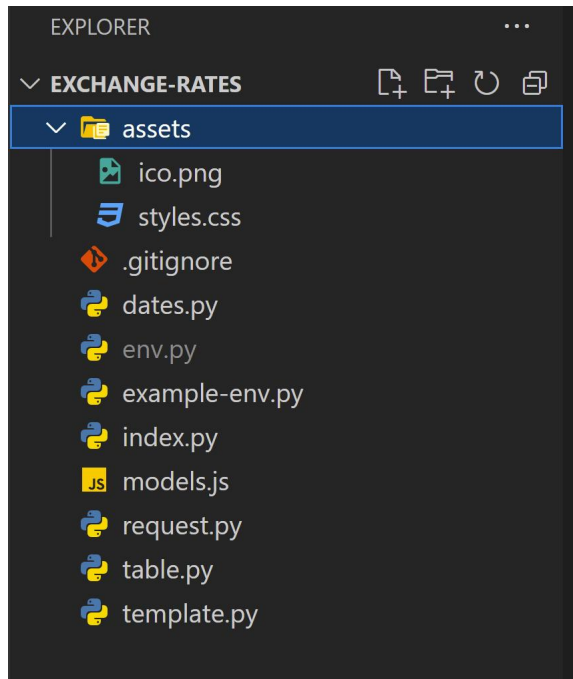


ACTIVIDAD 1. CONSUMO DE TIPOS DE CAMBIOS DE API

- Estructura del código



- Descripción de cada archivo

- ✓ *assets* : Carpeta donde están archivos de diseño
 - ◆ *ico.png*: imagen utilizada como favicon en la página
 - ◆ *styles.css*: hoja de estilos para la página HTML
- ✓ *.gitignore*: archivo de configuración para control de versiones, más información en <https://git-scm.com/doc>
- ✓ *dates.py*: aquí está la configuración de los días que seleccionará el calendario (según instrucciones eran los últimos 5 días hábiles)
- ✓ *env.py* / *example-env.py*: son las variables de entorno dentro del proyecto, el archivo *env.py* lo debe crear cada programador que ejecute el programa usando de referencia *example-env.py*
- ✓ *index.py*: es donde el programa se inicia
- ✓ *models.js*: archivo de referencia para validar como reestructurar los datos que retorna el API en formato JSON

- ✓ *request.py*: aqui estan las funciones que hacen las consultas a la API
- ✓ *table.py*: es la tabla resultante al consultar los datos
- ✓ *template.py*: es la plantilla del proyecto (html)
- Codigo de cada archivo

➤ *styles.css*

```
@import url('https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap');

body {
  background-color: #f2f2f2;
}
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Lato', sans-serif;
}
.wrapper {
  padding: 1rem 2rem;
  height: 100%;
}
.main-title {
  font-size: 1.5rem;
}
.paragraph-lead {
  font-size: 1.25rem;
  color: #777777;
}
.card {
  background-color: #ffffff;
  margin: 1.5rem 0;
  padding: 1rem 1.25rem;
  border: 0.1rem solid #c4c4c4;
  border-radius: 0.3rem;
  min-height: 18rem;
  display: grid;
}
.card-title {
  margin-bottom: 1rem;
}
.card-text {
  margin-bottom: 3.5rem;
}
.card-button {
  background-color: #0099f9;
  color: #ffffff;
  font-weight: bold;
  border: 0.2rem solid #0099f9;
  border-radius: 0.5rem;
  padding: 1rem 1.5rem;
  cursor: pointer;
  transition: all 0.15s;
  margin: auto;
  text-align: center;
  width: 100%;
}
.button {
  margin: 15px auto 5px;
  font-size: 1rem;
}
.card-button:hover {
  background-color: #ffffff;
  color: #0099f9;
}
.loader {
  display: none;
}
.block {
  display: block;
}
.tabla {
  margin-bottom: 20px !important;
}
.label {
  font-size: 1rem;
  font-style: italic;
  color: #cecece;
  margin-top: 5px;
}
```

