PROYECTO PYTHON

ALUMNO: Anderson Olivos Gamarra
CURSO: 1° DAW
MÓDULO: Programación

ÍNDICE

1. Hito 1 - Fichero RankingPadel.py

a. Importación librerías

Importamos las librerías necesarias para realizar el web scraping:

- Requests para las peticiones https
- bs4 para la obtención del código y filtrar la búsqueda de elementos
- re para patrones de búsqueda

```
import requests
from bs4 import BeautifulSoup
import re
```

b. Función obtener_soup_pagina(url)

Esta función realiza la petición a la página web y obtiene su contenido y devuelve el soup directamente.

```
#Función para obtener la 'sopa' de una página

def obtener_soup_pagina(url):  2 usages  new *
    respuesta = requests.get(url).content
    soup = BeautifulSoup(respuesta, features: 'html.parser')
    return soup
```

c. Función web_scraping(enlace_pagina, numero_maximo_jugadores)

Esta función recibe 2 parámetros:

- enlace_pagina: página a la que vamos a realizar el web scraping.
- numero_maximo_jugadores: el nº máximos de jugadores que devolverá.

```
#Función principal para realizar Web Scraping

def web_scraping(enlace_pagina, numero_maximo_jugadores):  2 usages  new *
```

i. Obtener 'sopa' página de todos los jugadores

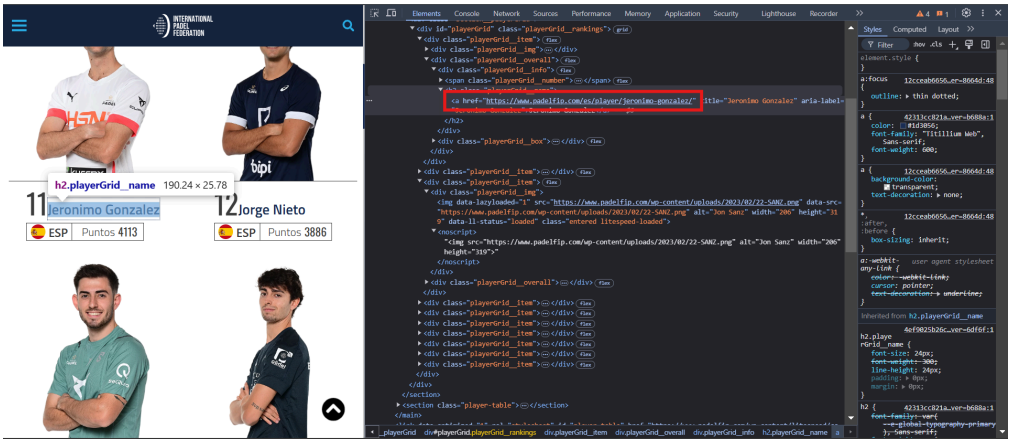
Obtenemos la sopa de la página de todos los jugadores.

```
soup_pagina = obtener_soup_pagina(enlace_pagina)
```

ii. Obtener enlaces de los jugadores

Realizamos búsqueda de todos los enlaces que el href empiecen de la forma 'https://www.padelfip.com/player/'. Además creo la lista jugadores donde meteremos cada uno de los diccionarios de los jugadores. También inicializo la variable max = 0 para luego realizar salir del bucle en el caso de que se llegue al máximo de jugadores.

```
enlaces = soup_pagina.find_all( name: 'a', href= re.compile("https://www.padelfip.com/player/"))

#Creamos una lista para almacenar los datos de los jugadores

jugadores = []

max = 0
```



1. Condición máximo jugadores

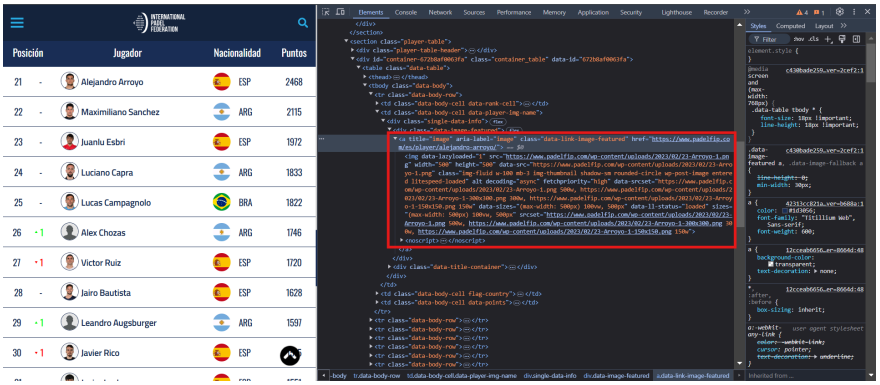
Si max es igual al número máximo de jugadores pues sale del bucle, sino suma 1 a max.

```
for enlace in enlaces:

    #Se almacenarán solo 20 jugadores

    if max==numero_maximo_jugadores: break
    else: max+=1
```

2. Condición enlaces no repetidos

En mi caso, tenía enlaces repetidos pero me fijé que los que quería no tenían una etiqueta img dentro por lo que si aplico este filtro no realizará la función para enlaces no deseados.

```
if enlace.find('img') is None:
```



