['teamName']}").pack()
        tk.Label(self.team_details_window, text=f"Nivel: {team['teamLevel']}").pack()

        # Consultar jugadores del equipo
        players = players_collection.find({"teamName": team_name})
        if players:
            tk.Label(self.team_details_window, text="Jugadores:").pack()
            for player in players:
                tk.Label(self.team_details_window,
                           text=f"    - Nombre: {player['playerName']}, Edad: {player['playerAge']}, Sexo: {player['playerSex']}, Nivel: {player['playerL
            else:
                tk.Label(self.team_details_window, text="No hay jugadores registrados para este equipo.").pack()
        else:
            messagebox.showerror(title="Error", message="Equipo no encontrado.")
```

Conclusiones:

Raymond:

En fin, fue algo interesante poder ver todo como trabaja un base de datos conjunto con un programa, ya que es interesante pensar en que así sirve la mayoría de programas con muchas llaves puestas.

Ediel:

Para concluir debería decir que este proyecto tomo mucho tiempo para ser desarrollado, no es una tarea complicada, solamente fue tardada, intentamos abarcar todo el contenido visto durante el semestre en la materia, así como los conocimientos de materias pasadas para la realización en otros lenguajes.

Codigo Sig. Pagina


```

import random
import tkinter as tk
from tkinter import messagebox
from pymongo import MongoClient

url = "mongodb://localhost:27017"
cluster = MongoClient(url)
db = cluster['Evidencia3']

users_collection = db['admins']
players_collection = db['players']
teams_collection = db['teams']
tourney_collection = db['tourney']

players_collection.create_index([("playerID", 1)], unique=True)
tourney_collection.create_index([("tournamentName", 1)], unique=True)
teams_collection.create_index([("teamName", 1)], unique=True)

class TorneoApp:
    def __init__(self, master):
        self.master = master
        master.title("TorneoApp")

        self.label = tk.Label(master, text="¡Bienvenido!")
        self.label.pack()

        self.login_button = tk.Button(master, text="Iniciar Sesión",
command=self.login)
        self.login_button.pack()

    def login(self):
        self.login_window = tk.Toplevel(self.master)
        self.login_window.title("Inicio de Sesión")

        self.username_label = tk.Label(self.login_window, text="Usuario:")
        self.username_label.grid(row=0, column=0)
        self.username_entry = tk.Entry(self.login_window)
        self.username_entry.grid(row=0, column=1)

        self.password_label = tk.Label(self.login_window, text="Contraseña:")
        self.password_label.grid(row=1, column=0)
        self.password_entry = tk.Entry(self.login_window, show="*")
        self.password_entry.grid(row=1, column=1)

        self.login_button = tk.Button(self.login_window, text="Ingresar",
command=self.validate_login)
        self.login_button.grid(row=2, columnspan=2)

    def validate_login(self):

```

```

        username = self.username_entry.get()
        password = self.password_entry.get()

        user = users_collection.find_one({"personName": username, "idPassword":
password})
        if user:
            messagebox.showinfo("Exitoso!", f"Bienvenido, {username}!")
            self.show_initial_menu()
        else:
            messagebox.showerror("Error", "Credenciales inválidas! Intenta de
nuevo!")

    def show_initial_menu(self):
        self.initial_menu = tk.Toplevel(self.master)
        self.initial_menu.title("Menú Inicial")

        tk.Button(self.initial_menu, text="Consultas",
command=self.show_consults).pack(pady=5)
        tk.Button(self.initial_menu, text="Gestión",
command=self.show_main_menu).pack(pady=5)

    def show_main_menu(self):
        username = self.username_entry.get()
        password = self.password_entry.get()

        user = users_collection.find_one({"personName": username, "idPassword":
password})
        if user["personType"] == "admin":
            self.admin_menu = tk.Toplevel(self.master)
            self.admin_menu.title("Menú de Administrador")

