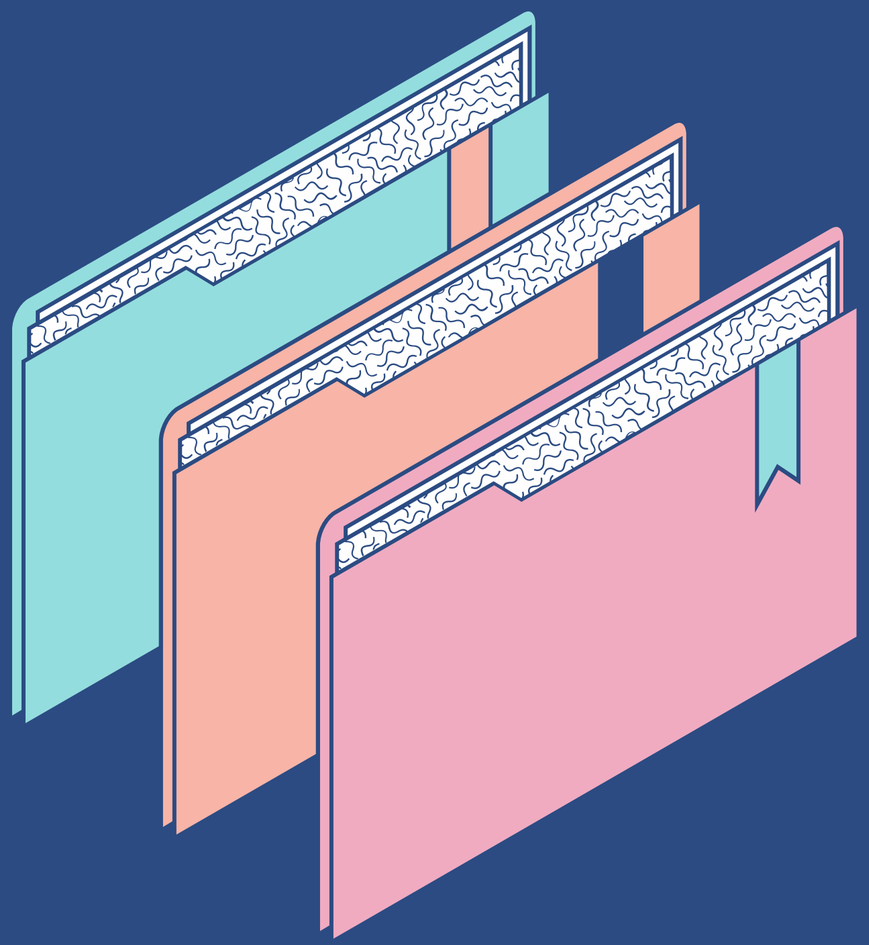
UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍAS
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN E INFORMÁTICA

Grupo ZZZ:
Nathalie Alfaro Quesada - B90221
Diego Bolaños Villalobos - C21256
José Pablo Mora Cubillo - B75044

CI - 0124 Computabilidad y Complejidad
II Semestre, 2024



TAREA PROGRAMADA 1
FINAL

routes.py

Avance5 > app > routes.py

```python
"""This module contains all the webapp routes and feature processing."""

import base64
import json
from io import BytesIO
from flask import render_template
from flask import current_app as app
from parser import parser
from parser import xml_dict
from .utilities import get_category_frequency,get_image_frequent_categories

# EXAMPLE DATA
dictionary = {}
with open("app/data/dataset.json", "r", encoding="utf8") as data:
    dictionary = json.loads(data.read())

#with open("data/mediplus_example.xml", "r", encoding='utf-8') as file:
with open("data/mplus_topics_2024-08-10(2).xml", "r", encoding='utf-8') as file:
    data = file.read()
    parser.parse(data)
#print(xml_dict)

"""
We generate the plot once per server execution. The `10 most frequent categories` plot
is saved to a BytesIO buffer.
"""
frequent_category_plot_buffer = get_image_frequent_categories(get_category_frequency(xml_dict))

