

Pia\_G2

# PODCAST

## PREVIA

### CARLOS

Bueno, listos para nuestro podcast grupo 2.

-Hay que decir que al principio escoger la problemática y la estructura de la misma fue algo muy complicado ya que no teníamos un punto de inicio, así que decidimos hacer una lluvia e ideas (lo interrumpe alguien)

### EMILY

-Fue ahí cuando se nos ocurrió, que tal si hacemos un programa que nos filtre a todos los países que empiecen con A y hacer un análisis de información donde podríamos calcular cosas como pueden ser: su media, mediana, moda, varianza y desviación estandar.(interrupción)

### SAMANTHA

-Aunque después de eso identificamos dos problemas, la primera, el usuario se confundiría al ver toda esta información como una tabla y la segunda, de dónde sacamos toda esa información?(interrupción)

### ANA

-- Fue como si hubiéramos estado de nuevo en el punto de inicio ya que teníamos un nuevo problema, lo recuerdan? Dábamos demasiadas alternativas de cómo solucionar esos problemas...(interrupción)

### EMILY

-Si que lo recuerdo, fue demasiado complicado...

### CARLOS

-El lado positivo es que pudimos efectuar soluciones al respecto, como que a la vez que le demos una tabla al usuario se le muestre una gráfica por 10 segundos !!

### ANA

-Y los datos que se usarán para poder usarlos, se saquen de una API

Pia\_G2

# PODCAST

## **PACHUCA**

-API?, si los habia escuchado hablar de eso pero siendo sincer@ jamas entendi a que se referian con eso.

## **SAMANTHA**

-Porque no lo dijiste antes?, mira una API es como si fuera un sitio de preguntas, donde cualquier duda que tengas te la responde, a esto se le llaman solicitudes y con estas tenemos la información.

## **ANA**

-Hey aunque no te olvides que las respuestas que te dan estan en JSON, asi que las respuestas que nos dan se guardan en ese tipo de archivo y python los vuelve diccionarios. asi tenemos un mejor control de la información.

## **EMILY**

-Eso fue lo bonito, pero preguntenme como le tuve que hacer para sacar todos esos calculos.

## **CARLOS**

-Eso es verdad, que tuviste que hacer entonces? recuerdo haber estado distraido tanto por la API y el JASON que una vez hechos nisiquiera me preocupe por lo demas

## **PACHUCA**

-Aunque no lo creas, en Python tambien se pueden usar las matematicas de una forma mas avanzada (NOOOOOOOOO) asi es, con fracciones y toda la cosa, pero para la media,mediana y moda use algunos modulos, lo cual, tengo que ser sincer@ fue algo sencillo.

## **SAMANTHA**

-Es enserio?, yo no sabia que habian modulos....todo lo eh hecho en base a operaciones.

## **PACHUCA**

Pero para mi, la parte mas complicada fue hacer la grafica, o tu que opinas azul

-Yo tambien tengo esa duda con respecto a la grafica, como siquiera se pudo graficar?, parecia como si fuera en excel.

## **ANA**

-Pues porque fue en excel (QUEEEEEEE)

# COMIENZO

# COMIENZO

## [INTRO - MÚSICA SUAVE DE FONDO]

- CARLOS:

¡HOLA, HOLA! BIENVENIDOS A UN NUEVO EPISODIO DEL PODCAST ESTUDIANTIL. HOY LES TRAEMOS ALGO DIFERENTE: UN PROYECTO QUE HICIMOS EN EQUIPO, Y LA NETA, NOS QUEDÓ BASTANTE BIEN. ¿VERDAD, EQUIPO?

- EVELYN:

¡SÍ! FUE UN PROYECTO DONDE USAMOS PYTHON, APIS, ANÁLISIS ESTADÍSTICO Y HASTA VISUALIZACIÓN DE DATOS.

- RAÚL:

Y TODO BASADO EN INFORMACIÓN REAL SOBRE PAÍSES DE TODO EL MUNDO. ¡UNA LOCURA LO QUE DESCUBRIMOS!

- SAMANTHA:

Y AUNQUE HUBO MOMENTOS DE ESTRÉS (RISAS), APRENDIMOS MUCHÍSIMO. ASÍ QUE ACOMPÁÑENNOS MIENTRAS LES CONTAMOS PASO A PASO CÓMO LO HICIMOS.

## [SECCIÓN 1: CONECTAR CON LA API]

- SAMANTHA:

"YO EMPECÉ CON LA PRIMERA FUNCIÓN: CONECTAR CON LA API DE REST COUNTRIES. USAMOS REQUESTS PARA HACER LA CONSULTA, Y LA VERDAD ES QUE ME EMOCIONÉ CUANDO VI LA RESPUESTA COMPLETA CON DATOS DE TODOS LOS PAÍSES. ¡FUE COMO ABRIR UNA CAJA DE SORPRESAS! CADA PAÍS TENÍA INFORMACIÓN DETALLADA: NOMBRES, CAPITALES, POBLACIONES, BANDERAS... TODO LO QUE NECESITÁBAMOS."

- CARLOS:

¡"SÍ! RECUERDO PERFECTAMENTE CUANDO DIJESTE: '¡YA ME REGRESÓ UN JSON GIGANTE!'. FUE IMPRESIONANTE VER CÓMO LA API NOS DEVOLVÍA DATOS TAN ESTRUCTURADOS Y COMPLETOS. ¡ERA COMO TENER ACCESO A UN MUNDO ENTERO EN FORMATO DIGITAL!"

EVELYN:

"Y CLARO, DESPUÉS TODOS QUERÍAMOS EXPLORAR QUÉ TANTA INFO TRAÍA. NOMBRES, CAPITALES, POBLACIONES, BANDERAS... INCLUSO IDIOMAS Y MONEDAS. ERA COMO TENER UN ATLAS INTERACTIVO EN NUESTRAS MANOS. ¡TODO ESTABA ALLÍ!".

