

# Haz click

Y que se haga la luz

Ana Pineda  
Juan Durán  
Silvia Alonso



# Recomendador de Tarifas de Luz

Un sistema innovador para recomendar la tarifa más conveniente y sugerir soluciones de energía solar.



# Motivación del proyecto

La elección del tema se debió a varios aspectos principales:

- Utilidad del proyecto
- Practicidad
- Reto para aplicar lo aprendido



# Estructura del proyecto >>>

1

2

3

4

5

## Obtención de datos

Obtención de tarifas con webscrapping a través de las páginas de las distintas compañías.

## Limpieza de datos

Limpieza de los datos obtenidos y formateo de los mismos.

## Creación de formulario

Desarrollo de un sistema que sugiere tarifas de luz basado en entradas del usuario.

## Placas solares

Calcula si la instalación de placas solares sería económica mente beneficiosa.

## Aplicación web

Queremos que todos puedan utilizar la aplicación, aunque no tengan de sus conocimientos de Python.

# Organización

- Uso de **Trello** para la organización y seguimiento de tareas.
- Uso de **Slack** para mantener la comunicación del equipo.

The image shows a Trello board titled "Proyecto". The board has four columns: "Lista de tareas", "Tareas pendientes", "En proceso", and "Hecho". Each column contains several cards with tasks and their status. The "Lista de tareas" column has one card: "Documentar procesos". The "Tareas pendientes" column has three cards: "Desarrollo App Web / Streamlit", "Realizar presentación: método y programa de presentación", and "Cierre del proyecto". The "En proceso" column has four cards: "Desarrollo de la lógica del modelo: funciones de cálculo de tarifas y placas solares", "Visualizar resultados: matplotlib, seaborn", "Explorar datos (EDA)", and "Limpiar y preprocesar datos". The "Hecho" column has five cards: "Definir objetivos del proyecto", "Identificar fuentes de datos: web de diferentes compañías eléctricas", "Seleccionar herramientas y tecnologías: VS Code, GitHub, Trello", "Configurar entorno de trabajo: repositorio GitHub", and "Recopilar datos: Web Scraping".

Projecto ★ Visible para el Espacio de trabajo Tablero Tabla Power-Ups Automatización Filtros JB A

**Lista de tareas**

- Documentar procesos

+ Añade una tarjeta

**Tareas pendientes**

- Desarrollo App Web / Streamlit
- Realizar presentación: método y programa de presentación
- Cierre del proyecto

+ Añade una tarjeta

**En proceso**

- Desarrollo de la lógica del modelo: funciones de cálculo de tarifas y placas solares
- Visualizar resultados: matplotlib, seaborn
- Explorar datos (EDA)
- Limpiar y preprocesar datos

+ Añade una tarjeta

**Hecho**

- Definir objetivos del proyecto
- Identificar fuentes de datos: web de diferentes compañías eléctricas
- Seleccionar herramientas y tecnologías: VS Code, GitHub, Trello
- Configurar entorno de trabajo: repositorio GitHub
- Recopilar datos: Web Scraping

+ Añade una tarjeta

# 1.- Obtención de datos

## Uso de web scraping

Realizamos la extracción de datos directamente desde los sitios web de varias compañías energéticas en España.

## Creación de dataframes

Obteniendo datasets útiles para el almacenamiento y posterior uso de los datos requeridos en el programa.

```
url = "https://www.enelenergy.es/precios"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Encontrar las tarjetas de tarifas
tarjetas = soup.find_all('div', {'data-testid': 'tariff-card'})[0]

# Extraer los datos de cada tarjeta
tarjetas = []
for tarjeta in tarjetas:
    titulo = tarjeta.find('h3').text.strip() if tarjeta.find('h3') else None
    descripcion = tarjeta.find('p', class_='sc-192m2ep-0 jZCOH')
    precios = tarjeta.find_all('p', class_='sc-192m2ep-0 jZCOH')

    # Extraer los precios de energía consumida
    energia_punta = "No encontrado"
    energia_llano = "No encontrado"
    energia_valle = "No encontrado"
    energia_labels = tarjeta.find_all('p', class_='sc-192m2ep-0 jZCOH')
    energia_values = tarjeta.find_all('h5', class_='sc-192m2ep-0 jZCOH')

    for label, value in zip(energia_labels, energia_values):
        if "Punta" in label.text and "€/kWh" in value.text:
            energia_punta = value.text
        if "Llano" in label.text and "€/kWh" in value.text:
            energia_llano = value.text
        if "Valle" in label.text and "€/kWh" in value.text:
            energia_valle = value.text

    tarjetas.append({
        'titulo': titulo,
        'descripcion': descripcion,
        'precios': {
            'punta': energia_punta,
            'llano': energia_llano,
            'valle': energia_valle
        }
    })
```