3. Obtener 'sopa' del jugador

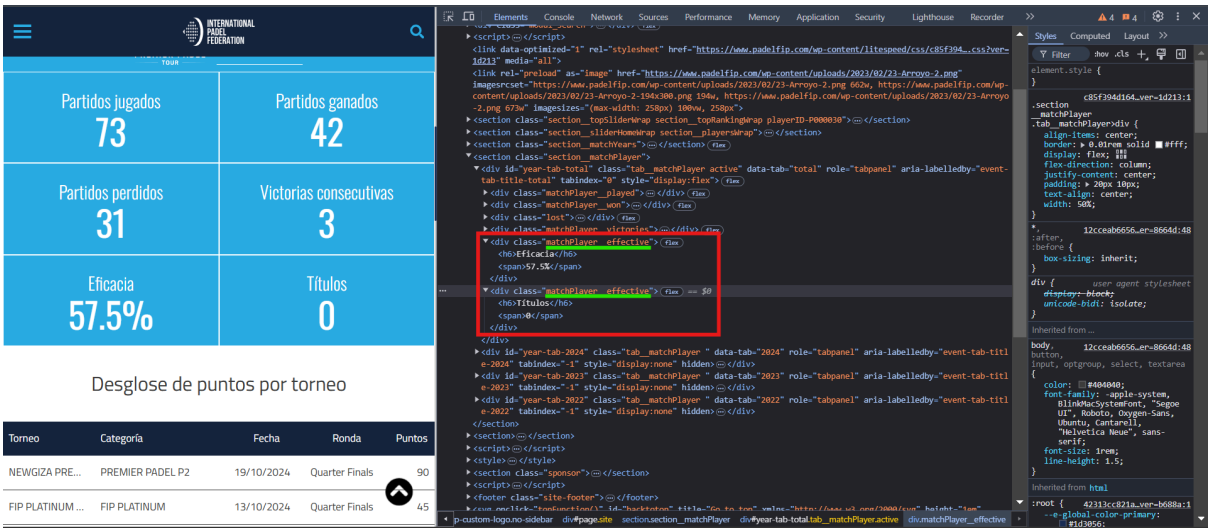
Obtenemos la sopa de la página del jugador.

```
soup_jugador = obtener_soup_pagina(enlace['href'])
```

4. Obtener efectividad y título del jugador

En mi caso, el dato Efectividad y el dato Titulos tenían la misma clase y además se repetía en la página varias veces, por lo que guardo en una lista todos los resultados.

```
#Obtener efectividad y titulos ganados del jugador

efectividades_soup = soup_jugador.find_all( name: 'div', class_='matchPlayer__effective')

efectividades = []

for efectividad in efectividades_soup:
    efectividades.append(efectividad.find('span').text)
```

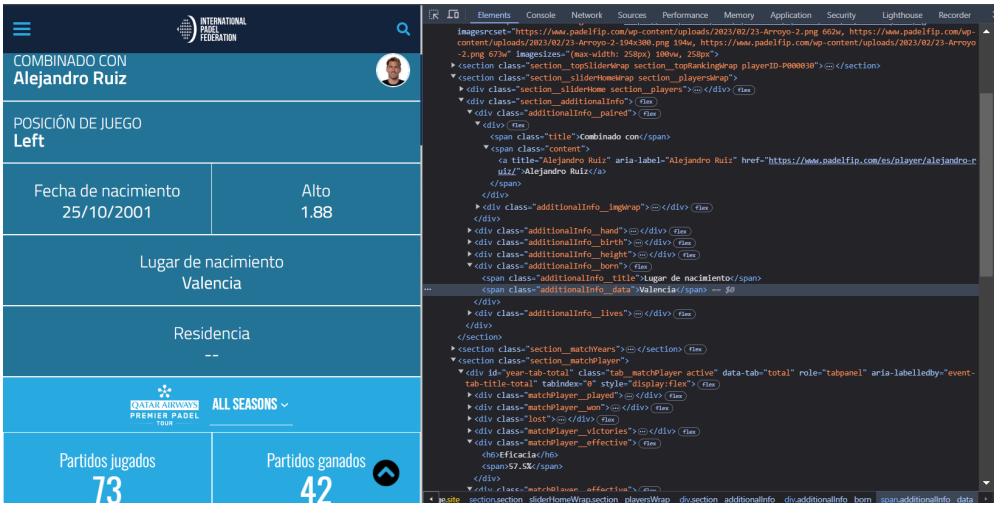


5. Creación del diccionario

Para optimizar el código y ahorrarme algunas líneas de código, decidí poner el valor de las claves directamente del resultado de las búsquedas.

```
#Creamos un diccionario con todos los datos del jugador

diccionario = {'Nombre': soup_jugador.find('h2',
class_='slider__name player__name').text,
               'Numero': int(soup_jugador.find('span',
class_='slider__number player__number').find(string=True,
recursive=False)),
               'Pais': soup_jugador.find('p',
class_='slider__country player__country').text,
               'Puntos': float(soup_jugador.find('span',
class_='slider__pointTNumber player__pointTNumber').text),
               'Posicion': soup_jugador.find('div',
class_='additionalInfo__hand').find('span',
class_='content').text,
               'Pareja': soup_jugador.find('div',
class_='additionalInfo__paired').find('a').text,
               'Anyo_nacimiento': soup_jugador.find('div',
class_='additionalInfo__birth').find('span',
class_='additionalInfo__data').text,
```

```
                'Altura': float(soup_jugador.find('div',
class_='additionalInfo__height').find('span',
class_='additionalInfo__data').text),
                'Nacido en': soup_jugador.find('div',
class_='additionalInfo__born').find('span',
class_='additionalInfo__data').text,
                'Vive en': soup_jugador.find('div',
class_='additionalInfo__lives').find('span',
class_='additionalInfo__data').text,
                'Partidos jugados':
int(soup_jugador.find('div',
class_='matchPlayer__played').find('span').text),
                'Partidos ganados':
int(soup_jugador.find('div',
class_='matchPlayer__won').find('span').text),
                'Partidos perdidos':
int(soup_jugador.find('div',
class_='lost').find('span').text),
                'Victorias consecutivas':
int(soup_jugador.find('div',
class_='matchPlayer__victories').find('span').text),
                'Efectividad': efectividades[0],
                'Titulos': int(efectividades[1]),
                'Foto jugador': soup_jugador.find('img',
class_='attachment-258x400')['data-src']
                }
```