- SAMANTHA:

"PERO TAMBIÉN FUE CLAVE VALIDAR QUE LA API RESPONDIERA CORRECTAMENTE. ES DECIR, ASEGUARNOS DE QUE EL CÓDIGO DE ESTADO FUERA 200. PORQUE SI NO, NADA FUNCIONABA. SI LA CONEXIÓN FALLABA O LA RESPUESTA NO ERA VÁLIDA, TODO SE DESMORONABA. ASÍ QUE TUVIMOS QUE SER MUY METICULOSOS AL MANEJAR ERRORES."

- RAÚL:

"¡LO BUENO ES QUE SAM FUE PACIENTE CON ESO! A VECES LA API TARDABA MUCHO O DABA ERRORES, Y HABÍA QUE PROBAR VARIAS VECES. PERO GRACIAS A SU PACIENCIA Y A LAS EXCEPCIONES BIEN MANEJADAS EN EL CÓDIGO, PUDIMOS SEGUIR ADELANTE SIN CAER EN PÁNICO. ¡ESO FUE CLAVE PARA MANTENERNOS ENFOCADOS!"

## **[SECCIÓN 2: ESTRUCTURA DE DATOS]**

- CARLOS:

"LUEGO TOCÓ ORGANIZAR LA INFORMACIÓN, Y AUNQUE ESTA PARTE LA CHECO YO, AÚN ESTOY AFINANDO LOS DETALLES. PERO LA IDEA FUE USAR DICCIONARIOS Y LISTAS, ¿SE ACUERDAN?"

- EVELYN:

"SÍ, CADA PAÍS LO PUSIMOS COMO UN DICCIONARIO CON CLAVES COMO 'NOMBRE', 'POBLACIÓN', 'ÁREA'... ASÍ PODÍAMOS ACCEDER A TODO MÁS FÁCILMENTE."

- RAÚL:

"Y ESO NOS AYUDÓ MUCHO DESPUÉS PARA HACER CÁLCULOS O FILTRAR DATOS. ¡TODO ESTABA SÚPER CLARO!"

- SAMANTHA:

"LO COOL FUE QUE YA QUE TENÍAMOS ESA ESTRUCTURA, PODÍAMOS RECORRER LA LISTA Y HACER PRÁCTICAMENTE CUALQUIER COSA CON LA INFO."

- ANA PAMELA:

"RECUERDO QUE TAMBIÉN TUVIMOS QUE CALCULAR LA DENSIDAD POBLACIONAL. FUE INTERESANTE PORQUE ALGUNOS PAÍSES NO TIENEN ÁREA REGISTRADA EN LA API. ¿CÓMO MANEJAMOS ESO?"

- CARLOS:

"EXACTO. PARA ESO USAMOS LA FUNCIÓN CALCULAR\_DENSIDAD(POBLACION, AREA). SI EL ÁREA ES CERO O NO ESTÁ DISPONIBLE, SIMPLEMENTE DEVOLVEMOS CERO PARA EVITAR ERRORES DE DIVISIÓN POR CERO. ADEMÁS, REDONDEAMOS EL RESULTADO A DOS DECIMALES PARA MEJORAR LA LEGIBILIDAD."

- ANA PAMELA:

"¡AHORA ENTIENDO POR QUÉ ALGUNAS ENTRADAS DE DENSIDAD SALÍAN COMO '0.00'! ERA PORQUE ESOS PAÍSES NO TENÍAN DATOS COMPLETOS. PERO AL MENOS EVITAMOS QUE EL PROGRAMA SE CAIGA."

- CARLOS:

"SÍ, EXACTO. Y ESO NOS PERMITIÓ SEGUIR ADELANTE SIN INTERRUPCIONES. LA CLAVE FUE MANEJAR BIEN LAS EXCEPCIONES Y ASEGURARNOS DE QUE TODO FUNCIONARA INCLUSO CON DATOS INCOMPLETOS."

## **[SECCIÓN 3: MANEJO DE ARCHIVOS Y EXPRESIONES RE]**

- RAÚL:

"AHORA SÍ, TODOS NOS METIMOS A GUARDAR LOS DATOS Y LIMPIARLOS. USAMOS ARCHIVOS .JSON PARA GUARDAR LOCALMENTE LA INFORMACIÓN QUE TRAÍA LA API. FUE UNA FORMA SENCILLA DE PERSISTIR LOS DATOS ESTRUCTURADOS."

- EVELYN:

"Y COMO NO TODO VENÍA BONITO, USAMOS EXPRESIONES REGULARES PARA LIMPIAR, POR EJEMPLO, TEXTOS CON SÍMBOLOS RAROS O CAMPOS VACÍOS. ESTO FUE CLAVE PARA ASEGURARNOS DE QUE LOS DATOS ESTUVIERAN LISTOS PARA ANÁLISIS Y VISUALIZACIÓN."

- CARLOS:

"RECUERDO QUE NOS PELEAMOS CON UNA Ñ MAL CODIFICADA (RISAS). PERO LO RESOLVIMOS USANDO ENSURE\_ASCII=False AL ESCRIBIR EL ARCHIVO JSON. ¡FUE UN GRAN APRENDIZAJE!"

- SAMANTHA:

"¡Y ESO FUE CLAVE! PORQUE SI NO LIMPIÁBAMOS BIEN, LUEGO LOS GRÁFICOS O LOS ANÁLISIS DABAN RESULTADOS RARÍSIMOS. LA LIMPIEZA INICIAL ES FUNDAMENTAL PARA GARANTIZAR QUE TODO FUNCIONE CORRECTAMENTE EN ETAPAS POSTERIORES."