# 2.- Limpieza de datos

- Algunas compañías tienen varias tarifas y tuvimos que unificarlas.
- Conseguimos un dataframe por compañía.
- Limpiar y unificar formatos.
- Corregir datos faltantes.
- Sacar datos horarios de .txt y pasar a dataframe para poder utilizarlos.



```
# Importamos los datos
df_naturgy = pd.read_csv('naturgy.txt', sep=';')
df_natsolar = pd.read_csv('natsolar.txt', sep=';')
df_octopus = pd.read_csv('octopus.txt', sep=';')
df_repsol = pd.read_csv('repsol.txt', sep=';')
df_xem = pd.read_csv('xem.txt', sep=';')

# Unificamos las tarifas
# Filtramos la fila 0
fila_0 = df_naturgy.loc[0, 'Punta']
valor_punta = re.search(r'Punta: ([\d,]+€/kWh)', fila_0)
df_naturgy.at[0, 'Punta'] = valor_punta.group(1) if valor_punta else None
df_naturgy.at[0, 'Llano'] = valor_punta.group(1) if valor_punta else None
df_naturgy.at[0, 'Valle'] = valor_punta.group(1) if valor_punta else None

# Filtramos la fila 1
fila_1 = df_naturgy.loc[1, 'Punta']
valor_punta = re.search(r'Punta: ([\d,]+€/kWh)', fila_1)
valor_llano = re.search(r'Llano: ([\d,]+€/kWh)', fila_1)
valor_valle = re.search(r'Valle: ([\d,]+€/kWh)', fila_1)
df_naturgy.at[1, 'Punta'] = valor_punta.group(1) if valor_punta else None
df_naturgy.at[1, 'Llano'] = valor_llano.group(1) if valor_llano else None
df_naturgy.at[1, 'Valle'] = valor_valle.group(1) if valor_valle else None

# Unimos todos los datos en una misma dataframe
df_companias = pd.concat([df_naturgy, df_natsolar, df_octopus, df_repsol, df_xem])

# Añadimos el coste de la batería virtual de Repsol
df_companias['Bateria'] = df_companias['Excedentes']
df_companias.loc[2, 'Bateria'] = 0.0
df_companias.loc[6, 'Bateria'] = 0.0
df_companias.loc[10, 'Bateria'] = 1.80

# Cambiamos los NaN por 0
df_companias = df_companias.fillna(0)

# Ordenamos las columnas
orden = ['Empresa', 'Tarifa', 'Descripción', 'Punta', 'Llano', 'Valle', 'P1', 'P3', 'Excedentes', 'Coste de gestión', 'Bateria']
df_companias = df_companias[orden]

# Cambiamos las unidades
def unidades(celda):
    if '€/kWh' in celda:
        celda = str(celda).replace('€/kWh', '')
    return celda

df_companias = df_companias.applymap(unidades)
```

# 3.- Formulario de recomendación de tarifas

- Diseñamos un formulario interactivo para captar los datos del usuario y ofrecerle una recomendación sobre la tarifa energética más adecuada.
- Los datos ingresados son comparados con nuestro conjunto de datos de tarifas y los coeficientes de consumo ajustados según el perfil de hogar.
- Al completar el formulario, el usuario recibe una recomendación optimizada de tarifas, basada en su perfil energético.



```
    - (empresa["punta"] * datos_consumo["punta"] +
     empresa["llano"] * datos_consumo["llano"] +
     empresa["valle"] * datos_consumo["valle"])
     excedentes_total = potencia * dias * (empresa["P1"] + empresa["P3"])
     #excedentes_total = excedentes * empresa["excedente"]

     #precio = ((potencia_total + bono_social * dias + consumo_total) * (1 + impuesto) +
     #           | #equipos + empresa["bateria"] * dias) * iva - excedentes_total

     #precio_sin = ((potencia_total + bono_social * dias + max(consumo_total - excedentes_total, 0)) * (1 + impuesto) +
     #               | #equipos + empresa["bateria"] * dias) * iva

     precionormal = ((potencia_total + bono_social * dias + consumo_total) * (1 + impuesto) +
                     | equipos) * iva

     calculado = {
         "Empresa": empresa["compañia"],
         "Tarifa": empresa["nombre"],
         "Precio": precionormal
     }
     comparativa.append(calculado)

     comparativa.sort(key=lambda x: x["Precio"])
     mejor_tarifa = comparativa[0]

     print("\nComparativa de tarifas:")
     for tarifa in comparativa:
         print(f"\t{tarifa['Empresa']} - {tarifa['Tarifa']}: {tarifa['Precio']:.2f} €")

     print(f"\nLa tarifa más económica es la de {mejor_tarifa['Empresa']} - {mejor_tarifa['Tarifa']} con un precio de {mejor_tarifa['Precio']:.2f} €")

# Ejecución del formulario y cálculo de la mejor tarifa
tarifas = cargar_tarifas()
datos_consumo = formulario_consumo()
calcular_mejor_tarifa(datos_consumo, tarifas)
```