```

```python
@app.route('/', methods=['GET'])
def index():
    """Returns rendered root page."""
    return render_template("index.html", date=xml_dict['date_created'],
                                         time=xml_dict['time_created'],
                                         counts=xml_dict['total'])

@app.route('/feature1.html', methods=['GET'])
def render_feature_1():
    """Returns rendered feature1 page."""
    #return render_template("feature1.html", json_data=dictionary)
    return render_template("feature1.html", json_data= {'health_topics' : xml_dict['health_topics'][0:50]})

@app.route('/feature2.html', methods=['GET'])
def render_feature_2():
    """Returns rendered feature2 page."""
    # Encode the image to base64 for embedding in HTML
    category_plot = base64.b64encode(frequent_category_plot_buffer.getvalue()).decode('utf8')

    # Render the template with the enbedding plot
    return render_template('feature2.html', category_plot=category_plot)

@app.route('/feature3.html', methods=['GET'])
def render_feature_3():
    """Returns rendered feature3 page."""
    return render_template("feature3.html",json_data=xml_dict)
```

utilities.py

Avance5 > app > utilities.py

```python
from io import BytesIO
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

def get_category_frequency(medline_dict):
    """
    This function takes a dictionary `medline_dict` and counts the frequency of information categories
    within health topics. It returns a dictionary with the categories and their respective counts.
    """
    # dictionary used to storage the category counts
    inf_category_counts = {}
    # verify that the key exists
    if 'health_topics' in medline_dict:
        # verify the value is a list
        if isinstance(medline_dict['health_topics'], list):
            for topic in medline_dict['health_topics']:
                # verify that topic is a dict and the existence of the key
                if isinstance(topic, dict) and 'tags' in topic:
                    #print("Topic es dic y existe tags")
                    if 'site' in topic['tags']:
                        if isinstance(topic['tags']['site'], list):
                            for site in topic['tags']['site']:
                                if ('tags' in site) and (isinstance(site['tags'], dict)):
                                    if ('information-category' in site['tags']) and (isinstance(site['tags']['information-category'], list)):
                                        for category in site['tags']['information-category']:
                                            if category in inf_category_counts:
                                            # if the key is in the dict, increase the counter
                                                inf_category_counts[category] += 1
                                            else:
                                                # add the key with initial value of 1
                                                inf_category_counts[category] = 1
```

```
    print("-----------------------------------------------------------------------------")
    print(f"Log (frequency of `information-category` tag):\n{inf_category_counts}")
    print("-----------------------------------------------------------------------------")
    return inf_category_counts


def get_image_frequent_categories(inf_category_counts):
    """
    This function takes a dictionary `inf_category_counts` and generates a horizontal bar plot
    of the 10 most frequent categories. The plot is saved to a BytesIO buffer and returned.
    """
    columns=['Category', 'Count']

    # cast the `inf_category_counts` dictionary to a dataFrame
    df_category_counts = pd.DataFrame(list(inf_category_counts.items()), columns=columns)
    # get the 10 most popular categories
    popular_category_counts = df_category_counts.sort_values(by='Count', ascending=False).head(10)

    # create a horizontal barplot
    plt.figure(figsize=(6, 6))
    sns.barplot(x='Count', y='Category', data=popular_category_counts, orient='h')
    plt.xlabel('Count')
    plt.ylabel('Category')
    plt.tight_layout()
    buffer = BytesIO()
    plt.savefig(buffer, format='png')
    plt.close()
    return buffer
```

__init__.py

Avance5 > app > __init__.py

```python
"""
Creates and configures flask app. Uses the factory design pattern.
"""
from flask import Flask
import sys

def create_app():
    """Returns a flask app"""
    try:
        app = Flask(__name__)
        app.jinja_env.trim_blocks = True
        app.jinja_env.lstrip_blocks = True
        with app.app_context():
            from . import routes
    except SyntaxError as e:
        print(f"{e}\nServer stopped.")
        sys.exit(1)  # Stop the server with exit code 1
    except Exception as e:
        print(f"Error creating the application: {e}")
        sys.exit(1)  # Stop the server with exit code 1
    return app

