Fundamentos de Programación con Python

EmTech - Proyecto N°01

Reporte - 01

Al. Vázquez Sánchez
Axel

Índice general

Capítulo 1

Introducción

Python remonta su origen a principios de los años 90, cuando Guido Van Rossum, un trabajador del Centrum Wiskunde & Informatica (CWI), un centro de investigación holandés, tuvo la idea de desarrollar un nuevo lenguaje basándose en un proyecto anterior, el lenguaje de programación 'ABC', que él mismo había desarrollado junto a sus compañeros.

Su filosofía fue la misma desde el primer momento: crear un lenguaje de programación que fuera muy fácil de aprender, escribir y entender, sin que esto frenara su potencial para crear cualquier tipo de aplicación. En aquellos años, el hardware que había no permitía tal cosa, y es por eso por lo que Python ha resurgido durante los últimos años, porque el avance de la tecnología ha permitido alcanzar el objetivo inicial de este lenguaje de programación adelantado a su tiempo.

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario 'traducirlo' a lenguaje máquina.

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años, ya que facilita trabajar con inteligencia artificial, big data, machine learning y data science, entre muchos otros campos en auge.

Capítulo 2

Definición del código

2.1. Login

Se implementa el apartado del *login* dentro del código, cuyo usuario es **admin** y clave de acceso es **3mt3ch**, se cuentan con 3 intentos para poder acceder al sistema, observe el siguiente código:

```
user = "admin"
pwd = "3mt3ch"

acces = False
tries = 0

while not acces:
    # Primero ingresa Credenciales
    usuario = input('Usuario: ')
    contra = input('Contraseña: ')
    tries += 1
    if usuario == user and contra == pwd:
        print(f'¡Bienvenido, {user}!')
        acces = True
    else:
        print(f'Tienes {3 - tries} intentos restantes')
        if usuario == user:
            print('Te equivocaste en la contraseña')
        else:
            print(f'El usuario: "{usuario}" no está registrado')

    if tries == 3:
        exit()
```

2.2. 5 productos con mayores ventas

Mediante la implementación de listas y diccionarios, se proporciona lo que pide en este inciso para el reporte, accediendo a lifestore_products, es de donde se va a obtener los datos de interés para poder dar solución a este apartado, observe el código empleado:

```
ventaprod = []; veces = []

```

```
ventaprod = [venta[1] for venta in lifestore_sales]

for venta in lifestore_products:
    nido = []
    veces.append(nido)
    for k in range(1):
        nido.append(venta[0])
        nido.append(venta[1])
        #veces vendidas mediante  implementación de count a var venta[0]
        nido.append(ventaprod.count(venta[0])) #[2]
        nido.append(venta[-2]) #lifestore_products posición [-2] \equiv category

def Sort(veces):
    #extrayendo x[2] \equiv var. aloja datos veces vendidas sobre producto
    veces.sort(key = lambda x: x[2])
    return veces

veces = Sort(veces)

print("Datos sobre los 5 productos con mayores ventas: \n")

#5 productos
for i in range(-1,-6,-1):
    print(f'ID: {veces[i][0]} \n Nombre: {veces[i][1]} \n No. Ventas: {veces[i][2]}')
```

Mediante la función **sort**, es como se ordena de forma ascendente la lista con los datos de interés respecto al número de ventas de los productos.

2.3. 10 productos con mayores búsquedas

Muy similar al proceso anterior, es lo que se implementó para este apartado, sólo que ahora se emplea a **lifestore_searches** para extraer la información de los datos de interés. También, se emplea la función **count** para la parte del conteo.

```
#Listas vacias para var's de interés:
busqueda_prod = [] ; veces_busqueda = []

#datos de interés en **lifestore_searches**

busqueda_prod = [busqueda[1] for busqueda in lifestore_searches]

for busqueda in lifestore_products:
    nido = []
    veces_busqueda.append(nido)
    for k in range(1):
        nido.append(busqueda[0])
        nido.append(busqueda[1])
        #similar a procedimiento en proc. 5 prods. más vendidos
        nido.append(busqueda_prod.count(busqueda[0]))
        nido.append(busqueda[-2])

def Sort(veces_busqueda):
    veces_busqueda.sort(key = lambda x: x[2])
    return veces_busqueda

veces_busqueda = Sort(veces_busqueda)

print('10 Productos con mayores busqueda: ')

#rango de -1 a -10 con step de -1 porque están almacenados de menor a mayor busqueda
for i in range(-1,-11,-1):
```

```
    print(f'ID: {veces_busqueda[i][0]} \n Nombre: {veces_busqueda[i][1]} No. busquedas: {veces_busqueda[i][2]}\n ')
```

para imprimir el número de datos que se solicitan, se indica implementando un **for** y el número de productos mediante el uso de **range()**.