### **[SECCIÓN 3: MANEJO DE ARCHIVOS Y EXPRESIONES RE]**

- ANA PAMELA:

"HABLANDO DE EXPRESIONES REGULARES, RECUERDO QUE TAMBIÉN IMPLEMENTAMOS LA FUNCIÓN FILTRAR\_PAISES\_CON\_REGEX. ¿CÓMO TE FUE CON ESA PARTE, RAÚL?"

- RAÚL:

"LA VERDAD, FUE BASTANTE INTERESANTE. CON RE.COMPILE() PUDIMOS HACER BÚSQUEDAS AVANZADAS COMO PAÍSES QUE COMIENZAN CON 'A' O TERMINAN EN 'LAND'. ADEMÁS, USAMOS RE.IGNORECASE PARA QUE LAS BÚSQUEDAS FUERAN INSENSIBLES A MAYÚSCULAS Y MINÚSCULAS. POR EJEMPLO, SI QUEREMOS BUSCAR PAÍSES QUE EMPIECEN CON 'A', SIMPLEMENTE PASAMOS EL PATRÓN '^A'."

- ANA PAMELA:

"¡EXACTO! Y ESO NOS PERMITIÓ PERSONALIZAR MUCHO MÁS LAS CONSULTAS. POR EJEMPLO, PODRÍAMOS FILTRAR PAÍSES CUYOS NOMBRES CONTENGAN NÚMEROS O QUE TENGAN CIERTOS CARACTERES ESPECÍFICOS. ¿TE ACUERDAS DE CÓMO MANEJAMOS ERRORES EN CASO DE PATRONES INVÁLIDOS?"

- RAÚL:

"SÍ, CAPTURAMOS EXCEPCIONES CON EXCEPT RE.ERROR AS E:. SI ALGUIEN INTRODUCE UN PATRÓN INCORRECTO, COMO [^, LE AVISAMOS INMEDIATAMENTE. ESO EVITA QUE EL PROGRAMA SE CAIGA Y PERMITE SEGUIR TRABAJANDO SIN PROBLEMAS."

- ANA PAMELA:

"¡MUY PRÁCTICO! ASÍ PODEMOS OFRECER UNA EXPERIENCIA MÁS ROBUSTA AL USUARIO FINAL. LAS EXPRESIONES REGULARES REALMENTE AMPLÍAN LAS POSIBILIDADES DE FILTRADO Y BÚSQUEDA."

## **[SECCIÓN 4: ANÁLISIS ESTADÍSTICO]**

- RAÚL:

"ESTA PARTE ME TOCÓ A MÍ Y ESTUVO BUENÍSIMA. HICIMOS MEDIA, MEDIANA, MODA, VARIANZA Y DESVIACIÓN ESTÁNDAR DE DATOS COMO POBLACIÓN O ÁREA. FUE UN RETO INTERESANTE PORQUE TENÍAMOS QUE ASEGURARNOS DE MANEJAR BIEN LOS VALORES NULOS O NO NUMÉRICOS."

- CARLOS:

"SÍ, USAMOS EL MÓDULO STATISTICS PARA LA MAYORÍA DE LOS CÁLCULOS, PERO TAMBIÉN IMPLEMENTAMOS ALGUNAS FUNCIONES A MANO PARA ENTENDER MEJOR CÓMO FUNCIONABAN LAS MÉTRICAS ESTADÍSTICAS. POR EJEMPLO, CALCULAMOS LA MEDIA MANUALMENTE PARA VER CÓMO SE DISTRIBUÍAN LOS VALORES ANTES DE USAR STATISTICS.MEAN()."

- EVELYN:

"A MÍ ME SORPRENDIÓ MUCHO LA DESVIACIÓN ESTÁNDAR. ALGUNOS PAÍSES TIENEN DATOS SÚPER ALEJADOS DEL PROMEDIO, LO QUE REALMENTE IMPACTA EN LOS RESULTADOS. ES COMO SI HUBIERA UNA 'COLA LARGA' DE PAÍSES CON POBLACIONES EXTREMADAMENTE ALTAS O BAJAS."

- SAMANTHA:

"¡EXACTO! PAÍSES COMO INDIA O CHINA TIENEN POBLACIONES TAN GRANDES QUE CAMBIAN POR COMPLETO LAS ESTADÍSTICAS GLOBALES. LA MEDIA PUEDE SER MUY ALTA DEBIDO A ESTOS VALORES EXTREMOS, MIENTRAS QUE LA MEDIANA NOS DA UNA VISIÓN MÁS REALISTA DE LO QUE OCURRE EN LA MAYOR PARTE DE LOS PAÍSES."

- RAÚL:

"FUE UN GRAN EJERCICIO PARA ENTENDER CÓMO LOS DATOS EXTREMOS INFLUYEN EN LOS RESULTADOS. POR EJEMPLO, CUANDO CALCULAMOS LA MEDIA DE POBLACIÓN, VIMOS QUE PAÍSES COMO CHINA O INDIA ELEVAN MUCHO ESE VALOR, MIENTRAS QUE LA MEDIANA NOS MUESTRA UNA IMAGEN MÁS EQUILIBRADA DE LA DISTRIBUCIÓN."

## [SECCIÓN 5: EXPORTAR A EXCEL]

- CARLOS:

"DESPUÉS DE ANALIZAR LOS DATOS Y OBTENER ESTADÍSTICAS, TOCÓ EXPORTARLOS A UN FORMATO MÁS ACCESIBLE PARA COMPARTIRLOS. RAÚL SE ENCARGÓ DE ESTA PARTE, ¿VERDAD?"