➤ *.gitignore (eso archivos no se incluyen en git)*

```
env.py
__pycache__
```

➤ *dates.py*

```
from datetime import datetime, timedelta

today = datetime.now()
name_day = today.strftime('%A')

if name_day == 'Thursday':
    date_end = today - timedelta(days=6)
elif name_day == 'Wednesday':
    date_end = today - timedelta(days=5)
elif name_day == 'Tuesday':
    date_end = today - timedelta(days=4)
elif name_day == 'Monday':
    date_end = today - timedelta(days=3)
elif name_day == 'Sunday':
    date_end = today - timedelta(days=2)
elif name_day == 'Saturday':
    date_end = today - timedelta(days=1)
else:
    date_end = today

date_start = date_end - timedelta(days=4)
```

➤ *example-env.py*

```
url_api = 'https://...'
key_api = 'key'
prod = True
```

➤ *index.py*

```
import dash_bootstrap_components as dbc
from dash import Dash, ctx
from dash.dependencies import Input, Output

from env import prod
from request import getData
from table import table_Data
from template import body

app = Dash(__name__, title='Exchange Rates',
           external_stylesheets=[dbc.themes.BOOTSTRAP])
app._favicon = ("ico.png")
app.layout = body

@app.callback(
    Output('res', 'children'),
    [Input('submit', 'n_clicks'),
     Input('my-date-picker-range', 'start_date'),
     Input('my-date-picker-range', 'end_date'),
     Input('country-select', 'value')],
)

def update_output(btn, start_date, end_date, select):
    if 'submit' == ctx.triggered_id:
        s = select
        rows = 1
        if (type(select) == list):
            s = ",".join(select)
            rows = len(select)
        data = getData(start_date, end_date, s)
        # info = 'Revisa la consola'
        info = table_Data(data, rows)
    else:
        info = 'Para obtener los datos da completa la informacion y da click en el boton'
    return info

if __name__ == '__main__':
    app.run_server(debug=prod)
```

➤ *models.js*

```
// Como responde el api
const response = {
  base: 'USD',
  end_date: '2023-02-09',
  rates: {
    '2023-02-06': {
      GTQ: 7.84737,
      JPY: 132.626495,
    },
    '2023-02-07': {
      GTQ: 7.838025,
      JPY: 131.046501,
    },
    '2023-02-08': {
      GTQ: 7.839479,
      JPY: 131.378504,
    }
  },
  start_date: '2023-02-06',
  success: true,
  timeseries: true,
};

// Trasformar a
const datos = [
  {
    fecha: '2023-02-06',
    monedas: [
      {
        iso: 'JPY',
        pais: 'Japan',
        valor: 132.626495,
      }
    ],
  },
  {
    fecha: '2023-02-07',
    monedas: [
      {
        iso: 'JPY',
        pais: 'Japan',
        valor: 132.626495,
      }
    ],
  },
];

// Como responde el API
const symbols = {
  success: true,
  symbols: {
    AED: 'United Arab Emirates Dirham',
    AFN: 'Afghan Afghani',
    ALL: 'Albanian Lek',
    AMD: 'Armenian Dram',
    ANG: 'Netherlands Antillean Guilder',
    AOA: 'Angolan Kwanza',
    ARS: 'Argentine Peso',
    AUD: 'Australian Dollar',
    AWG: 'Aruban Florin',
    AZN: 'Azerbaijani Manat',
  },
};

// Trasnformar a
const monedas = [
  {
    value: 'AED',
    label: 'United Arab Emirates Dirham',
  },
  {
    value: 'AFN',
    label: 'Afghan Afghani',
  },
];
```