2.4. 5 productos con menores ventas por categoría

Se implementan diccionarios para obtener el número exactos de categorías junto con una lista que incluya a todas y cada una de ellas. Se implementa una lista vacía para c/u y un ciclo for que recorre toda la lista **veces**.

```

categoria = []

#creando para c/u de las 8 categorías

processors = []; gpu = []; mother = []; drive = []; usb = []
screen = []; speaker = []; headphone = []


#extrayendo categoria de lifestore_products, posición [-2]
categoria = [item[-2] for item in lifestore_products]

categoria = list(dict.fromkeys(categoria))

#pos. var interes [:3]

for item in veces:
    if categoria[0] in item:
        processors.append(item[:3])
    elif categoria[1] in item:
        gpu.append(item[:3])
    elif categoria[2] in item:
        mother.append(item[:3])
    elif categoria[3] in item:
        drive.append(item[:3])
    elif categoria[4] in item:
        usb.append(item[:3])
    elif categoria[5] in item:
        screen.append(item[:3])
    elif categoria[6] in item:
        speaker.append(item[:3])
    elif categoria[7] in item:
        headphone.append(item[:3])

print(f'Menor venta de productos pertenecientes a la categoria: {categoria[0]}')

for i in range(5):
    print(f'ID: {processors[i][0]} \n Nombre: {processors[i][1]} \n No. Ventas: {processors[i][-1]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[1]}')

for i in range(5):
    print(f'ID: {gpu[i][0]} \n Nombre: {gpu[i][1]} \n No. Ventas: {gpu[i][2]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[2]}')

for i in range(5):
```

```
    print(f'ID: {mother[i][0]} \n Nombre: {mother[i][1]} \n No. Ventas: {mother[i][2]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[3]}')

for i in range(5):
    print(f'ID: {drive[i][0]} \n Nombre: {drive[i][1]} \n No. Ventas: {drive[i][2]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[4]}')

for i in range(2):
    print(f'ID: {usb[i][0]} \n Nombre: {usb[i][1]} \n No. Ventas: {usb[i][2]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[5]}')

for i in range(5):
    print(f'ID: {screen[i][0]} \n Nombre: {screen[i][1]} \n No. Ventas: {screen[i][2]}')



print(f'Menor venta de productos pertenecientes a la categoria: {categoria[6]}')

for i in range(5):
    print(f'ID: {speaker[i][0]} \n Nombre: {speaker[i][1]} \n No. Ventas: {speaker[i][2]}')


print(f'Menor venta de productos pertenecientes a la categoria: {categoria[7]}')

for i in range(5):
    print(f'ID: {headphone[i][0]} \n Nombre: {headphone[i][1]} \n No. Ventas: {headphone[i][2]}')
```

2.5. 10 productos con menores búsqueda por categoría

Similar a un proceso que se llevo a cabo en un apartado anterior, se implementa un ciclo for que recorre la lista **veces_busqueda**, observe:

```
print('10 Productos con Menores Busquedas: \n')
for i in range(10):
    print(f'ID: {veces_busqueda[i][0]} \n Nombre: {veces_busqueda[i][1]} \n No. busquedas: {veces_busqueda
    [i][2]}')
```

Finalmente, como ya se tenían en una lista de manera ascendente, por lo que para imprimir los primeros 10 productos se emplea **range()**.

2.6. productos con las mejores y peores reseñas

Para este apartado, hay que obtener la calificación final por producto, con ayuda de ciclo for, se emplean diversas listas para almacenar los datos de interés. Para la parte del promedio, se realiza la respectiva operación del total de puntuación sobre las veces que se ha vendido respectivamente todos y cada uno de ellos. Observe el código:

```
tempsum = 0 ; punt_final = []; punt_promedio = []