            tk.Label(self.admin_menu, text="Jugadores:").pack()
            tk.Button(self.admin_menu, text="Crear Jugador",
command=self.create_person).pack(pady=5)
            tk.Button(self.admin_menu, text="Eliminar Jugador",
command=self.drop_person).pack(pady=5)
            tk.Button(self.admin_menu, text="Actualizar Jugador",
command=self.update_players).pack(pady=5)

            tk.Label(self.admin_menu, text="Equipos:").pack()
            tk.Button(self.admin_menu, text="Crear Equipo",
command=self.create_team).pack(pady=5)
            tk.Button(self.admin_menu, text="Eliminar Equipo",
command=self.drop_team).pack(pady=5)
            tk.Button(self.admin_menu, text="Actualizar Equipos",
command=self.update_teams).pack(pady=5)

            tk.Label(self.admin_menu, text="Torneos:").pack()

```

```

        tk.Button(self.admin_menu, text="Crear Torneo",
command=self.create_tournament).pack(pady=5)
        tk.Button(self.admin_menu, text="Inscribir al Torneo",
command=self.enter_tournament).pack(pady=5)
        tk.Button(self.admin_menu, text="Eliminar Torneo",
command=self.drop_tournament).pack(pady=5)

    else:
        self.user_menu = tk.Toplevel(self.master)
        self.user_menu.title("Menú de Usuario")

        tk.Button(self.user_menu, text="Crear Persona",
command=self.create_person).pack(pady=5)
        tk.Button(self.user_menu, text="Crear Equipo",
command=self.create_team).pack(pady=5)
        tk.Button(self.user_menu, text="Crear Torneo",
command=self.create_tournament).pack(pady=5)

    def show_consults(self):
        self.consults_menu = tk.Toplevel(self.master)
        self.consults_menu.title("Consultas")

        tk.Button(self.consults_menu, text="Lista de Torneos",
command=self.show_tournaments).pack(pady=5)
        tk.Button(self.consults_menu, text="Equipos Inscritos en Torneo",
command=self.show_tournament_teams).pack(pady=5)
        tk.Button(self.consults_menu, text="Búsqueda de Equipos",
command=self.search_teams).pack(pady=5)
        tk.Button(self.consults_menu, text="Detalles de Equipo",
command=self.show_team_details).pack(pady=5)

    def show_tournament_teams(self):
        self.tournament_teams_window = tk.Toplevel(self.master)
        self.tournament_teams_window.title("Equipos Inscritos en Torneo")

        tk.Label(self.tournament_teams_window, text="Nombre del Torneo:").pack()
        self.tournament_name_entry = tk.Entry(self.tournament_teams_window)
        self.tournament_name_entry.pack()

        confirm_button = tk.Button(self.tournament_teams_window, text="Buscar",
command=self.show_tournament_teams_result)
        confirm_button.pack()

    def show_tournament_teams_result(self):
        tournament_name = self.tournament_name_entry.get()

        tournament = tourney_collection.find_one({"tournamentName":
tournament_name})
        if tournament:

```



```

        tk.Label(self.tournament_teams_window, text="Equipos
Inscritos:").pack()
        for team in tournament['tournamentTeams']:
            tk.Label(self.tournament_teams_window, text=f"    -
{team}").pack()
        else:
            messagebox.showerror("Error", "Torneo no encontrado.")

def search_teams(self):
    self.search_teams_window = tk.Toplevel(self.master)
    self.search_teams_window.title("Búsqueda de Equipos")

    tk.Label(self.search_teams_window, text="Nombre del Equipo:").pack()
    self.team_name_entry = tk.Entry(self.search_teams_window)
    self.team_name_entry.pack()

    confirm_button = tk.Button(self.search_teams_window, text="Buscar",
                                command=self.search_teams_result)
    confirm_button.pack()

def search_teams_result(self):
    team_name = self.team_name_entry.get()

    team = teams_collection.find_one({"teamName": team_name})
    if team:
        tk.Label(self.search_teams_window, text=f"Nombre:
{team['teamName']}").pack()
        tk.Label(self.search_teams_window, text=f"Nivel:
{team['teamLevel']}").pack()
    else:
        messagebox.showerror("Error", "Equipo no encontrado.")