➤ *Request.py*

```

import json
import pandas as pd
import requests
import tableprint

from env import key_api, url_api

def getData(start, end, coins):
    url = url_api + "/timeseries?start_date=" + start + "&end_date=" + end + \
        "&base=USD&symbols="+coins
    payload = {}
    headers = {
        "apikey": key_api
    }
    response = requests.request("GET", url, headers=headers, data=payload)
    result = response.text
    datos = json.loads(result)
    rates = datos['rates']
    resultado = [
        {
            'fecha': fecha,
            'monedas': [
                {
                    'iso': iso,
                    'pais': '',
                    'valor': round(valor, 2),
                } for iso, valor in monedas.items()
            ]
        } for fecha, monedas in rates.items()
    ]
    symbols = getSymbols() # Obtener la lista de símbolos
    for r in resultado:
        for m in r['monedas']:
            for s in symbols:
                if s['value'] == m['iso']:
                    m['pais'] = s['label'] # Buscar el país y asignar su valor
    pd.set_option('display.max_rows', 1500)
    df = pd.DataFrame(
        [(d['fecha'], x['iso'], x['pais'], round(x['valor'], 2))
         for d in resultado for x in d['monedas']],
        columns=['Fecha', 'Moneda ISO', 'Nombre Moneda', 'Valor']
    )
    data = [(d['fecha'], x['iso'], x['pais'], round(x['valor'], 2))
            for d in resultado for x in d['monedas']]
    headers = ['Fecha', 'Moneda ISO', 'Nombre Moneda', 'Valor']
    print('----- Cambios de USD -----')
    # print(df)
    tableprint.table(data, headers)
    print("_____")
    return resultado

def getSymbols():
    url = url_api + "/symbols"
    payload = {}
    headers = {
        "apikey": key_api
    }
    response = requests.request("GET", url, headers=headers, data=payload)
    result = response.text
    datos = json.loads(result)
    symbols = datos['symbols']
    resultado = [
        {
            'value': iso,
            'label': nombre
        } for iso, nombre in symbols.items()
    ]
    return resultado

```

➤ *table.py*

```

import dash_bootstrap_components as dbc
from dash import html

def table_Data(data, span):
    rows = []
    for row in data:
        fecha = row['fecha']
        is_first_row = True
        for moneda in row['monedas']:
            iso = moneda['iso']
            pais = moneda['pais']
            valor = moneda['valor']
            if is_first_row: # si es la primera fila con esa fecha
                rows.append(html.Tr([
                    html.Td(fecha, rowspan=span, style={
                        'verticalAlign': 'middle'}),
                    html.Td(iso),
                    html.Td(pais),
                    html.Td(round(valor, 2))
                ]))
            is_first_row = False # marcar como que ya no es la primera fila con esa fecha
        else:
            rows.append(html.Tr([
                html.Td(iso),
                html.Td(pais),
                html.Td(round(valor, 2))
            ]))

    return html.Div([
        html.Div([
            dbc.Label('Resultados de cambios a USD', className='card-title'),
        ]),
        html.Div([
            html.Table([
                html.Thead([
                    html.Tr([
                        html.Th('Fecha', style={'width': '25px'}),
                        html.Th('Moneda'),
                        html.Th('Nombre Moneda'),
                        html.Th('Valor')
                    ])
                ]),
                html.Tbody(rows)
            ], className='table table-bordered table-primary table-striped text-center border
            border-dark tabla',
            style={'width': '90%', 'margin': 'auto'})
        ])
    ])