- RAÚL:

"SÍ, USAMOS PANDAS PARA CONVERTIR LA LISTA DE DICCIONARIOS EN UN DATAFRAME Y LUEGO EXPORTARLO A EXCEL CON .TO\_EXCEL(). EL ARCHIVO QUEDÓ MUY LIMPIO Y ORDENADO, ¡PERFECTO PARA REVISIÓN O PRESENTACIÓN!"

- EVELYN:

"Y ESO ES CLAVE PORQUE AHORA PODEMOS COMPARTIR ESTOS DATOS CON COLEGAS O PROFESORES QUE NO NECESARIAMENTE SABEN PYTHON. CON UN ARCHIVO EXCEL, TODO EL MUNDO PUEDE VER Y ENTENDER FÁCILMENTE LO QUE HICIMOS."

- SAMANTHA:

"ADEMÁS, NOS PERMITIÓ REVISAR VISUALMENTE TODOS LOS DATOS ANTES DE FINALIZAR EL PROYECTO. FUE ÚTIL PARA DETECTAR POSIBLES ERRORES O INCONSISTENCIAS QUE PODRÍAN HABER PASADO DESAPERCIBIDOS EN EL CÓDIGO."

## [SECCIÓN 6: VISUALIZACIÓN CON GRÁFICAS]

- SAMANTHA:

"Y CLARO, TODOS METIMOS MANO PARA HACER GRÁFICAS CON MATPLOTLIB. FUE UN PASO CLAVE PARA TRANSFORMAR DATOS ABSTRACTOS EN INFORMACIÓN VISUALMENTE COMPRENSIBLE."

- CARLOS:

"SÍ, HICIMOS UNA GRÁFICA DE BARRAS CON LOS PAÍSES MÁS POBLADOS. ¿SE ACUERDAN? MOSTRAMOS LOS TOP 10 PAÍSES POR POBLACIÓN, LO QUE FUE MUY IMPACTANTE AL VER CÓMO ALGUNOS PAÍSES COMO CHINA Y INDIA DOMINABAN LA LISTA."

- RAÚL:

"¡Y TAMBIÉN HICIMOS UNA GRÁFICA DE LÍNEAS MOSTRANDO EL ÁREA POR CONTINENTE! ESA QUEDÓ REALMENTE ESTÉTICA. USAMOS COLORES SUAVES Y ETIQUETAS CLARAS PARA QUE FUERA FÁCIL DE LEER."

- EVELYN:

"LO CHIDO FUE QUE PERSONALIZAMOS TODO: LOS COLORES, LOS TÍTULOS, LAS ETIQUETAS... INCLUSO ROTAMOS LAS ETIQUETAS DEL EJE X PARA EVITAR SOLAPAMIENTOS. ¡INCLUSO AGREGAMOS REJILLAS PARA MEJORAR LA LEGIBILIDAD!"

- CARLOS:

"SÍ, Y CREO QUE AHÍ FUE DONDE TODOS DIJIMOS 'OK, ESTO YA PARECE UN PROYECTO REAL' (RISAS). VER ESOS GRÁFICOS TAN PROFESIONALES NOS DIO MUCHA CONFIANZA. ERA COMO SI HUBIÉRAMOS PASADO DE SER PRINCIPIANTES A EXPERTOS EN VISUALIZACIÓN DE DATOS."

## **[SECCIÓN 7: INTERPRETACIÓN DE RESULTADOS]**

- **ANA PAMELA:**

"Y YA CON TODO ESO, NOS SENTAMOS A INTERPRETAR LOS RESULTADOS. FUE INTERESANTE VER CÓMO LOS DATOS ESTADÍSTICOS SE CONECTABAN CON EL CONTEXTO GEOGRÁFICO Y DEMOGRÁFICO. POR EJEMPLO, VIMOS QUE PAÍSES CON MUCHA POBLACIÓN NO SIEMPRE TIENEN MUCHA SUPERFICIE."

- **RAÚL:**

"SÍ, COMO BANGLADESH: UN PAÍS PEQUEÑO EN TÉRMINOS DE ÁREA, PERO CON UNA DENSIDAD POBLACIONAL MUY ALTA. ES UN CLARO EJEMPLO DE CÓMO LA DISTRIBUCIÓN DE RECURSOS PUEDE SER DESAFIANTE EN ESPACIOS REDUCIDOS."

- **SAMANTHA:**

"¡EXACTO! Y AL CONTRARIO ESTÁ AUSTRALIA: UN PAÍS ENORME, PERO CON UNA DENSIDAD DE POBLACIÓN RELATIVAMENTE BAJA. ESTO NOS HIZO REFLEXIONAR SOBRE CÓMO LA GEOGRAFÍA INFLUYE DIRECTAMENTE EN CÓMO LAS PERSONAS VIVEN Y SE ORGANIZAN."

- **CARLOS:**

"NOS HIZO PENSAR MUCHO SOBRE TEMAS COMO LA DISTRIBUCIÓN DE RECURSOS, LA PLANIFICACIÓN URBANA Y HASTA ASPECTOS ECONÓMICOS. ¿RECUERDAN CUANDO ANALIZAMOS LA RELACIÓN ENTRE DENSIDAD POBLACIONAL Y DESARROLLO ECONÓMICO?"

## **[REFLEXIÓN FINAL]**

- **SAMANTHA:**

"ESTE PROYECTO NO SOLO NOS ENSEÑÓ A USAR UNA API, SINO A TRABAJAR JUNTOS, DIVIDIR TAREAS Y CONSTRUIR SOLUCIONES INTELIGENTES PASO A PASO."

- **CARLOS:**

"ES COMO DECÍAN EN LOS APUNTES: YA NO ESTAMOS EMPEZANDO. YA FORMAMOS PARTE DEL MUNDO DE LA PROGRAMACIÓN. AHORA VEMOS UN PROBLEMA Y SABEMOS CÓMO RESOLVERLO."