def show_team_details(self):
    self.team_details_window = tk.Toplevel(self.master)
    self.team_details_window.title("Detalles de Equipo")

    tk.Label(self.team_details_window, text="Nombre del Equipo:").pack()
    self.team_name_entry = tk.Entry(self.team_details_window)
    self.team_name_entry.pack()

    confirm_button = tk.Button(self.team_details_window, text="Buscar",
                                command=self.show_team_details_result)
    confirm_button.pack()

def show_team_details_result(self):
    team_name = self.team_name_entry.get()

    team = teams_collection.find_one({"teamName": team_name})
    if team:

```

```

        tk.Label(self.team_details_window, text=f"Nombre:
{team['teamName']}").pack()
        tk.Label(self.team_details_window, text=f"Nivel:
{team['teamLevel']}").pack()

    # Consultar jugadores del equipo
    players = players_collection.find({"teamName": team_name})
    if players:
        tk.Label(self.team_details_window, text="Jugadores:").pack()
        for player in players:
            tk.Label(self.team_details_window,
                    text=f"    - Nombre: {player['playerName']}, Edad:
{player['playerAge']}, Sexo: {player['playerSex']}, Nivel:
{player['playerLevel']}").pack()
        else:
            tk.Label(self.team_details_window, text="No hay jugadores
registrados para este equipo.").pack()
    else:
        messagebox.showerror("Error", "Equipo no encontrado.")

def create_person(self):
    self.person_window = tk.Toplevel(self.master)
    self.person_window.title("Crear Persona")

    tk.Label(self.person_window, text="Nombre:").grid(row=0, column=0)
    self.player_name_entry = tk.Entry(self.person_window)
    self.player_name_entry.grid(row=0, column=1)

    tk.Label(self.person_window, text="Edad:").grid(row=1, column=0)
    self.player_age_entry = tk.Entry(self.person_window)
    self.player_age_entry.grid(row=1, column=1)

    tk.Label(self.person_window, text="Sexo:").grid(row=2, column=0)
    self.player_sex_var = tk.StringVar(self.person_window)
    self.player_sex_var.set("M")
    self.player_sex_menu = tk.OptionMenu(self.person_window,
self.player_sex_var, "M", "F", "Other")
    self.player_sex_menu.grid(row=2, column=1)

    tk.Label(self.person_window, text="Nivel:").grid(row=3, column=0)
    self.player_level_var = tk.StringVar(self.person_window)
    self.player_level_var.set("Rookie")
    self.player_level_menu = tk.OptionMenu(self.person_window,
self.player_level_var, "Rookie", "Beginner", "Intermediate", "Expert")
    self.player_level_menu.grid(row=3, column=1)

    tk.Label(self.person_window, text="Equipo (opcional):").grid(row=4,
column=0)

```

```

        team_names = ["Ninguno"] + [team['teamName'] for team in
teams_collection.find()]

        self.player_team_var = tk.StringVar(self.person_window)
        self.player_team_var.set("Ninguno")
        self.player_team_menu = tk.OptionMenu(self.person_window,
self.player_team_var, *team_names)
        self.player_team_menu.grid(row=4, column=1)

        create_button = tk.Button(self.person_window, text="Crear",
command=self.save_person)
        create_button.grid(row=5, columnspan=2)

    def save_person(self):
        player_id = random.randint(10000,25000)
        player_name = self.player_name_entry.get()
        player_age = int(self.player_age_entry.get())
        player_sex = self.player_sex_var.get()
        player_team = self.player_team_var.get()
        player_level = self.player_level_var.get()

        while players_collection.find_one({"playerID": player_id}):
            player_id = random.randint(10000, 25000)