```

➤ *Template.py*

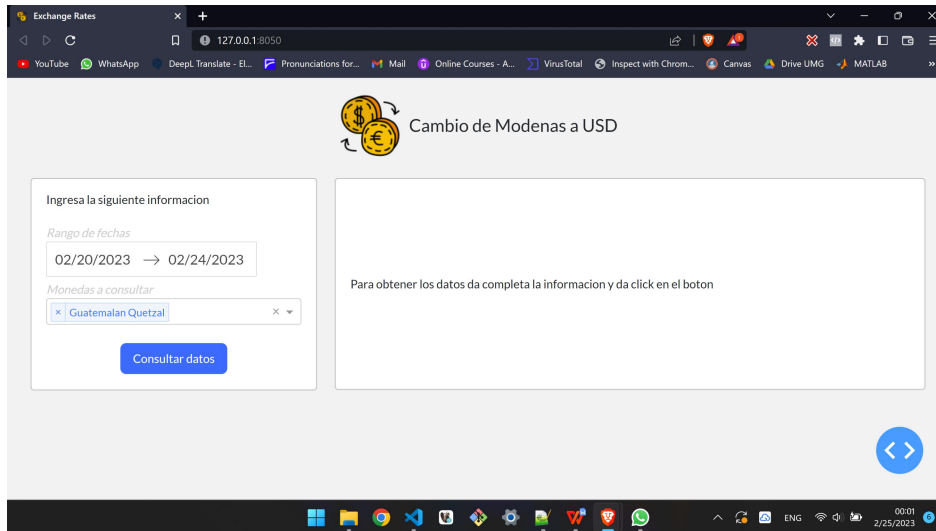
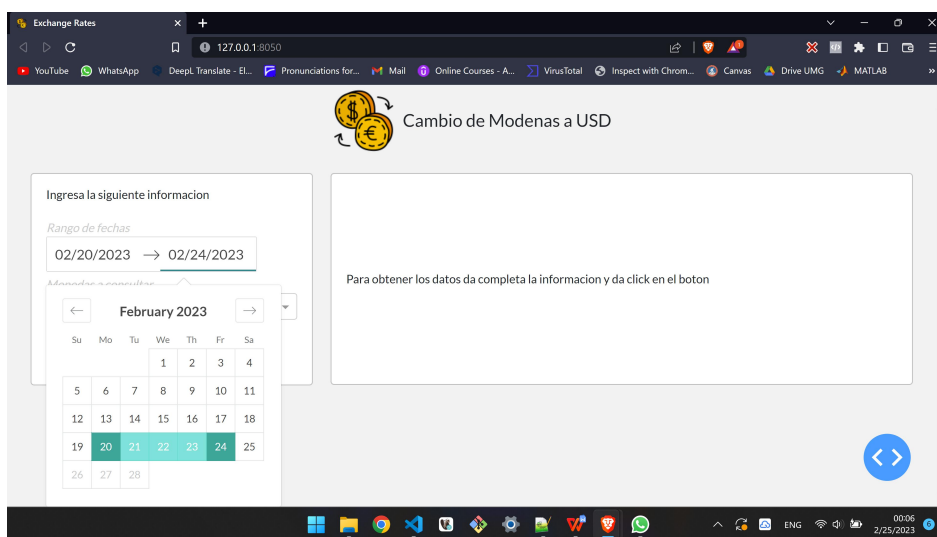
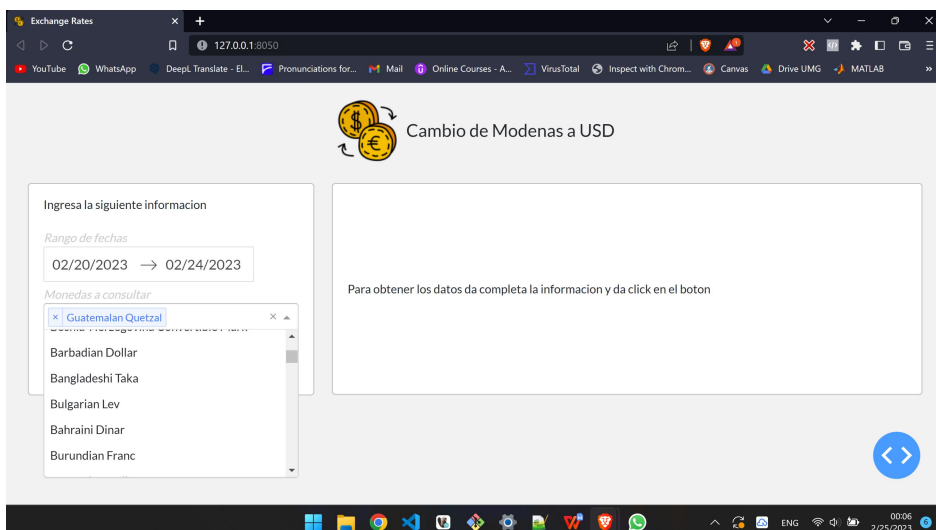
```

from datetime import date

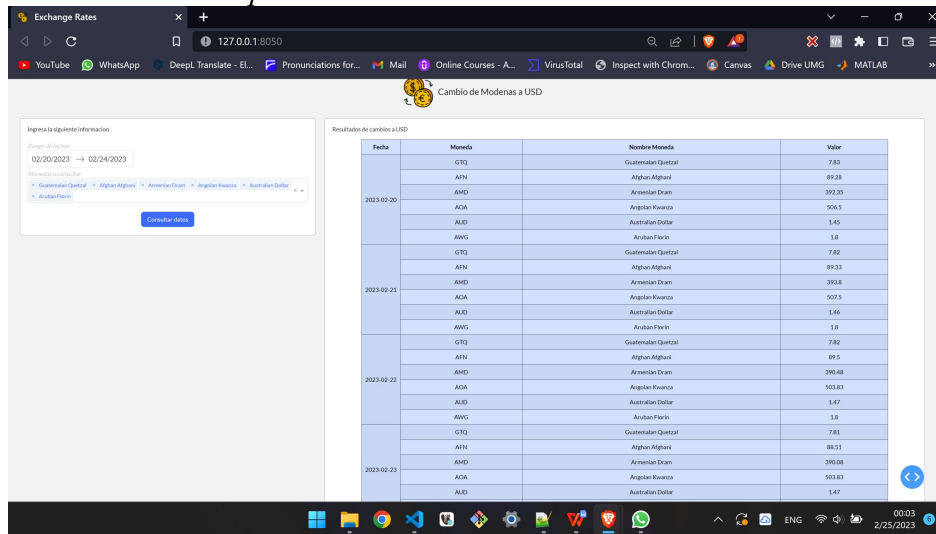
import dash_bootstrap_components as dbc
from dash import dcc, html
from dates import date_end, date_start, today
from request import getSymbols
all_option = []
body = html.Div(
    className='wrapper',
    children=[
        dbc.Row([
            dbc.Col([
                html.Div(children=[
                    html.Img(src='assets/ico.png'),
                    html.H3('Cambio de Monedas a USD', className='main-title',
                        style={'margin': 'auto 5px'}),
                ], style={'display': 'flex', 'margin': 'auto',
                    'justifyContent': 'center'})
            ])
        ]),
        dbc.Row([
            dbc.Col([
                html.Div(
                    className='card',
                    children=[
                        dbc.Label('Ingresa la siguiente informacion',
                            className='card-title'),
                        html.Span("Rango de fechas", className='label'),
                        dcc.DatePickerRange(
                            id='my-date-picker-range',
                            min_date_allowed=date(2023, 1, 1),
                            max_date_allowed=date(
                                today.year, today.month, today.day),
                            initial_visible_month=date(
                                today.year, today.month, today.day),
                            end_date=date(
                                date_end.year, date_end.month, date_end.day),
                            start_date=date(date_start.year,
                                date_start.month, date_start.day),
                            className='date-picker'
                        ),
                        html.Span("Monedas a consultar", className='label'),
                        dcc.Dropdown(
                            id="country-select",
                            options=getSymbols(),
                            multi=True,
                            style={'marginBottom': '5px'},
                            placeholder='Selecciona....',
                            value='GTQ'
                        ),
                        html.Button('Consultar datos', id='submit',
                            className='btn btn-primary btn-lg button', n_clicks=0)
                    ]
                ),
            ], width=4),
            dbc.Col([
                html.Div(
                    className='card',
                    children=[
                        dbc.Spinner(children=[html.Div(id='res', className='date-picker', style={'margin': 'auto'})],
                            size="lg", color="primary", type="border", fullscreen=True,
                            spinner_style={'margin': 'auto'}),
                    ], style={'alignItems': 'center'})
            ], width=8)
        ]),
    ]
)