- **EVELYN:**

"HOY EN DÍA, ESTOS CONOCIMIENTOS ESTÁN EN TODAS PARTES: INGENIERÍA, ECONOMÍA, CIENCIA DE DATOS, INCLUSO EN IA. Y NOSOTROS LOS DOMINAMOS."

- **ANA:**

"NO SOLO ENTENDIMOS PYTHON. ENTENDIMOS CÓMO USARLO PARA RESOLVER PROBLEMAS REALES. Y ESO, PARA MÍ, ES EL MAYOR LOGRO."

- **SAMANTHA:**

"ASÍ QUE, SI ALGUIEN PREGUNTA: ¿PARA QUÉ SIRVE PROGRAMAR EN PYTHON? NUESTRA RESPUESTA ES CLARA: PARA CONSTRUIR SOLUCIONES ÚTILES, CLARAS Y ESCALABLES. COMO ESTA."

- **RAÚL:**

"Y AUNQUE HAYA ERRORES, COMO ESA Ñ MAL CODIFICADA... SIEMPRE HAY UNA FORMA DE RESOLVERLOS. SOLO SE NECESITA PACIENCIA, LÓGICA Y TRABAJO EN EQUIPO."

- **CARLOS:**

"¡ESTE NO ES EL FINAL! ES APENAS EL COMIENZO. PRONTO HABLAREMOS DE HOJAS DE CÁLCULO, VISUALIZACIÓN GRÁFICA Y MANEJO DE ARCHIVOS, COMO DICE LA FASE III DE LOS APUNTES."

**[CIERRE - MÚSICA SUAVE DE FONDO]**

- **CARLOS:**  
Y ASÍ LLEGAMOS AL FINAL DE ESTE EPISODIO. NOS ENCANTÓ TRABAJAR JUNTOS Y APRENDER TANTO CON ESTE PROYECTO.
- **EVELYN:**  
FUE UN RETO, PERO LO RESOLVIMOS COMO EQUIPO. ¡GRACIAS POR ESCUCHARNOS!
- **SAMANTHA:**  
ESPERAMOS QUE LES HAYA GUSTADO Y QUE LES INSPIRE A HACER SUS PROPIOS PROYECTOS.
- **RAÚL:**  
NOS DESPEDIMOS: ANA PAMELA, EVELYN, SAMANTHA, RAÚL Y YO, CARLOS.
- **TODOS:**  
¡HASTA LA PRÓXIMA!

**PIA**  
**GRUPO**

*Previa*

PIA  
GRUPO  
*2ntro*

**PIA  
GRUPO  
2**

*Sección*

1

```
import requests
import json
import re
import statistics
import pandas as pd
import matplotlib.pyplot as plt
```

```

"""
modulo.py - Funciones para interactuar con la API REST Countries (https://restcountries.com).
Incluye métodos para obtener, estructurar y procesar datos geográficos y demográficos de países.
"""

def obtener_datos_paises():
    """
    Obtiene datos de todos los países desde la API REST Countries.

    Realiza una solicitud HTTP GET al endpoint oficial de REST Countries y devuelve
    información como nombre, población, área, idiomas, monedas, entre otros.

    Returns:
        list: Lista de diccionarios con datos de cada país (ej.: [{"nombre": "Colombia", "población": 50_882_891, ...}]).  

        None: Si ocurre un error en la conexión o la respuesta no es válida (ej.: código HTTP != 200).

    Ejemplo de dato devuelto:
    {
        "commonName": "Colombia",
        "officialName": "República de Colombia",
        "countryCode": "170",
        "region": "Americas",
        "population": 50882891,
        "area": 1141748,
        "languages": {"spa": "Spanish"},
        "currencies": {"COP": {"name": "Colombian peso", "symbol": "$"}}
    }
    """

# Endpoint oficial de REST Countries para obtener datos de todos los países
url = "https://restcountries.com/v3.1/all"
```

```

try:
    # Enviar solicitud GET con un tiempo máximo de espera de 10 segundos
    # (Evita que el programa se bloquee indefinidamente por fallos de red)
    respuesta = requests.get(url, timeout=10)

    # Verificar si la solicitud fue exitosa (código HTTP 200)
    # Lanza una excepción si el código de estado es 4xx o 5xx
    respuesta.raise_for_status()

    # Convertir la respuesta JSON en una lista de diccionarios de Python
    # La API devuelve datos en formato JSON, que Python interpreta automáticamente
    return respuesta.json()

except requests.exceptions.RequestException as e:
    # Manejar errores comunes de red:
    # - Fallos de conexión (DNS, servidor caído)
    # - Tiempo de espera excedido (timeout)
    # - Respuestas inválidas (ej.: JSON malformado)
    print(f"Error al conectar con la API: {e}")
    return None
```