        if player_team != "Ninguno":
            team = teams_collection.find_one({"teamName": player_team})
            if team:
                if player_level != team["teamLevel"]:
                    messagebox.showerror("Error", "El nivel del Equipo es mayor
o menor que el del Jugador!")
                    return

        new_player = {
            "playerID": player_id,
            "playerName": player_name,
            "playerAge": player_age,
            "playerSex": player_sex,
            "playerLevel": player_level,
            "teamName": player_team
        }

        players_collection.insert_one(new_player)

        messagebox.showinfo("Exito", f"Persona creada exitosamente. Su matricula
es: {player_id}")

    def drop_person(self):
        self.drop_person_window = tk.Toplevel(self.master)
        self.drop_person_window.title("Eliminar Persona")

```

```

        tk.Label(self.drop_person_window, text="Ingresa la matrícula del jugador
a eliminar:").pack()
        self.player_id_entry = tk.Entry(self.drop_person_window)
        self.player_id_entry.pack()

        confirm_button = tk.Button(self.drop_person_window, text="Confirmar",
command=self.confirm_player_drop)
        confirm_button.pack()

    def confirm_player_drop(self):
        player_id = self.player_id_entry.get()

        # Validate if the player ID is numeric
        if not player_id.isdigit():
            messagebox.showerror("Error", "La matrícula del jugador debe ser un
número.")
            return

        result = players_collection.delete_one({"playerID": int(player_id)})
        if result.deleted_count > 0:
            messagebox.showinfo("Éxito", f"Jugador #{player_id} eliminado
exitosamente.")
        else:
            messagebox.showerror("Error", f"No se encontró al jugador
#{player_id}.")

        self.drop_person_window.destroy()

    def create_team(self):
        self.team_window = tk.Toplevel(self.master)
        self.team_window.title("Crear Equipo")

        tk.Label(self.team_window, text="Nombre:").grid(row=0, column=0)
        self.team_name_entry = tk.Entry(self.team_window)
        self.team_name_entry.grid(row=0, column=1)

        tk.Label(self.team_window, text="Nivel:").grid(row=1, column=0)
        self.team_level_var = tk.StringVar(self.team_window)
        self.team_level_var.set("Rookie")
        self.team_level_menu = tk.OptionMenu(self.team_window,
self.team_level_var, "Rookie", "Beginner", "Intermediate", "Expert")
        self.team_level_menu.grid(row=1, column=1)

        create_button = tk.Button(self.team_window, text="Crear",
command=self.save_team)
        create_button.grid(row=2, columnspan=2)

    def save_team(self):

```

```

team_name = self.team_name_entry.get()
team_level = self.team_level_var.get()

if teams_collection.find_one({"teamName": team_name}):
    messagebox.showerror("Error", "Ya existe un equipo con ese nombre!")
    return

new_team = {
    "teamName": team_name,
    "teamLevel": team_level
}

teams_collection.insert_one(new_team)

messagebox.showinfo("Éxito", f"Equipo {team_name} creado exitosamente.")

def drop_team(self):
    self.drop_team_window = tk.Toplevel(self.master)
    self.drop_team_window.title("Eliminar Equipo")

    tk.Label(self.drop_team_window, text="Ingresa el nombre del equipo a
eliminar:").pack()
    self.team_name_entry = tk.Entry(self.drop_team_window)
    self.team_name_entry.pack()

    confirm_button = tk.Button(self.drop_team_window, text="Confirmar",
command=self.confirm_team_drop)
    confirm_button.pack()

    def confirm_team_drop(self):
        team_name = self.team_name_entry.get()

        result = players_collection.update_many({"teamName": team_name},
{"$set": {"teamName": "Ninguno"}})
        if result.modified_count > 0:
            messagebox.showinfo("Éxito",
f"Se han actualizado {result.modified_count}
jugadores con el equipo eliminado.")
        else:
            messagebox.showinfo("Info", "No se encontraron jugadores asociados
al equipo eliminado.")