6. Añadir diccionario a lista de jugadores

Añadimos el diccionario creado a la lista anteriormente creada.

```
#Añadimos este diccionario a la lista de jugadores
```

```
jugadores.append(diccionario)
```

7. Devolver lista de jugadores

Devolvemos la lista con todos los jugadores y sus datos.

```
#Devolvemos la lista con todos los datos de los jugadores

return jugadores
```

d. Lista de jugadores y jugadoras

Realizamos web scraping tanto de la página de ranking masculino como el femenino y almacenamos los datos en listas (jugadores_masculinos y jugadores_femeninos).

```
jugadores_masculinos =
web_scraping('https://www.padelfip.com/ranking-male/', 20)
jugadores_femeninos =
web_scraping('https://www.padelfip.com/ranking-female/', 20)
```

2. Hito 1 - Fichero convertirXLSX.py

a. Importación funciones y librerías

Las librerías que necesitaremos para realizar la conversión a XLSX son: csv y pandas. También necesitaremos importar la función web_scraping del fichero creado en RankingPadel.

```
import csv
from RankingPadel import web_scraping
import pandas as pd
```

b. Función convertir_a_xlsx

Esta función recibe 2 parámetros:

- nombre_fichero: El nombre del fichero que se va a crear.
- jugadores: el array con los diccionarios de los jugadores.

i. Conversión a CSV

En esta parte de la función realizaremos primero el almacenamiento de los datos en un fichero CSV. Para ello primero tendremos que abrir el fichero con la función open(nombre_del_fichero, modo, codificación, nueva_linea).

Luego especificamos dónde vamos a realizar la escritura de los datos con la función csv.writer(nombre_del_fichero).

A continuación, escribimos primero las claves, cogemos el primer diccionario del array y obtenemos sus claves en forma de lista. Luego, escribimos los valores del diccionario de cada uno de los jugadores.

Una vez finalizada la escritura, cerramos el fichero con la función close().

```
#Conversión a CSV

csv_file = open(f'{nombre_fichero}.csv', 'w', encoding='utf-8', newline='')
writer = csv.writer(csv_file)
writer.writerow(list(jugadores[0].keys()))
for jugador in jugadores:
    writer.writerow(jugador.values())
csv_file.close()
```

ii. Conversión a XLSX

Crearemos a continuación un fichero .xlsx a partir del fichero .csv creado en el apartado anterior.

Con el uso de la librería pandas especificamos el fichero csv que vamos a leer con la función pd.read_csv(nombre_del_fichero).

Luego simplemente lo transformamos a un fichero excel con extensión .xlsx con la función to_excel(nombre_del_fichero.xlsx, index=False). El index=False lo utilizamos para que no aparezca la enumeración.

```
#Conversión a XLSX

df = pd.read_csv(f'{nombre_fichero}.csv')

df.to_excel(f'{nombre_fichero}.xlsx', index=False)
```

c. Obtención jugadores y jugadoras

Obtenemos los jugadores masculinos y femeninos aplicando la función de web_scraping. En mi caso, quería almacenar 20 de cada uno.

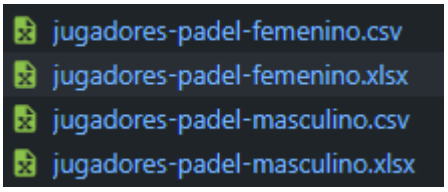
```
jugadores_masculinos = web_scraping('https://www.padelfip.com/ranking-male/', 20)
jugadores_femeninos = web_scraping('https://www.padelfip.com/ranking-female/', 20)
```

d. Llamada a la función convertir_a_xlsx

Llamamos a la función creada para que cree los ficheros xlsx.

```
convertir_a_xlsx('jugadores-padel-masculino',jugadores_masculinos)
convertir_a_xlsx('jugadores-padel-femenino',jugadores_femeninos)
```

Tendremos a continuación, todos los jugadores en un ficheros csv y xlsx:



En mi caso, he instalado un extensión de VSCode para visualizar la tabla, quería algo tal que así:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Nombre	Nur	Pais	Puntos	Posicion	Pareja	Anyo_nacimien	Altur	Nacido en	Vive	Part	Partid	Partid	Victo	Efectiv	Titu	Foto jugador
2	Ariana Sanchez Fallada	1	ESP	14,521	Left	Paula Josemaria	19/07/1997	1.65	Reus	--	85	74	11	11	87.1%	9	https://www.padelfip.com/wp-content
3	Paula Josemaria Martin	1	ESP	14,521	Right	Ariana Sanchez I	31/10/1996	1.60	Moraleja	--	85	74	11	11	87.1%	9	https://www.padelfip.com/wp-content
4	Gemma Triay Pons	3	ESP	10,108	Left	Claudia Fernand	28/06/1992	1.73	Mahón	--	86	69	17	12	80.2%	5	https://www.padelfip.com/wp-content
5	Claudia Fernandez Sanchez	4	ESP	8,975	Right	Gemma Triay Po	28/02/2006	1.64	Madrid	--	77	59	18	12	76.6%	4	https://www.padelfip.com/wp-content
6	Delfina Brea Senesi	5	ARG	8,967	Right	Beatriz Gonzalez	05/12/1999	1.70	Buenos Aires	--	73	60	13	19	82.2%	7	https://www.padelfip.com/wp-content
7	Beatriz Gonzalez Fernandez	6	ESP	7,152	Left	Delfina Brea Sen	23/11/2001	1.74	Málaga	--	60	51	9	19	85.0%	7	https://www.padelfip.com/wp-content
8	Marta Ortega Gallego	7	ESP	6,249	Right	Sofia Araujo	14/02/1997	1.70	Madrid	--	78	58	20	8	74.4%	3	https://www.padelfip.com/wp-content
9	Jessica Castello Lopez	8	ESP	5,554	Left	Alejandra Salaza	29/10/1997	1.69	Villena	--	66	46	20	8	69.7%	1	https://www.padelfip.com/wp-content
10	Sofia Araujo	9	POR	5,474	Left	Marta Ortega Gá	30/11/1994	1.72	Lisboa	--	74	53	21	4	71.6%	2	https://www.padelfip.com/wp-content
11	Lucia Sainz Pelegri	10	ESP	5,192	Right	Patricia Llaguno	05/10/1984	1.73	Barcelona	--	72	49	23	4	68.1%	0	https://www.padelfip.com/wp-content
12	Patricia Llaguno Zielinski	11	ESP	5,003	Left	Lucia Sainz Pele	25/02/1985	1.67	Cartagena	--	67	45	22	4	67.2%	0	https://www.padelfip.com/wp-content
13	Claudia Jensen Sirvent	12	ARG	4,985	Right	Tamara Icardo A	21/07/2005	1.58	Madrid	--	62	41	21	8	66.1%	1	https://www.padelfip.com/wp-content
14	Alejandra Salazar Bengoechea	13	ESP	4,861	Right	Jessica Castello I	31/12/1985	1.68	Madrid	--	53	35	18	4	66.0%	0	https://www.padelfip.com/wp-content
15	Andrea Ustero Prieto	14	ESP	4,516	Right	Alejandra Alonsc	12/05/2007	1.70	Barcelona	--	54	35	19	4	64.8%	0	https://www.padelfip.com/wp-content
16	Veronica Virseda	15	ESP	4,441	Left	Aranzazu Osoro	01/07/1992	1.70	Toledo	--	65	42	23	3	64.6%	0	https://www.padelfip.com/wp-content
17	Aranzazu Osoro Ulrich	16	ARG	3,538	Right	Veronica Virseda	03/06/1996	1.66	Parana	--	56	34	22	3	60.7%	0	https://www.padelfip.com/wp-content
18	María Virginia Riera	17	ARG	3,529	Left	Carmen Goenag	27/09/1988	1.65	Resistencia	--	56	35	21	4	62.5%	0	https://www.padelfip.com/wp-content
19	Tamara Icardo Alcorisa	18	ESP	3,419	Right	Claudia Jensen S	10/10/1995	1.78	Valencia	--	35	23	12	3	65.7%	0	https://www.padelfip.com/wp-content
20	Alejandra Alonso De Villa	19	ESP	3,255	Left	Andrea Ustero P	29/05/2006	1.69	Valladolid	--	46	29	17	4	63.0%	0	https://www.padelfip.com/wp-content
21	Marina Guinart España	20	ESP	1,988	Right	Victoria Iglesias	30/06/1996	1.78	Castellar del Vall	--	42	20	22	2	47.6%	0	https://www.padelfip.com/wp-content

3. Hito 2 - Fichero metodos.py.

Para este hito he cambiado el nombre del fichero convertirXLSX.py y le he puesto el nombre de metodos.py que incluirá tanto la conversión a xlsx como los métodos de este nuevo hito.

a. Importación de librería

Para realizar esta parte del proyecto necesitaremos importar la librería mysq en nuestro fichero. Lo renombramos como 'conector' para que sea más fácil a la hora de hacer la llamada a la función.

```
import mysql.connector as conector
```

b. Método conexión a base de datos

Creamos una función para realizar la conexión a la base de datos. Introducimos los parámetros necesarios para la conexión:

```
#FUNCION: CONEXIÓN A LA BASE DE DATOS
def conectar_bbdd():  4 usages

    parametros_conexion = {
        'user': 'root',
        'password': 'usuario',
        'host': 'localhost',
        'database': 'padel',
        'port': 3306,
        'autocommit': True
    }

    return conector.connect(**parametros_conexion)
```

Retornamos el valor de la conexión.

c. Método inserción de datos a partir del web scraping

Para todos los métodos tendremos que realizar lo siguiente: conexión a la bbdd, abrir cursor, crear sentencia, ejecutar sentencia y cerrar conexión. Lo explicaré solamente en este apartado para no ser repetitivo.