PIA  
GRUPO  
2 2

*Sección*

```
"""
modulo.py - Funciones para estructurar y transformar datos de países obtenidos desde la API REST Countries.
"""

def estructurar_datos_paises(datos):
    """
    Convierte datos crudos de países en una lista de diccionarios con campos clave y normalizados.

    Procesa cada país para extraer y organizar información relevante como nombre, población, área,
    región, subregión, idiomas, monedas y densidad poblacional. Usa métodos seguros (.get())
    para evitar errores por campos inexistentes en la respuesta de la API.

    Args:
        datos (list): Lista de diccionarios obtenida desde la API REST Countries.
                       Cada diccionario representa un país con todos sus datos crudos.

    Returns:
        list: Lista de diccionarios con los siguientes campos para cada país:
              - Nombre: Nombre común del país (ej.: "Colombia").
              - Población: Número total de habitantes.
              - Área (km²): Extensión territorial en kilómetros cuadrados.
              - Densidad (hab/km²): Población dividida entre área, redondeado a 2 decimales.
              - Región: Macroregión geográfica (ej.: "América").
              - Subregión: Subdivisión de la región (ej.: "Sudamérica").
              - Idiomas: Idiomas oficiales separados por comas (ej.: "Español, Inglés").
              - Monedas: Monedas oficiales con su nombre completo (ej.: "COP (Peso colombiano)").

    """
    paises_estructurados = []

    for pais in datos:
        try:
            # Extraer nombre común del país desde el campo anidado "name"
            nombre = pais["name"]["common"]

            # Usar .get() para evitar KeyError si el campo no existe
            poblacion = pais.get("population", 0) # Población, 0 si no está disponible
            area = pais.get("area", 0) # Área en km², 0 si no está disponible

            # Región y subregión con valor por defecto "N/A" si no están presentes
            region = pais.get("region", "N/A")
            subregion = pais.get("subregion", "N/A")

            # Procesar idiomas: obtener valores del diccionario y unirlos en una cadena
            # Ejemplo: {"spa": "Spanish"} -> "Español"
            idiomas = ", ".join(pais.get("languages", {}).values()) or "N/A"

            # Procesar monedas: obtener diccionario de monedas y formatearlas
            # Ejemplo: {"code": "COP", "info": {"name": "Colombian peso"}} -> "COP (Colombian peso)"
            monedas = ", ".join([
                f"{info['code']} ({info['name']})" for code, info in pais.get("currencies", {}).items()])
            monedas or "N/A"

            # Calcular densidad poblacional (habitantes por km²)
            # Si el área es 0 (ej.: datos faltantes), la densidad se establece en 0
            densidad = calcular_densidad(poblacion, area)

            # Añadir el país procesado a la lista estructurada
            paises_estructurados.append({
                "Nombre": nombre,
                "Población": poblacion,
                "Área (km²)": area,
                "Densidad (hab/km²)": densidad,
                "Región": region,
                "Subregión": subregion,
                "Idiomas": idiomas,
                "Monedas": monedas
            })

        except Exception as e:
            # Registrar errores específicos de procesamiento sin detener la ejecución
            print(f"Error procesando país: {e}")

    return paises_estructurados
```

```
def calcular_densidad(poblacion, area):
    """
    Calcula la densidad poblacional de un país (habitantes por km²).

    Evita divisiones por cero y redondea el resultado a 2 decimales.

    Args:
        poblacion (int): Población total del país.
        área (float): Área territorial del país en km².

    Returns:
        float: Densidad poblacional calculada o 0 si el área es inválida (<= 0).
    """
    try:
        # Calcular densidad y redondear para mejorar legibilidad
        return round(poblacion / área, 2) if área > 0 else 0
    except ZeroDivisionError:
        # Manejar explícitamente división por cero (aunque ya está cubierto por el condicional)
        return 0
```

**PIA**  
**GRUPO**  
*Sección*  
2 3





**PIA**  
**GRUPO**  
*Sección*  
2 4

```
"""
modulo.py - Funciones adicionales para análisis estadístico de datos de países.
"""
```

```
def analizar_estadisticas(datos_estructurados, campo="Población"):
    """
    Calcula y organiza estadísticas descriptivas para un campo numérico específico en una lista de diccionarios.

    Esta función permite analizar campos como "Población", "Área (km²)" o "Densidad (hab/km²)"
    de los datos estructurados de países, proporcionando métricas clave para interpretación geográfica/demográfica.

    Args:
        datos_estructurados (list): Lista de diccionarios con datos de países (ej.: [{"Nombre": "Colombia", "Población": 50_882_891, ...}]). 
        campo (str): Nombre del campo numérico a analizar. Por defecto: "Población".

    Returns:
        dict: Diccionario con las siguientes estadísticas:
            - Campo: Nombre del campo analizado.
            - Media: Valor promedio del campo.
            - Mediana: Valor central del campo ordenado.
            - Moda: Valor más frecuente.
            - Varianza: Dispersion de los valores respecto a la media.
            - Desviación Estándar: Variabilidad promedio de los valores.
        None: Si no hay datos válidos o el campo no es numérico.

    Ejemplo de retorno exitoso:
    {
        "Campo": "Población",
        "Media": 39378755.88,
        "Mediana": 6888660.0,
        "Moda": 50882891,
        "Varianza": 12516570609612345,
        "Desviación Estándar": 111877480.37
    }
    """

    # Extraer valores del campo especificado, filtrando valores <= 0 (ej.: países sin dato disponible)
    valores = [pais[campo] for pais in datos_estructurados if pais.get(campo, 0) > 0]
```

```
if not valores:
    print(f"No hay datos válidos en el campo '{campo}'")
    return None

try:
    # Calcular estadísticas básicas usando el módulo statistics
    # Media: Promedio aritmético de los valores
    media = round(statistics.mean(valores), 2)

    # Mediana: Valor que divide los datos ordenados en dos partes iguales
    mediana = statistics.median(valores)

    # Moda: Valor que aparece con mayor frecuencia
    moda = statistics.mode(valores)

    # Varianza: Promedio del cuadrado de las diferencias respecto a la media (mide dispersión)
    varianza = round(statistics.variance(valores), 2)

    # Desviación estandar: Raíz cuadrada de la varianza (más interpretable que la varianza)
    desviacion_std = round(statistics.stdev(valores), 2)

    # Devolver estadísticas en un diccionario estructurado
    return {
        "Campo": campo,
        "Media": media,
        "Mediana": mediana,
        "Moda": moda,
        "Varianza": varianza,
        "Desviación Estándar": desviacion_std
    }

except statistics.StatisticsError as e:
    # Manejar errores comunes en cálculos estadísticos:
    # - StatisticsError: Si hay múltiples modas (en versiones antiguas de Python) o pocos datos para varianza
    print(f"Error al calcular estadísticas: {e}")
    return None
```