        result = teams_collection.delete_one({"teamName": team_name})
        if result.deleted_count > 0:
            messagebox.showinfo("Éxito", f"Equipo {team_name} eliminado
exitosamente.")
        else:
            messagebox.showerror("Error", f"No se encontró al equipo
{team_name}.")

```

```

        self.drop_team_window.destroy()

    def create_tournament(self):
        self.tournament_window = tk.Toplevel(self.master)
        self.tournament_window.title("Crear Torneo")

        tk.Label(self.tournament_window, text="Nombre:").grid(row=0, column=0)
        self.tournament_name_entry = tk.Entry(self.tournament_window)
        self.tournament_name_entry.grid(row=0, column=1)

        tk.Label(self.tournament_window, text="Fecha:").grid(row=1, column=0)
        self.tournament_date_entry = tk.Entry(self.tournament_window)
        self.tournament_date_entry.grid(row=1, column=1)

        tk.Label(self.tournament_window, text="Lugar:").grid(row=2, column=0)
        self.tournament_location_entry = tk.Entry(self.tournament_window)
        self.tournament_location_entry.grid(row=2, column=1)

        tk.Label(self.tournament_window, text="Nivel:").grid(row=4, column=0)
        self.tournament_level_var = tk.StringVar(self.tournament_window)
        self.tournament_level_var.set("Rookie")
        self.tournament_level_menu = tk.OptionMenu(self.tournament_window,
self.tournament_level_var, "Rookie", "Beginner", "Intermediate", "Expert")
        self.tournament_level_menu.grid(row=4, column=1)

        create_button = tk.Button(self.tournament_window, text="Crear",
command=self.save_tournament)
        create_button.grid(row=5, columnspan=2)

    def save_tournament(self):
        tournament_name = self.tournament_name_entry.get()
        tournament_date = self.tournament_date_entry.get()
        tournament_location = self.tournament_location_entry.get()
        tournament_level = self.tournament_level_var.get()

        if tourney_collection.find_one({"tournamentName": tournament_name}):
            messagebox.showerror("Error", "Ya existe un torneo con ese nombre!")
            return

        new_tournament = {
            "tournamentName": tournament_name,
            "tournamentDate": tournament_date,
            "tournamentLocation": tournament_location,
            "tournamentStatus": "inactive",
            "tournamentLevel": tournament_level,
            "tournamentTeams": []
        }

```

```

    tourney_collection.insert_one(new_tournament)

    messagebox.showinfo("Éxito", f"Torneo {tournament_name} creado exitosamente.")

    def enter_tournament(self):
        self.enter_tournament_window = tk.Toplevel(self.master)
        self.enter_tournament_window.title("Inscribir Equipo al Torneo")

        tk.Label(self.enter_tournament_window, text="Selecciona el equipo:").pack()

        team_names = [team['teamName'] for team in teams_collection.find()]

        self.team_var = tk.StringVar(self.enter_tournament_window)
        self.team_var.set(team_names[0]) # Setear el primer equipo por defecto
        self.team_menu = tk.OptionMenu(self.enter_tournament_window,
self.team_var, *team_names)
        self.team_menu.pack()

        tk.Label(self.enter_tournament_window, text="Ingresa el nombre del torneo:").pack()
        self.tournament_name_entry = tk.Entry(self.enter_tournament_window)
        self.tournament_name_entry.pack()

        confirm_button = tk.Button(self.enter_tournament_window,
text="Confirmar",
                                command=self.confirm_entry_tournament)
        confirm_button.pack()

    def confirm_entry_tournament(self):
        team_name = self.team_var.get()
        tournament_name = self.tournament_name_entry.get()

        team = teams_collection.find_one({"teamName": team_name})
        tournament = tourney_collection.find_one({"tournamentName": tournament_name})

        if team and tournament:
            if team["teamLevel"] != tournament["tournamentLevel"]:
                messagebox.showerror("Error", "El nivel del equipo no coincide con el nivel del torneo.")
                return
            confirm = messagebox.askyesno("Confirmar Inscripción",
f"Seguro que deseas inscribir al equipo {team_name} al torneo {tournament_name}?")
            if confirm:
                tourney_collection.update_one({"tournamentName": tournament_name},

```