```

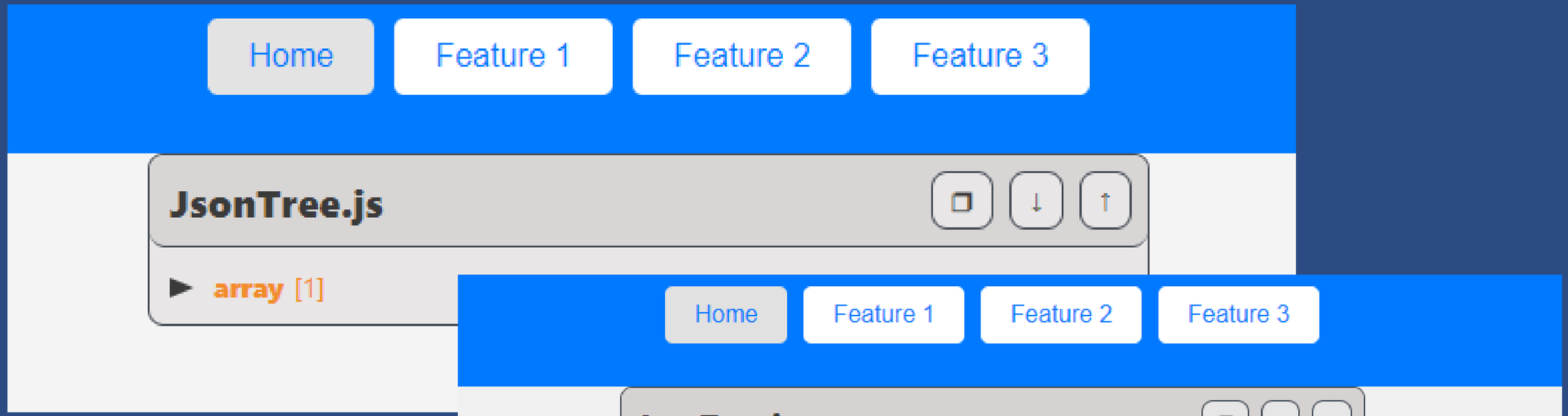
Home | Feature 1 | Feature 2 | Feature 3