**PIA**  
**GRUPO**  
*Sección*  
2 5

```
"""
modulo.py - Funciones adicionales para exportar datos estructurados a Excel usando pandas.
"""
```

```
def exportar_datos_excel(datos, nombre_archivo="datos_paises.xlsx"):
    """
    Exporta una lista de diccionarios a un archivo Excel (.xlsx) y muestra una vista previa en consola.

    Este método convierte los datos estructurados (ej.: países con campos como nombre, población, región)
    en un DataFrame de pandas, lo que permite generar archivos Excel con formato tabular limpio y legible.

    Args:
        datos (list): Lista de diccionarios con datos de países (ej.: [{"Nombre": "Colombia", "Población": 50_882_891, ...}]). 
        nombre_archivo (str): Nombre del archivo Excel de salida. Por defecto: "datos_paises.xlsx".

    Returns:
        None: La función no devuelve valores, pero imprime mensajes de éxito o error.

    Ejemplo de salida en Excel:
    +-----+-----+-----+-----+-----+-----+-----+-----+
    | Nombre | Población | Área (km²) | Densidad (hab/km²) | Región | Subregión | Idiomas | Monedas |
    +-----+-----+-----+-----+-----+-----+-----+-----+
    | Colombia | 50882891 | 1141748 | 44.56 | América | Sudamérica | Español | COP (Peso colombiano) |
    | ... | ... | ... | ... | ... | ... | ... | ... |
    +---+---+---+---+---+---+---+---+
```

try:

```
# Convertir la lista de diccionarios a un DataFrame de pandas
# Un DataFrame es una estructura tabular ideal para análisis y exportación a Excel
df = pd.DataFrame(datos)

# Mostrar una vista previa de los datos que se exportarán (primeras 5 filas)
# Esto permite verificar que los campos estén correctamente estructurados antes de guardar
print(f"\nMostrando las primeras filas que se exportarán a {nombre_archivo}:")
print(df.head())
```

```
# Mostrar una vista previa de los datos que se exportarán (primeras 5 filas)
# Esto permite verificar que los campos estén correctamente estructurados antes de guardar
print(f"\nMostrando las primeras filas que se exportarán a {nombre_archivo}:")
print(df.head())

# Guardar el DataFrame en un archivo Excel (.xlsx)
# Parámetros clave:
# - index=False: Evita guardar el índice numérico por defecto de pandas
# - engine="openpyxl": Especifica el motor para trabajar con archivos .xlsx modernos
df.to_excel(nombre_archivo, index=False, engine="openpyxl")
print(f"Datos exportados exitosamente a {nombre_archivo}")
```

```
except Exception as e:
    # Manejar errores comunes:
    # - Falta de bibliotecas (pandas, openpyxl no instaladas)
    # - Datos con tipos no compatibles (ej.: objetos complejos en lugar de valores atómicos)
    print(f"Error al exportar a Excel: {e}")
```

**PIA**  
**GRUPO**  
*Sección*  
2<sup>6</sup>

```
"""
modulo.py - Funciones para generar gráficos con matplotlib basados en datos estructurados.
"""


```

```
def graficar_datos(datos, tipo_grafico="barras", titulo="Gráfico", eje_x="X", eje_y="Y", campo_x="Nombre", campo_y="Población"):
    """
    Genera gráficos de barras o líneas a partir de datos estructurados de países.

    Esta función usa matplotlib para visualizar datos como población, área o densidad de países.
    Permite personalizar el tipo de gráfico, etiquetas y campos a representar.

    Args:
        datos (list): Lista de diccionarios con datos de países (ej.: [{"Nombre": "Colombia", "Población": 50_882_891, ...}]). 
        tipo_grafico (str): Tipo de gráfico ("barras" o "lineas").
        titulo (str): Título del gráfico (ej.: "Top 10 Países por Población").
        eje_x (str): Etiqueta del eje X (ej.: "País").
        eje_y (str): Etiqueta del eje Y (ej.: "Población").
        campo_x (str): Campo de los datos para el eje X (ej.: "Nombre", "Región").
        campo_y (str): Campo de los datos para el eje Y (ej.: "Población", "Área (km²)"). 

    Returns:
        None: La función no devuelve valores, pero imprime mensajes de éxito o error.

    Ejemplo de uso:
        graficar_datos(
            top_poblacion,
            tipo_grafico="barras",
            titulo="Top 10 Países por Población",
            eje_x="País",
            eje_y="Población",
            campo_x="Nombre",
            campo_y="Población"
        )
    """