```

{"$addToSet": {"tournamentTeams":
team_name}})
        messagebox.showinfo("Éxito", f"Equipo {team_name} inscrito al
torneo {tournament_name} exitosamente.")
    else:
        messagebox.showerror("Error", "Equipo o Torneo no encontrado.")

    self.enter_tournament_window.destroy()
def drop_tournament(self):
    self.drop_tournament_window = tk.Toplevel(self.master)
    self.drop_tournament_window.title("Eliminar Torneo")

    tk.Label(self.drop_tournament_window, text="Ingresa el nombre del Torneo
a eliminar:").pack()
    self.tournament_name_entry = tk.Entry(self.drop_tournament_window)
    self.tournament_name_entry.pack()

    confirm_button = tk.Button(self.drop_tournament_window,
text="Confirmar", command=self.confirm_tournament_drop)
    confirm_button.pack()

def confirm_tournament_drop(self):
    tournament_name = self.tournament_name_entry.get()

    result = tourney_collection.delete_one({"tournamentName":
tournament_name})
    if result.deleted_count > 0:
        messagebox.showinfo("Éxito", f"Torneo {tournament_name} eliminado
 exitosamente.")
    else:
        messagebox.showerror("Error", f"No se encontró el torneo
{tournament_name}.")

    self.drop_tournament_window.destroy()

def show_players(self):
    self.players_window = tk.Toplevel(self.master)
    self.players_window.title("Jugadores")

    players = players_collection.find()

    tk.Label(self.players_window, text="Jugadores Registrados:").pack()
    for player in players:
        tk.Label(self.players_window,
            text=f"ID: {player['playerID']}, Nombre:
{player['playerName']}, Edad: {player['playerAge']}, Sexo:
{player['playerSex']}, Nivel: {player['playerLevel']}, Equipo:
{player['teamName']}").pack()

```



```

def show_teams(self):
    self.teams_window = tk.Toplevel(self.master)
    self.teams_window.title("Equipos")

    teams = teams_collection.find()

    tk.Label(self.teams_window, text="Equipos Registrados:").pack()
    for team in teams:
        tk.Label(self.teams_window, text=f"Nombre: {team['teamName']},
Nivel: {team['teamLevel']}").pack()

def show_tournaments(self):
    self.tournaments_window = tk.Toplevel(self.master)
    self.tournaments_window.title("Torneos")

    tournaments = tourney_collection.find()

    tk.Label(self.tournaments_window, text="Torneos Registrados:").pack()
    for tournament in tournaments:
        tk.Label(self.tournaments_window,
            text=f"Nombre: {tournament['tournamentName']}, Fecha:
{tournament['tournamentDate']}, Lugar:
{tournament['tournamentLocation']}").pack()

def update_teams(self):
    self.update_teams_window = tk.Toplevel(self.master)
    self.update_teams_window.title("Actualizar Equipos")

    tk.Label(self.update_teams_window, text="Selecciona el equipo a
actualizar:").pack()

    team_names = [team['teamName'] for team in teams_collection.find()]

    self.team_var = tk.StringVar(self.update_teams_window)
    self.team_var.set(team_names[0]) # Setear el primer equipo por defecto
    self.team_menu = tk.OptionMenu(self.update_teams_window, self.team_var,
*team_names)
    self.team_menu.pack()

    tk.Label(self.update_teams_window, text="Nuevo Nivel:").pack()
    self.new_team_level_var = tk.StringVar(self.update_teams_window)
    self.new_team_level_var.set("Rookie")
    self.new_team_level_menu = tk.OptionMenu(self.update_teams_window,
self.new_team_level_var, "Rookie",
"Beginner", "Intermediate",
"Expert")
    self.new_team_level_menu.pack()

```