```

```
for i in range(0, len(lifestore_products)):
    for k in range(0, len(lifestore_sales)):
        if lifestore_sales[k][1] == veces[i][0]:
            tempsum += lifestore_sales[k][2]

    punt_final.append(tempsum)
    tempsum = 0

for producto in lifestore_products:
    for reseña in lifestore_sales:
        if producto[0] == reseña[1]:
            tempsum += reseña[2]

    punt_final.append(tempsum)
    tempsum = 0

# veces conformado por [id_prod, nombre, veces_vendido, categoria]

for reseña in lifestore_products:
    nido = []
    # reseñas de productos vendidos
    if veces[reseña[0]-1][2] > 0:
        punt_promedio.append(nido)
        for k in range(1):
            nido.append(reseña[0])
            nido.append(reseña[1])
            # veces proporcionando reseña al producto
            nido.append(veces[reseña[0]-1][2])
            # Calculo promedio
            nido.append(punt_final[reseña[0]-1]/veces[reseña[0]-1][2])

for reseña in veces:
    nido = []
    if reseña[2] > 0:
        nido.append(reseña[0])
        nido.append(reseña[1])
        # Ocasiones de reseña
        nido.append(reseña[2])
        #calculando promedio
        nido.append(punt_final[reseña[0]-1]/reseña[2])

def Sort(punt_promedio):
    punt_promedio.sort(key = lambda x: x[-1])
    return punt_promedio

punt_promedio = Sort(punt_promedio)

print('5 Productos con mejores reseñas \n')

for i in range(-1,-6,-1):
    print(f'ID: {punt_promedio[i][0]} \n Nombre: {punt_promedio[i][1]} \n Puntuación Promedio: {punt_
    promedio[i][-1]} \n Veces reseña: {punt_promedio[i][2]}')

print('5 Productos con peores reseñas \n')

for i in range(5):
    print(f'ID: {punt_promedio[i][0]} \n Nombre: {punt_promedio[i][1]} \n Puntuación Promedio: {punt_
    promedio[i][-1]} \n Veces reseña: {punt_promedio[i][2]}')
```