i. Obtención de lista de diccionarios de jugadores mediante web_scraping

Guardamos los datos de los jugadores y jugadoras en variables distintas

```
jugadores_masculinos = web_scraping( enlace_pagina: 'https://www.padelfip.com/ranking-male/', numero_maximo_jugadores: 20)
jugadores_femeninos = web_scraping( enlace_pagina: 'https://www.padelfip.com/ranking-female/', numero_maximo_jugadores: 20)
```

ii. Realizar la conexión a la bbdd

Hacemos llamada a la función creada para conectarnos a nuestra base de datos.

```
#Abrimos conexion con la bbdd
conexion = conectar_bbdd()
```

iii. Abrir cursor

Abrimos un cursor:

```
#Abrimos cursor
cursor = conexion.cursor()
```

iv. Sentencia de INSERT

Creamos la sentencia de inserción con todos los valores que vamos a insertar.

```
#Sentencia INSERT
script_insercion = "insert into jugador (nombre,numero_ranking, sexo, pais, puntos, posicion, pareja, anyo_nacimiento, altura, nacido_en, reside_en, partidos_jugados, partidos_ganados,
```

v. Ejecutar la inserción

En mi caso, al tener jugadores tanto femeninos como masculinos, tengo que ejecutar la inserción dos veces:

```
#Cada pala se convierte en un script de inserción
for jugador in jugadores_masculinos:
    cursor.execute(script_insercion, params: (jugador['Nombre'], jugador['Numero_ranking'],
    'M', jugador['Pais'], jugador['Puntos'], jugador['Posicion'], jugador['Pareja'],
    jugador['Anyo_nacimiento'], jugador['Altura'], jugador['Nacido en'], jugador['Reside en'],
    jugador['Partidos jugados'], jugador['Partidos ganados'], jugador['Partidos perdidos'],
    jugador['Victorias consecutivas'], jugador['Efectividad'], jugador['Titulos'], jugador['Foto jugador']))

for jugador in jugadores_femeninos:
    cursor.execute(script_insercion, params: (jugador['Nombre'], jugador['Numero_ranking'],
    'F', jugador['Pais'], jugador['Puntos'], jugador['Posicion'], jugador['Pareja'],
    jugador['Anyo_nacimiento'], jugador['Altura'], jugador['Nacido en'],
    jugador['Reside en'], jugador['Partidos jugados'], jugador['Partidos ganados'],
    jugador['Partidos perdidos'], jugador['Victorias consecutivas'],
    jugador['Efectividad'], jugador['Titulos'], jugador['Foto jugador']))
```

vi. Cerrar conexión

```
#Cerramos la conexion con la base de datos
conexion.close()
```

d. Método de recuperación de base de datos de datos

La diferencia con el apartado anterior es que en este método vamos a recuperar los datos en vez de insertarlos. Realizamos la conexión a la base de datos, iniciamos el cursor y especificamos que queremos los datos en forma de diccionario con el argumento ‘dicctionary=True’. Creamos una lista vacía llamada lista_jugadores donde almacenaremos todos nuestros jugadores. Ejecutamos la consulta y guardamos el resultado obtenido en la lista de lista_jugadores y retornamos esta lista.

```python
#FUNCION: Consulta todos los datos de la bbdd
def consultar_datos():

    #Inicamos conexión
    conexion = conectar_bbdd()

    #Iniciamos cursor y especificamos que queremos los datos en forma de diccionario
    cursor = conexion.cursor(dictionary=True)

    #Lista de jugadores
    lista_jugadores = []

    #Script de consulta
    script_consulta = "SELECT * FROM jugador"

    #Ejecutar la consulta
    cursor.execute(script_consulta)

    #Para obtener los datos de la consulta ejecutada anterior
    lista_jugadores = cursor.fetchall()

    #Cerramos conexión
    conexion.close()

    return lista_jugadores
```

e. Método de inserción de registro nuevo a partir de diccionario

Igual que en el apartado anterior, en este caso, nuestra función recibe un el parámetro id que será el id del jugador que queremos borrar. Lo único que cambia en este método es el script, que en este caso es un DELETE FROM jugador WHERE id={id}

```python
#FUNCION: Elimina jugador por id
def eliminar_por_id(id):

    # Inicamos conexión
    conexion = conectar_bbdd()

    # Iniciamos cursor y especificamos que queremos los datos en forma de diccionario
    cursor = conexion.cursor()

    # Script de consulta
    script_eliminar = "DELETE FROM jugador WHERE id=" + str(id)

    # Ejecutar la consulta
    cursor.execute(script_eliminar)

    # Cerramos conexión
    conexion.close()
```