```

```
try:
    # Extraer valores para los ejes X e Y desde los datos estructurados
    # Ejemplo: Si campo_x="Nombre", valores_x = ["Colombia", "Brasil", ...]
    valores_x = [pais[campo_x] for pais in datos]
    valores_y = [pais[campo_y] for pais in datos]

    # Configurar estilo y tamaño del gráfico
    # Tamaño grande (12x6 pulgadas) para mejorar legibilidad
    plt.figure(figsize=(12, 6))

    # Título y etiquetas de ejes (alineado con estándares internacionales de gráficas)
    # Ejemplo: "Relación entre temperatura y tiempo de ebullición"
    plt.title(titulo)
    plt.xlabel(eje_x) # Ejemplo: "País"
    plt.ylabel(eje_y) # Ejemplo: "Población"

    # Agregar rejilla con estilo punteado y transparencia para no sobrecargar la visualización
    plt.grid(True, linestyle='--', alpha=0.5)

    # Seleccionar y dibujar el tipo de gráfico especificado
    if tipo_grafico == "barras":
        # Gráfico de barras para comparar categorías (ej.: países)
        plt.bar(valores_x, valores_y, color='skyblue')
    elif tipo_grafico == "lineas":
        # Gráfico de líneas para mostrar tendencias (ej.: cambio de población en el tiempo)
        plt.plot(valores_x, valores_y, marker='o', color='green', linestyle='-')
    else:
        # Validación de tipo de gráfico permitido
        print("Tipo de gráfico no válido. Use 'barras' o 'lineas'.")
        return

    # Rotar etiquetas del eje X para evitar solapamiento (ej.: nombres largos)
    # Alineación a la derecha para mejor visualización
    plt.xticks(rotation=45, ha='right')
```

```
# Rotar etiquetas del eje X para evitar solapamiento (ej.: nombres largos)
# Alineación a la derecha para mejor visualización
plt.xticks(rotation=45, ha='right')

# Guardar gráfico como imagen PNG con nombre basado en el título
# Ejemplo: "top_10_países_por_población.png"
nombre_archivo = f"{titulo.lower().replace(' ', '_)}.png"
plt.tight_layout() # Evitar recortes en etiquetas largas
plt.savefig(nombre_archivo) # Guardar gráfico
print(f"Gráfico guardado como {nombre_archivo}")

# Mostrar gráfico temporalmente y cerrarlo automáticamente (no bloquea ejecución)
plt.show(block=False)
plt.pause(2) # Mantener ventana abierta 2 segundos
plt.close() # Cerrar figura para liberar memoria

except Exception as e:
    # Manejar errores comunes:
    # - Claves inexistentes en los datos (ej.: campo_x o campo_y no válidos)
    # - Tipos de datos no compatibles (ej.: valores no numéricos en gráficos de línea)
    print(f"Error al crear el gráfico: {e}")
```

**PIA**  
**GRUPO**  
*Sección*  
27

```
"""
modulo.py - Funciones para interpretar resultados estadísticos de datos de países.
"""

def interpretar_resultados(estadisticas, datos_originales=None, campo="Población"):
```

```
    """
    Interpreta estadísticas descriptivas (media, mediana, moda, varianza) y las relaciona con contexto geográfico/demográfico.

    Esta función transforma números abstractos en conclusiones interpretables, explicando patrones como asimetría, dispersión o valores extremos.
    Es especialmente útil para contextualizar análisis estadísticos en campos como población, área o densidad de países.
    """

    Args:
        estadisticas (dict): Diccionario con estadísticas calculadas (ej.: {"Media": 39378755.88, "Mediana": 6888660.0, ...}).
        datos_originales (list): Lista de diccionarios con datos completos de países (opcional, para identificar máximos/minimos).
        campo (str): Campo analizado (ej.: "Población", "Área (km²)"). Usado junto con datos_originales.
```

```
    Returns:
        str: Texto con interpretaciones claras, listo para imprimir o guardar en informes.
```

```
Ejemplo de retorno:
```

```
    ## Interpretación de Estadísticas: Población ##
    - La **media** (39378755.88) y la **mediana** (6888660.0) muestran alta asimetría,
      indicando que hay valores extremos influyendo en el promedio.
    - El valor más frecuente (**moda**) es 50882891, lo cual podría representar un grupo de países con características similares.
    - La **varianza** (12516570609612345) y la **desviación estándar** (111877480.37) indican alta dispersión entre los valores.

    ## Contexto Geográfico/Demográfico ##
    - El país con mayor Población es China (1402112000).
    - El país con menor Población es Vatican City (451).
"""

# Validación inicial: si no hay estadísticas, retornar mensaje informativo
if not estadisticas:
    return "No hay estadísticas disponibles para interpretar."
```

```
interpretacion = []

# Título de sección para organizar resultados
interpretacion.append(f"## Interpretación de Estadísticas: {estadisticas['Campo']} ##")
```

```
# Comparar Media y Mediana para detectar asimetría
media = estadisticas["Media"]
mediana = estadisticas["Mediana"]
diferencia = abs(media - mediana)
```

```
# Calcular el umbral del 20% para evaluar asimetría (recomendado en análisis estadístico)
umbral = 0.2 * max(media, mediana)
```

```
if diferencia > umbral:
    # Alta asimetría: media > mediana -> colas positivas (ej.: países muy poblados)
    tendencia = "alta asimetría" if media > mediana else "baja asimetría"
    interpretacion.append(
        f"- La **media** ({media}) y la **mediana** ({mediana}) muestran {tendencia}, "
        "indicando que hay valores extremos influyendo en el promedio. "
        "Esto suele ocurrir cuando unos pocos países (ej.: China, India) dominan el total global."
    )
else:
    interpretacion.append(
        f"- La **media** ({media}) y la **mediana** ({mediana}) son similares, "
        "sugiriendo una distribución relativamente simétrica. "
        "Los países tienen tamaños comparables en este campo."
    )
```

```
# Moda: valor más frecuente
moda = estadisticas["Moda"]
interpretacion.append(
    f"- El valor más frecuente (**moda**) es {moda}, lo cual podría representar un grupo de países con características similares. "
    "Ejemplo: múltiples pequeñas naciones insulares con poblaciones cercanas a 50 millones."
)
```

```
# Varianza y Desviación Estándar: dispersión de datos
varianza = estadisticas["Varianza"]
desviacion_std = estadisticas["Desviación Estándar"]
coeficiente_variacion = desviacion_std / media if media != 0 else float('inf')
```



**PIA  
GRUPO**

*Ejecucion*

# PIA GRUPO *Reflexion*

**PIA  
GRUPO  
2**

*Cierre*