2.7. Total de ingresos y ventas

Se implementan listas, donde una contiene los meses del año y la segunda contiene las ventas ordenadas en orden cronológico. Se hace la suma de las ventas de los ingresos por mes mediante un ciclo for.

```
from datetime import datetime; import calendar
fecha = [venta[3] for venta in lifestore_sales]
# formato fecha en orden
fecha.sort(key = lambda fecha: datetime.strptime(fecha, '%d/%m/%Y'))

mes = calendar.month_name[1:]

art_reembolso = []; total_reembolsos = 0; ticket_promedio = []; total_ventas = 0
meses_ventas = [0]*12; ganancia_mensual = [0]*12

venta_producto = [venta[1] for venta in lifestore_sales]

for i in range(0, len(lifestore_sales)):
    # apartado para verificar reembolso
    if int(lifestore_sales[i][-1]) == 0:
        total_ventas += lifestore_products[venta_producto[i]][2]
        if '/01/' in fecha[i]:
            ganancia_mensual[0] += lifestore_products[venta_producto[i]][2]
            meses_ventas[0] += 1
        elif '/02/' in fecha[i]:
            ganancia_mensual[1] += lifestore_products[venta_producto[i]][2]
            meses_ventas[1] += 1
        elif '/03/' in fecha[i]:
            ganancia_mensual[2] += lifestore_products[venta_producto[i]][2]
            meses_ventas[2] += 1
        elif '/04/' in fecha[i]:
            ganancia_mensual[3] += lifestore_products[venta_producto[i]][2]
            meses_ventas[3] += 1
        elif '/05/' in fecha[i]:
            ganancia_mensual[4] += lifestore_products[venta_producto[i]][2]
            meses_ventas[4] += 1
        elif '/06/' in fecha[i]:
            ganancia_mensual[5] += lifestore_products[venta_producto[i]][2]
            meses_ventas[5] += 1
        elif '/07/' in fecha[i]:
            ganancia_mensual[6] += lifestore_products[venta_producto[i]][2]
            meses_ventas[6] += 1
        elif '/08/' in fecha[i]:
            ganancia_mensual[7] += lifestore_products[venta_producto[i]][2]
            meses_ventas[7] += 1
        elif '/09/' in fecha[i]:
            ganancia_mensual[8] += lifestore_products[venta_producto[i]][2]
            meses_ventas[8] += 1
        elif '/10/' in fecha[i]:
            ganancia_mensual[9] += lifestore_products[venta_producto[i]][2]
            meses_ventas[9] += 1
        elif '/11/' in fecha[i]:
            ganancia_mensual[10] += lifestore_products[venta_producto[i]][2]
            meses_ventas[10] += 1
        elif '/12/' in fecha[i]:
            ganancia_mensual[11] += lifestore_products[venta_producto[i]][2]
            meses_ventas[11] += 1
    else:
        #ID producto
        art_reembolso.append(venta_producto[i])
        total_reembolsos += lifestore_products[venta_producto[i]][2]

for i in range(12):
    if meses_ventas[i] > 0:
        #ticket promedio
        ticket_promedio.append(ganancia_mensual[i]/meses_ventas[i])
```

```
    else:
        ticket_promedio.append(0)

ventas_mensuales = [list(l) for l in zip(mes, ganancia_mensual, meses_ventas, ticket_promedio)]

def Sort(ventas_mensuales):
    ventas_mensuales.sort(key = lambda x: x[1])
    return ventas_mensuales
ventas_mensuales = Sort(ventas_mensuales)

print('Meses con mayores ganancias')

for i in range(-1,-5,-1):
    print(f'Mes: {ventas_mensuales[i][0]} \n Ganancia: {"${:,.2f}".format(ventas_mensuales[i][1])} \n
    Ventas: {ventas_mensuales[i][2]} \n Promedio: {ventas_mensuales[i][-1]}')


def Sort(ventas_mensuales):
    ventas_mensuales.sort(key = lambda x: x[2])
    return ventas_mensuales
ventas_mensuales = Sort(ventas_mensuales)


print('Mayores ventas por mes')

for i in range(-1,-5,-1):
    print(f'Mes: {ventas_mensuales[i][0]} \n Ganancia: {"${:,.2f}".format(ventas_mensuales[i][1])} \n
    Ventas: {ventas_mensuales[i][2]} \n Promedio: {ventas_mensuales[i][-1]}')

def Sort(ventas_mensuales):
    ventas_mensuales.sort(key = lambda x: x[-1])
    return ventas_mensuales
ventas_mensuales = Sort(ventas_mensuales)

print('Ticket más alto por mes')

for i in range(-1,-5,-1):
    print(f'Mes: {ventas_mensuales[i][0]} \n Ganancia: {"${:,.2f}".format(ventas_mensuales[i][1])} \n
    Ventas: {ventas_mensuales[i][2]} \n Promedio: {ventas_mensuales[i][-1]}')

print(f'Ganancia total entre {fecha[0]} y {fecha[-1]}: {"${:,.2f}".format(total_ventas)}')
```

Capítulo 3

Resultados

Los 5 productos que tienen más ventas, son los que se observan en Figura 3.1

```
Datos sobre los 5 productos con mayores ventas:

ID: 54
 Nombre: SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm
 No. Ventas: 50
ID: 3
 Nombre: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth
 No. Ventas: 42
ID: 5
 Nombre: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake)
 No. Ventas: 20
ID: 42
 Nombre: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD
 No. Ventas: 18
ID: 57
 Nombre: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm
 No. Ventas: 15
```

Figura 3.1

Los 10 productos que tienen más búsquedas, son los que se observan en Figura 3.2

```
10 Productos con mayores busqueda:
ID: 54
 Nombre: SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm No. busquedas: 263

ID: 57
 Nombre: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm No. busquedas: 107

ID: 29
 Nombre: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD No. busquedas: 60

ID: 3
 Nombre: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth No. busquedas: 55

ID: 4
 Nombre: Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire No. busquedas: 41

ID: 85
 Nombre: Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul No. busquedas: 35

ID: 67
 Nombre: TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro No. busquedas: 32

ID: 7
 Nombre: Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) No. busquedas: 31

ID: 47
 Nombre: SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 No. busquedas: 30

ID: 5
 Nombre: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) No. busquedas: 30
```