f. Método de eliminación de registro a través de id

Este método es muy parecido al método de volcado de datos, la diferencia es que en vez de volcar todos los jugadores, solo vamos a insertar uno. Por lo que el código es muy parecido. Además, esta función recibe el parámetro sexo que podrá solamente tomar el valor 'F' o 'M', en el caso de que no lo sea devolverá un error y no se ejecutará la inserción. El otro parámetro será de tipo diccionario, que contendrá los datos que queremos insertar en nuestra base de datos.

```
#FUNCION: Inserta un nuevo jugador
def insertar_registro_nuevo(jugador,sexo):

    if sexo not in ['M','F']:
        print('Error: sexo introducido incorrecto')
    else:

        # Abrimos conexion con la bbdd
        conexion = conectar_bbdd()

        # Abrimos cursor
        cursor = conexion.cursor()

        # Sentencia INSERT
        script_insercion = "insert into jugador (nombre,numero_ranking, sexo, pais, puntos, posicion, pareja, anyo_nacim

        # Cada pala se convierte en un script de inserción
        cursor.execute(script_insercion, params: (
        jugador['Nombre'], jugador['Numero_ranking'], sexo , jugador['Pais'], jugador['Puntos'], jugador['Posicion'],
        jugador['Pareja'], jugador['Anyo_nacimiento'], jugador['Altura'], jugador['Nacido en'], jugador['Reside en'],
        jugador['Partidos jugados'], jugador['Partidos ganados'], jugador['Partidos perdidos'],
        jugador['Victorias consecutivas'], jugador['Efectividad'], jugador['Titulos'], jugador['Foto jugador']))

        # Cerramos conexión
        conexion.close()
```

4. Hito 3 - Fichero app.py

a. Importación librerías

Importamos las librerías necesarias para la creación de nuestras funciones:

```
from PyQt5.QtWidgets import QMainWindow, QApplication, QTableWidgetItem
from PyQt5.QtCore import QDate
from PyQt5.uic import loadUi
from metodos import *
import re
```

b. Función es_numero_valido(valor)

Esta función recibe un parámetro, reemplaza las ',' por '.' y comprueba si es decimal o no.

c. Función obtener_valor_o_none(valor)

Como tengo columnas en mi base de datos que pueden tomar valores nulos, he creado esta función para que en el caso de que no se haya introducido ningún valor se tome por defecto el valor None.

d. Clase VentanaPrincipal(QMainWindow)

i. Llamada a las funciones cuando se hace click en los botones.

En esta parte, simplemente le he asignado a cada botón la función que tiene que realizar. Algunos simplemente cambian de ventana y otros ejecutan funciones.

```
class VentanaPrincipal(QMainWindow):  2 usages
    def __init__(self):
        super(VentanaPrincipal, self).__init__()
        loadUi( uifile: './DISEÑO/crud_design.ui', self)
        self.modificar_id_jugador_actual = None


        # CONECTAR BOTONES CON PAGINAS
        self.nav_boton_jugadores.clicked.connect(self.cargar_datos_tabla)
        self.nav_boton_anadir.clicked.connect(lambda: self.main.setCurrentWidget(self.anadir))
        self.nav_boton_modificar.clicked.connect(lambda: self.main.setCurrentWidget(self.modificar))
        self.nav_boton_eliminar.clicked.connect(lambda: self.main.setCurrentWidget(self.eliminar))


        #CONECTAR BOTON REFRESCAR A LA FUNCION DE CARGAR DATOS TABLA
        self.btn_refrescar_jugadores.clicked.connect(self.cargar_datos_tabla)
        self.btn_buscar_jugadores.clicked.connect(self.cargar_datos_tabla_filtrado)
        self.btn_anadir_anadir.clicked.connect(self.anadir_jugador)
        self.btn_buscar_eliminar.clicked.connect(self.cargar_datos_tabla_eliminar)
        self.btn_eliminar.clicked.connect(self.eliminar_jugador)
        self.btn_buscar_modificar.clicked.connect(self.cargar_datos_modificar)
        self.btn_modificar.clicked.connect(self.modificar_jugador)
```

ii. Función cargar_datos_tabla

Esta función toma todos los datos de la base de datos y lo introduce en la tabla.

```python
# CARGAR DATOS
def cargar_datos_tabla(self):  3 usages
    self.inp_buscador_jugadores.setText('')
    self.main.setCurrentWidget(self.jugadores)
    jugadores = consultar_datos()
    self.tabla_jugadores.setRowCount(len(jugadores))

    for fila, jugador in enumerate(jugadores, start=0):
        for columna, campo_jugador in enumerate(jugador.values(), start=0):
            self.tabla_jugadores.setItem(fila, columna, QTableWidgetItem(str(campo_jugador)))
    self.tabla_jugadores.resizeColumnsToContents()
```

iii. Función cargar_datos_tabla_filtrado

Esta función es muy parecida a la anterior, la diferencia es que filtra por el nombre del jugador o el nombre de la pareja.

```python
def cargar_datos_tabla_filtrado(self):  1 usage
    texto_filtro = self.inp_buscador_jugadores.text()
    if texto_filtro != "":
        jugadores_filtrado = consultar_datos_filtrado(texto_filtro)
        self.tabla_jugadores.setRowCount(len(jugadores_filtrado))

        for fila, jugador_filtrado in enumerate(jugadores_filtrado, start=0):
            for columna, campo_jugador_filtrado in enumerate(jugador_filtrado.values(), start=0):
                self.tabla_jugadores.setItem(fila, columna, QTableWidgetItem(str(campo_jugador_filtrado)))
        self.tabla_jugadores.resizeColumnsToContents()
```

iv. Función cargar_datos_tabla_eliminar

Esta función carga todos los jugadores encontrados (parecido a la función de filtrado) en la tabla de eliminar.

```python
def cargar_datos_tabla_eliminar(self):  1 usage
    texto_filtro = self.inp_buscador_eliminar.text()
    if texto_filtro != "":
        jugadores_filtrado = consultar_datos_filtrado(texto_filtro)

        self.tabla_eliminar.setRowCount(len(jugadores_filtrado))
        for fila, jugador_filtrado in enumerate(jugadores_filtrado, start=0):
            for columna, campo_jugador_filtrado in enumerate(jugador_filtrado.values(), start=0):
                self.tabla_eliminar.setItem(fila, columna, QTableWidgetItem(str(campo_jugador_filtrado)))
        self.tabla_eliminar.resizeColumnsToContents()
```