```

        confirm_button = tk.Button(self.update_teams_window, text="Confirmar",
command=self.confirm_team_update)
        confirm_button.pack()

    def confirm_team_update(self):
        team_name = self.team_var.get()
        new_team_level = self.new_team_level_var.get()

        result = teams_collection.update_one({"teamName": team_name}, {"$set":
{"teamLevel": new_team_level}})
        if result.modified_count > 0:
            messagebox.showinfo("Éxito", f"Nivel del equipo {team_name}
actualizado exitosamente.")
        else:
            messagebox.showerror("Error", f"No se encontró el equipo
{team_name}.")

        self.update_teams_window.destroy()

    def update_players(self):
        self.update_players_window = tk.Toplevel(self.master)
        self.update_players_window.title("Actualizar Jugadores")

        tk.Label(self.update_players_window, text="Selecciona el jugador a
actualizar:").pack()

        player_names = [player['playerName'] for player in
players_collection.find()]

        self.player_var = tk.StringVar(self.update_players_window)
        self.player_var.set(player_names[0]) # Setear el primer jugador por
defecto
        self.player_menu = tk.OptionMenu(self.update_players_window,
self.player_var, *player_names)
        self.player_menu.pack()

        tk.Label(self.update_players_window, text="Nuevo Nivel:").pack()
        self.new_player_level_var = tk.StringVar(self.update_players_window)
        self.new_player_level_var.set("Rookie")
        self.new_player_level_menu = tk.OptionMenu(self.update_players_window,
self.new_player_level_var, "Rookie",
                                                    "Beginner", "Intermediate",
"Expert")
        self.new_player_level_menu.pack()

        tk.Label(self.update_players_window, text="Nueva Edad:").pack()
        self.new_player_age_entry = tk.Entry(self.update_players_window)
        self.new_player_age_entry.pack()

```

```

        tk.Label(self.update_players_window, text="Nuevo Equipo:").pack()
        team_names = ["Ninguno"] + [team['teamName'] for team in
teams_collection.find()]
        self.new_player_team_var = tk.StringVar(self.update_players_window)
        self.new_player_team_var.set("Ninguno")
        self.new_player_team_menu = tk.OptionMenu(self.update_players_window,
self.new_player_team_var, *team_names)
        self.new_player_team_menu.pack()

        tk.Label(self.update_players_window, text="Nuevo Género:").pack()
        self.new_player_sex_var = tk.StringVar(self.update_players_window)
        self.new_player_sex_var.set("M")
        self.new_player_sex_menu = tk.OptionMenu(self.update_players_window,
self.new_player_sex_var, "M", "F", "Other")
        self.new_player_sex_menu.pack()

        confirm_button = tk.Button(self.update_players_window, text="Confirmar",
command=self.confirm_player_update)
        confirm_button.pack()

    def confirm_player_update(self):
        player_name = self.player_var.get()
        new_player_level = self.new_player_level_var.get()
        new_player_age = int(self.new_player_age_entry.get())
        new_player_team = self.new_player_team_var.get()
        new_player_sex = self.new_player_sex_var.get()

        update_data = {
            "$set": {
                "playerLevel": new_player_level,
                "playerAge": new_player_age,
                "teamName": new_player_team,
                "playerSex": new_player_sex
            }
        }

        result = players_collection.update_one({"playerName": player_name},
update_data)
        if result.modified_count > 0:
            messagebox.showinfo("Éxito", f"Detalles del jugador {player_name}
actualizados exitosamente.")
        else:
            messagebox.showerror("Error", f"No se encontró el jugador
{player_name}.")

        self.update_players_window.destroy()

root = tk.Tk()

```

```
app = TorneoApp(root)
root.mainloop()
```