Figura 3.2

Los 10 productos que tienen menos búsquedas, son los que se observan en Figura 3.3

```
10 Productos con Menores Busquedas:

ID: 14
 Nombre: Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0
 No. busquedas: 0
ID: 16
 Nombre: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0
 No. busquedas: 0
ID: 19
 Nombre: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile, 4GB 128-bit GDDR5, PCI Express 3.0 x16
 No. busquedas: 0
ID: 20
 Nombre: Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060 SUPER WINDFORCE OC, 8 GB 256 bit GDDR6, PCI Express x16 3.0
 No. busquedas: 0
ID: 23
 Nombre: Tarjeta de Video MSI Radeon X1550, 128MB 64 bit GDDR2, PCI Express x16
 No. busquedas: 0
ID: 24
 Nombre: Tarjeta de Video PNY NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0
 No. busquedas: 0
ID: 30
 Nombre: Tarjeta Madre AORUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel
 No. busquedas: 0
ID: 32
 Nombre: Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel
 No. busquedas: 0
ID: 33
 Nombre: Tarjeta Madre ASUS ATX PRIME Z390-A, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel
 No. busquedas: 0
ID: 34
 Nombre: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD
 No. busquedas: 0
```

Figura 3.3

Los 5 productos que tienen mejores reseñas, son los que se observan en Figura 3.4

```
5 Productos con mejores reseñas

ID: 85
 Nombre: Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul
 Puntuación Promedio: 5.0
 Veces reseña: 7
ID: 81
 Nombre: Ghia Bocina Portátil BX900, Bluetooth, Inalámbrico, 2.1 Canales, 34W, USB, Negro - Resistente al Agua
 Puntuación Promedio: 5.0
 Veces reseña: 4
ID: 79
 Nombre: Naceb Bocina Portátil NA-0301, Bluetooth, Inalámbrico, USB 2.0, Rojo
 Puntuación Promedio: 5.0
 Veces reseña: 3
ID: 78
 Nombre: Ghia Bocina Portátil BX300, Bluetooth, Inalámbrico, 40W RMS, USB, Rojo - Resistente al Agua
 Puntuación Promedio: 5.0
 Veces reseña: 3
ID: 77
 Nombre: Verbatim Bocina Portátil Mini, Bluetooth, Inalámbrico, 3W RMS, USB, Blanco
 Puntuación Promedio: 5.0
 Veces reseña: 3
```

Figura 3.4

Los 5 productos que tienen peores reseñas, son los que se observan en Figura 3.5

Los meses con mayores ganancias y mayores ventas por mes, así como el ticket más alto por mes son los siguientes, se observan en Figura 3.6

```
5 Productos con peores reseñas

ID: 57
 Nombre: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm
 Puntuación Promedio: 1.0
 Veces reseña: 1
ID: 61
 Nombre: Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16
 Puntuación Promedio: 1.0
 Veces reseña: 1
ID: 83
 Nombre: Ghia Bocina Portátil BX500, Bluetooth, Inalámbrico, 10W RMS, USB, Gris
 Puntuación Promedio: 1.8333333333333333
 Veces reseña: 6
ID: 62
 Nombre: Makena Smart TV LED 32S2 32'', HD, Widescreen, Gris
 Puntuación Promedio: 2.0
 Veces reseña: 1
ID: 68
 Nombre: Makena Smart TV LED 40S2 40'', Full HD, Widescreen, Negro
 Puntuación Promedio: 3.0
 Veces reseña: 1
```

Figura 3.5

```
Meses con mayores ganancias
Mes: April
 Ganancia: $304,578.00
 Ventas: 72
 Promedio: 4230.25
Mes: February
 Ganancia: $302,409.00
 Ventas: 41
 Promedio: 7375.829268292683
Mes: March
 Ganancia: $189,473.00
 Ventas: 47
 Promedio: 4031.340425531915
Mes: May
 Ganancia: $153,965.00
 Ventas: 35
 Promedio: 4399.0
```

(a) Meses con mayores ganancias

```
Mayores ventas por mes
Mes: April
 Ganancia: $304,578.00
 Ventas: 72
 Promedio: 4230.25
Mes: January
 Ganancia: $127,428.00
 Ventas: 52
 Promedio: 2450.5384615384614
Mes: March
 Ganancia: $189,473.00
 Ventas: 47
 Promedio: 4031.340425531915
Mes: February
 Ganancia: $302,409.00
 Ventas: 41
 Promedio: 7375.829268292683
```

(b) Mayores ventas por mes

Figura 3.6

Capítulo 4

Conclusión

Lifestore es una tienda que vende una diversa variedad de productos, de los cuales, son un puñado los artículos que son top sellers. Por lo que habría que reducir el tipo de productos que se venden, lo cual puede incurrir en tener ahorros en caso de que se esté incurriendo en gastos de almacén o de esa índole.

Capítulo 5

Repositorio GitHub

El repositorio se encuentra en este link, que cuenta con los archivos empleados para dar solución al problema.