# 4.- Estudio de placas solares

Con los datos del usuario comprobamos si el uso de placas solares supondría un ahorro considerable en su caso.

Son varios los factores tenidos en cuenta para esta recomendación:



Ciudad



Horas de sol por mes



Irradiación solar



Cálculo de la distribución del consumo



```
'Nov': 30,  
'Dec': 31}
```

```
# Crear el diccionario para almacenar la  
meses = df_ciudad.columns.difference(['D  
horas_sol = {mes: df_ciudad[mes].sum()*f  
porcen_sol = {mes: round(df_ciudad[mes].  
# Aproximadamente entre el 40% y el 50%  
# Calcular el total de todos los meses
```

```
# Para un mes dado  
mes = 'Jul'  
c_punta = 188.39  
c_llano = 233.13  
c_valle = 355.02
```

```
#mes = 'Feb'  
#c_punta = 188.39  
#c_llano = 233.13  
#c_valle = 355.02
```

```
total = c_punta + c_llano + c_valle
```

```
consumo anual de energía por parte  
10 % de la energía anual generada
```

```
e_consumo = e_generada *0.8
```

```
n_placas = int(c_total / (pot_placa * 0.
```

```
ada = n_placas * pot_placa * 0.7  
ada #912.0481066666669
```

```
c_consumo = c_total * 0.4
```

# 5.- Aplicación web

-Interfaz de usuario amigable

-Posibilidad de introducir datos de consumo

-Interfaz basada en web usando Streamlit

-Cualquier persona con conexión a internet puede usarla.

Entra y prueba en:

<https://electric-tariff-recommender.streamlit.app>



## Formulario de Consumo

¿Para qué mes deseas hacer la comparativa?

enero

¿En qué provincia te encuentras?

¿Cuántas personas viven en tu domicilio?

1

¿Conoces tu consumo en kWh para los períodos de Valle, Llano y Punta?

- Sí  
 No

Vamos a estimar tu consumo. Por favor, indica si tienes los siguientes electrodomésticos:

¿Tienes Frigorífico?

- Sí  
 No

¿Tienes Lavadora?

- Sí  
 No

¿Tienes Lavavajillas?

- Sí  
 No

¿Tienes Televisor?

- Sí  
 No

¿Tienes Aire Acondicionado?

Sí

# Dificultades enfrentadas

## Variabilidad de las páginas web para el webscraping

- La diversidad en la estructura de las páginas web complicó la recolección de datos de manera uniforme.

## Gran cantidad de datos

- Trabajamos con muchos archivos.

## Lógica del programa

- La lógica del programa se complicaba cada vez más.

## Nuevas herramientas y librerías

- Para lograr los resultados deseados, tuvimos que investigar y utilizar nuevas librerías y herramientas.



# Librerías utilizadas



Para **webscrapping**:

- BeautifulSoup
- Request
- Selenium

Para la **limpieza**:

- pandas

Para la **aplicación**:

- streamlit

Para los **gráficos**:

- seaborn
- matplotlib

```
import streamlit as st
import pandas as pd
import os
import matplotlib.pyplot as plt
# Obtener la ruta absoluta
current_dir = os.path.abspath(os.path.dirname(__file__))

# Crear un DataFrame con los datos
df = pd.read_csv('data.csv')

# Mostrar el DataFrame en Streamlit
st.dataframe(df)
```

**Gracias por  
vuestra  
atención**



**electric-tariff-recommender.streamlit.app**