```

● Vista al iniciar el programa

● Resultados en pantalla



Exchange Rates

127.0.0.1:8050

Cambio de Monedas a USD

Ingresa la siguiente información

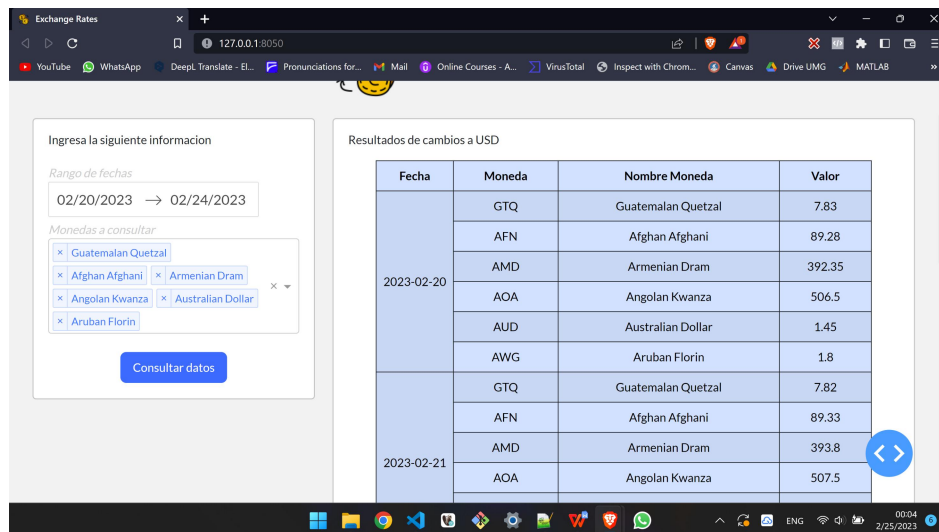
Rango de fechas: 02/20/2023 → 02/24/2023