MedlinePlus: health topics registry

Date of last update of data: 08/10/2024

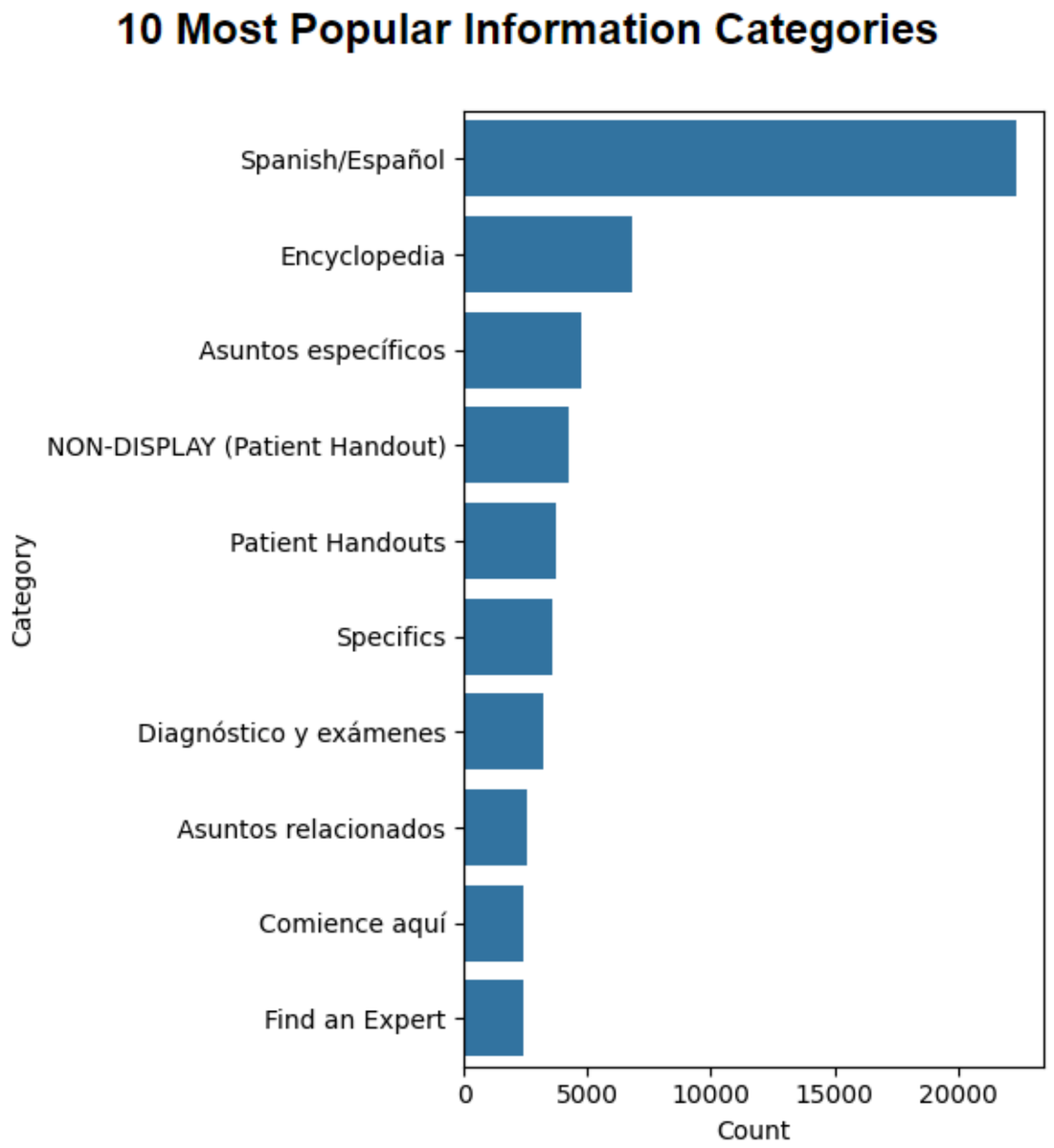
Time of last update of data: 02:30:32

Number of items: 2044


Home
Feature 1
Feature 2
Feature 3
JsonTree.js
array [1]

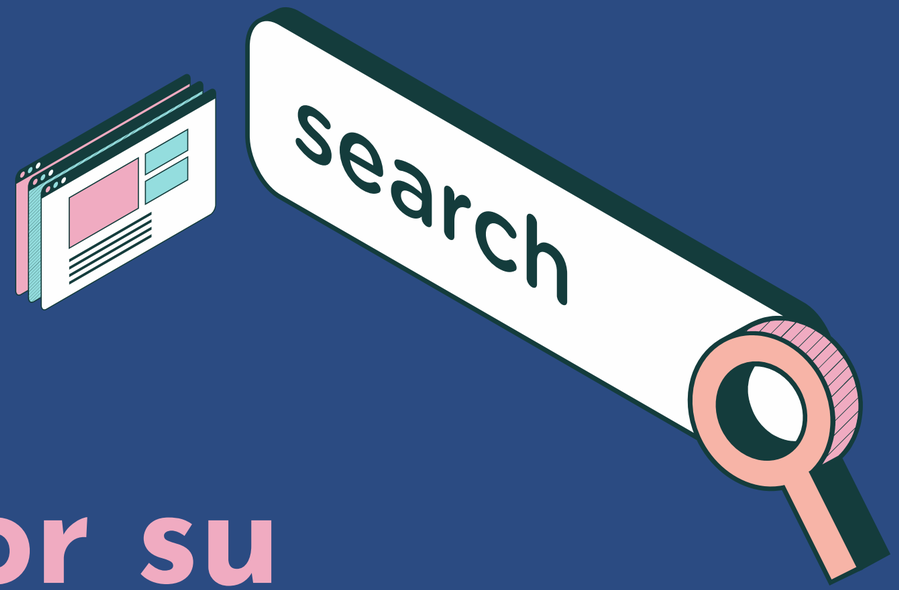

Home
Feature 1
Feature 2
Feature 3
JsonTree.js
array [1]
[0] : object {1}
health_topics : array [50]
[00] : object {2}
attributes : object {6}
date-created : 21st July 2002 00:00:00
id : 2083
language : "Spanish"
meta-desc : "Una úlcera péptica es una llaga en la mucosa qu
title : "Úlcera péptica"
url : "https://medlineplus.gov/spanish/pepticulcer.html"
tags : object {8}
also-called : array [2]
[0] : "Úlcera gástrica"

Home | Feature 1 | Feature 2 | Feature 3



Home | Feature 1 | Feature 2 | Feature 3

Title	Description	Also Called	Groups	Resources
Úlcera péptica	Una úlcera péptica es una llaga en la mucosa que recubre el estómago o el duodeno. El síntoma más común es ardor en el estómago. Entérese aquí.	• Úlcera gástrica • Úlcera duodenal	• Sistema digestivo	◦ Úlceras pépticas: Diagnóstico y tratamiento ◦ Úlceras (para adolescentes) ◦ Úlceras
Úlcera por presión	Las úlceras por presión son áreas de piel lesionada por permanecer en una misma posición durante mucho tiempo. Tratamiento y prevención.	• Llagas por presión • Escaras	• Piel, cabello y uñas	◦ Úlceras por presión (y cáncer) ◦ Úlceras por presión ◦ Úlceras de decúbito: Qué preguntarle al médico
Ántrax	El ántrax o carbunco es ocasionado por una bacteria. Entérese de síntomas, diagnóstico y tratamientos.	• Carbunco	• Infecciones	◦ Ántrax maligno (carbunco): No es algo del pasado ◦ Ántrax ◦ Guía básica para comprender el ántrax



Gracias por su atención