v. Función anadir_jugador

Esta función recoge los datos de los inputs, realiza un control de errores (si hay campos vacíos donde los tuviera que haber o si hay algún formato incorrecto).

```python
def anadir_jugador(self):  1 usage
    anadir_nombre = self.inp_anadir_nombre.text()
    anadir_ranking = self.inp_anadir_ranking.text()
    anadir_puntos = self.inp_anadir_puntos.text()
    anadir_posicion = self.inp_anadir_posicion.text()
    anadir_pareja = self.inp_anadir_pareja.text()
    anadir_pais = self.inp_anadir_pais.text()
    anadir_partidos_jugados = self.inp_anadir_partidos_jugados.text()
    anadir_partidos_ganados = self.inp_anadir_partidos_ganados.text()
    anadir_partidos_perdidos = self.inp_anadir_partidos_perdidos.text()
    anadir_victorias_consecutivas = self.inp_anadir_victorias_consecutivas.text()
    anadir_efectividad = self.inp_anadir_efectividad.text()
    anadir_titulos = self.inp_anadir_titulos.text()
    anadir_fecha_nacimiento = self.inp_anadir_fecha_nacimiento.date().toString("dd/MM/yyyy")
    anadir_pais_nacimiento = self.inp_anadir_pais_nacimiento.text()
    anadir_lugar_residencia = self.inp_anadir_lugar_residencia.text()
    anadir_altura = self.inp_anadir_altura.text()
    anadir_url_fotografia = self.inp_anadir_url_fotografia.text()
    anadir_inp_sexo = 'F' if self.inp_anadir_sexo_f.isChecked() else 'M'

    campos_obligatorios = [anadir_nombre, anadir_ranking, anadir_puntos, anadir_posicion, anadir_pais, a
                           anadir_pais_nacimiento, anadir_inp_sexo]

    campos_numericos = [anadir_ranking, anadir_puntos, anadir_partidos_jugados, anadir_partidos_ganados,
                        anadir_victorias_consecutivas, anadir_titulos, anadir_altura]

    if any(campo.strip() == "" for campo in campos_obligatorios):
        self.p_campos_obligatorios.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
        self.p_campos_obligatorios.setText('Error: falta algun campo obligatorio por rellenar')

    else:

        self.p_campos_obligatorios.setText('')

        if any(campo != "" and es_numero_valido(campo) is False for campo in campos_numericos):
            self.p_campos_obligatorios.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
            self.p_campos_obligatorios.setText('Error: formato incorrecto.')
        else:

            anadir_ranking = obtener_valor_o_none(anadir_ranking)
```

vi. Función eliminar_jugador

Esta función coge el valor de la columna id de la fila seleccionada y ejecuta la función creada en apartados anteriores eliminar_por_id.

```python
def eliminar_jugador(self):  1 usage
    # Obtener los elementos seleccionados
    items_seleccionados = self.tabla_eliminar.selectedItems()

    if not items_seleccionados:
        self.p_errores_eliminar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
        self.p_errores_eliminar.setText('Error: seleccione un jugador.')
        return None

    else:
        self.p_errores_eliminar.setText('')

        # Obtener la fila del primer elemento seleccionado
        fila_seleccionada = items_seleccionados[0].row()

        # Obtener el valor de la primera columna (columna 0) en la fila seleccionada
        jugador_id = self.tabla_eliminar.item(fila_seleccionada, 0).text()

        try:
            eliminar_por_id(jugador_id)
            self.p_errores_eliminar.setStyleSheet("font-size: 20px; font-weight: bold; color: lightgreen;")
            self.p_errores_eliminar.setText('Jugador eliminado con éxito.')
            return True
        except Exception as e:
            print(f"Error al eliminar el jugador: {e}")
```