Monedas a consultar: ☒ Guatemalan Quetzal ☒ Afghan Afghani ☒ Armenian Dram ☒ Angolan Kwanza ☒ Australian Dollar ☒ Aruban Florin

Consultar datos

Resultados de cambios a USD

Fecha	Moneda	Nombre Moneda	Valor
2023-02-20	GTQ	Guatemalan Quetzal	7.83
	AFN	Afghan Afghani	89.28
	AMD	Armenian Dram	392.35
	AOA	Angolan Kwanza	506.5
	AUD	Australian Dollar	1.45
	AWG	Aruban Florin	1.8
2023-02-21	GTQ	Guatemalan Quetzal	7.82
	AFN	Afghan Afghani	89.23
	AMD	Armenian Dram	392.8
	AOA	Angolan Kwanza	507.5
	AUD	Australian Dollar	1.46
	AWG	Aruban Florin	1.8
2023-02-22	GTQ	Guatemalan Quetzal	7.82
	AFN	Afghan Afghani	89.5
	AMD	Armenian Dram	390.48
	AOA	Angolan Kwanza	503.83
	AUD	Australian Dollar	1.47
	AWG	Aruban Florin	1.8
2023-02-23	GTQ	Guatemalan Quetzal	7.81
	AFN	Afghan Afghani	89.51
	AMD	Armenian Dram	380.08
	AOA	Angolan Kwanza	503.83
	AUD	Australian Dollar	1.47
	AWG	Aruban Florin	1.8



Exchange Rates

127.0.0.1:8050

Cambio de Monedas a USD

Ingresa la siguiente información

Rango de fechas: 02/20/2023 → 02/24/2023

Monedas a consultar: ☒ Guatemalan Quetzal ☒ Afghan Afghani ☒ Armenian Dram ☒ Angolan Kwanza ☒ Australian Dollar ☒ Aruban Florin

Consultar datos

Resultados de cambios a USD

Fecha	Moneda	Nombre Moneda	Valor
2023-02-20	GTQ	Guatemalan Quetzal	7.83
	AFN	Afghan Afghani	89.28
	AMD	Armenian Dram	392.35
	AOA	Angolan Kwanza	506.5
	AUD	Australian Dollar	1.45
	AWG	Aruban Florin	1.8
2023-02-21	GTQ	Guatemalan Quetzal	7.82
	AFN	Afghan Afghani	89.33
	AMD	Armenian Dram	393.8
	AOA	Angolan Kwanza	507.5

```
mconc@mconcoba MINGW64 ~/Desktop/Progra3/exchange-rates (main)
```

```
$ python index.py
```

```
Dash is running on http://127.0.0.1:8050/
```

```
* Serving Flask app 'index'
```

```
* Debug mode: on
```

```
----- Cambios de USD -----
```

Fecha	Moneda ISO	Nombre Moneda	Valor
2023-02-20	GTQ	Guatemalan Quetzal	7.83
2023-02-20	ALL	Albanian Lek	108.1
2023-02-20	AOA	Angolan Kwanza	506.5
2023-02-20	ARS	Argentine Peso	193.11
2023-02-20	AWG	Aruban Florin	1.8
2023-02-21	GTQ	Guatemalan Quetzal	7.82
2023-02-21	ALL	Albanian Lek	108.06
2023-02-23	AOA	Angolan Kwanza	503.83
2023-02-23	ARS	Argentine Peso	195.31
2023-02-23	AWG	Aruban Florin	1.8
2023-02-24	GTQ	Guatemalan Quetzal	7.81
2023-02-24	ALL	Albanian Lek	108.05
2023-02-24	AOA	Angolan Kwanza	506.5
2023-02-24	ARS	Argentine Peso	195.02
2023-02-24	AWG	Aruban Florin	1.8