vii. Función cargar_datos_modificar

Esta función se encarga de buscar el jugador por id introducido en el input y rellena los inputs del formulario con los datos del jugador.

```python
def cargar_datos_modificar(self):  1 usage
    texto_filtro = self.inp_buscar_modificar.text()
    if texto_filtro != "" and texto_filtro.isnumeric():

        jugador_modificar = consultar_datos_por_id(texto_filtro)

        if len(jugador_modificar) == 0:
            self.p_error_modificar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
            self.p_error_modificar.setText('Error: el ID introducido no existe.')
        else:

            self.inp_modificar_nombre.setText(jugador_modificar[0]["nombre"])
            self.inp_modificar_ranking.setText(str(jugador_modificar[0]["numero_ranking"]))
            self.inp_modificar_puntos.setText(str(jugador_modificar[0]["puntos"]))
            self.inp_modificar_posicion.setText(jugador_modificar[0]["posicion"])
            self.inp_modificar_pareja.setText(jugador_modificar[0]["pareja"])
            self.inp_modificar_pais.setText(jugador_modificar[0]["pais"])
            self.inp_modificar_partidos_jugados.setText(str(jugador_modificar[0]["partidos_jugados"]))
            self.inp_modificar_partidos_ganados.setText(str(jugador_modificar[0]["partidos_ganados"]))
            self.inp_modificar_partidos_perdidos.setText(str(jugador_modificar[0]["partidos_perdidos"]))
            self.inp_modificar_victorias_consecutivas.setText(str(jugador_modificar[0]["victorias_consecutivas"]))
            self.inp_modificar_efectividad.setText(jugador_modificar[0]["efectividad"])
            self.inp_modificar_titulos.setText(str(jugador_modificar[0]["titulos"]))

            fecha_nacimiento = QDate.fromString(jugador_modificar[0]["fecha_nacimiento"], "dd/MM/yyyy")
            self.inp_modificar_fecha_nacimiento.setDate(fecha_nacimiento)

            self.inp_modificar_pais_nacimiento.setText(jugador_modificar[0]["pais_nacimiento"])
            self.inp_modificar_altura.setText(str(jugador_modificar[0]["altura"]))
            self.inp_modificar_lugar_residencia.setText(jugador_modificar[0]["lugar_residencia"])
            self.inp_modificar_url_fotografia.setText(jugador_modificar[0]["url_foto"])

            if jugador_modificar[0]["sexo"] == "M":
                self.inp_modificar_sexo_m.setChecked(True)
                self.inp_modificar_sexo_f.setChecked(False)
            else:
                self.inp_modificar_sexo_f.setChecked(True)
                self.inp_modificar_sexo_m.setChecked(False)

            self.p_error_modificar.setText('')

            self.modificar_id_jugador_actual = jugador_modificar[0]["id"]


    else:
        self.p_error_modificar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
        self.p_error_modificar.setText('Error: ID inválido.')
        return None
```

viii. Función modificar_jugador

Esta función es bastante parecida a la de añadir un nuevo jugador, toma todos los valores de los inputs, verifica que tengan un formato correcto y modifican el jugador con el id indicado.

```python
def modificar_jugador(self):  1 usage

    if self.modificar_id_jugador_actual is None:
        self.p_error_modificar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
        self.p_error_modificar.setText("Error: no se ha cargado ningún jugador.")
        return

    modificar_nombre = self.inp_modificar_nombre.text()
    modificar_ranking = self.inp_modificar_ranking.text()
    modificar_puntos = self.inp_modificar_puntos.text()
    modificar_posicion = self.inp_modificar_posicion.text()
    modificar_pareja = self.inp_modificar_pareja.text()
    modificar_pais = self.inp_modificar_pais.text()
    modificar_partidos_jugados = self.inp_modificar_partidos_jugados.text()
    modificar_partidos_ganados = self.inp_modificar_partidos_ganados.text()
    modificar_partidos_perdidos = self.inp_modificar_partidos_perdidos.text()
    modificar_victorias_consecutivas = self.inp_modificar_victorias_consecutivas.text()
    modificar_efectividad = self.inp_modificar_efectividad.text()
    modificar_titulos = self.inp_modificar_titulos.text()
    modificar_fecha_nacimiento = self.inp_modificar_fecha_nacimiento.date().toString("dd/MM/yyyy")
    modificar_pais_nacimiento = self.inp_modificar_pais_nacimiento.text()
    modificar_lugar_residencia = self.inp_modificar_lugar_residencia.text()
    modificar_altura = self.inp_modificar_altura.text()
    modificar_url_fotografia = self.inp_modificar_url_fotografia.text()
    modificar_inp_sexo = 'F' if self.inp_modificar_sexo_f.isChecked() else 'M'

    campos_obligatorios = [modificar_nombre, modificar_ranking, modificar_puntos, modificar_posicion, modificar_pais,
                           modificar_fecha_nacimiento,
                           modificar_pais_nacimiento, modificar_inp_sexo]

    campos_numericos = [modificar_ranking, modificar_puntos, modificar_partidos_jugados, modificar_partidos_ganados,
                        modificar_partidos_perdidos,
                        modificar_victorias_consecutivas, modificar_titulos, modificar_altura]

    if any(campo.strip() == "" for campo in campos_obligatorios):
        self.p_error_modificar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
        self.p_error_modificar.setText('Error: falta algun campo obligatorio por rellenar')

    else:

        self.p_error_modificar.setText('')

        if any(campo != "" and es_numero_valido(campo) is False for campo in campos_numericos):
            self.p_error_modificar.setStyleSheet("font-size: 15px; font-weight: bold; color: red;")
            self.p_error_modificar.setText('Error: formato incorrecto.')
        else:

            modificar_ranking = obtener_valor_o_none(modificar_ranking)
            modificar_puntos = obtener_valor_o_none(modificar_puntos)
            modificar_partidos_jugados = obtener_valor_o_none(modificar_partidos_jugados)
            modificar_partidos_ganados = obtener_valor_o_none(modificar_partidos_ganados)
            modificar_partidos_perdidos = obtener_valor_o_none(modificar_partidos_perdidos)
            modificar_victorias_consecutivas = obtener_valor_o_none(modificar_victorias_consecutivas)
            modificar_titulos = obtener_valor_o_none(modificar_titulos)
            modificar_altura = obtener_valor_o_none(modificar_altura.replace(',', '.'))
```

e. Inicialización del programa

Iniciamos nuestro programa con las siguientes líneas:

```python
if __name__ == "__main__":
    import sys
    app = QApplication(sys.argv)
    ventana_principal = VentanaPrincipal()
    ventana_principal.show()
    ventana_principal.cargar_datos_tabla()
    sys.exit(app.exec_())
```