



UNIVERSIDAD
AUTÓNOMA
METROPOLITANA
Unidad Cuajimalpa

Tarea 4

Sistemas Operativos

Jorge Angel Juárez Vázquez
2213026247

Profesor: Jose Netz Romero Duran
12 September 2024

Tarea 4 - Memoria compartida

Nota

Negocio: WoldFork

Lenguaje de programacion: Python 

Archivo

Para la primera fase de la tarea se hizo uso de un archivo `.csv` que funcione como tipo base de datos, a los registros de los productos, se le agregó nuevo campo el cual será el de `stock` que representa la cantidad de artículos en existencia, el contenido del archivo se ve de la siguiente forma:

```
id_reg,name,category,weight,price,expiry,stock
1,Nuggets de pollo,Aves,0.8,120,2024-10-30,27
2,Pan de ajo grande,Panaderia,0.3,50,2024-10-15,29
3,Yogurt natural nuez,Lacteos,0.5,25,2024-10-30,40
4,Filete de res Angus,Carne,1.2,250,2024-12-31,20
5,Helado de vainilla,Helados,1.5,100,2024-09-30,10
6,Bolsa de vegetales,Vegetales,0.5,80,2024-12-10,12
7,Galletas de nuez,Galletas,0.2,20,2024-11-15,21
8,Camarones jumbo,Mariscos,0.5,180,2024-11-15,5
9,Salsa de tomate,Salsas,0.5,30,2024-12-31,12
10,Pan de ajo chico,Panaderia,0.1,20,2024-10-15,30
```

El cual va se ha leído y cada registro va a ser representado por un objeto de tipo `Register`, para que sea más fácil la manipulación de los mismos en las siguientes fases del desarrollo, aparte de la clase para representar cada registro, se crearon funciones para la lectura del archivo `.csv`, escritura en el archivo `.csv`, para crear una fila de una tabla con la información del registro (tabla mostrada en la interfaz del sistema), y para buscar un registro en una lista de objetos de tipo `Register`; esta clase y funciones fueron codificadas de la siguiente forma:

`core.py`

```
from typing import List, Dict, Union, Optional
from csv import DictReader, DictWriter
import flet as ft

class Register:
    def __init__(self, id_reg: int, name: str, category: str, weight: int,
                 price: float, expiry: str, stock: int, is_locked: Optional[bool] = None) →
```

None:

```
    self.id_reg = id_reg
    self.name = name
    self.category = category
    self.weight = weight
    self.price = price
    self.expiry = expiry
    self.stock = stock
    self.is_locked = is_locked

def getattribs_dict(self) → Dict:
    return {
        "id_reg": self.id_reg, "name": self.name, "category": self.category,
        "weight": self.weight, "price": self.price,
        "expiry": self.expiry, "stock": self.stock
    }

def __str__(self) → str:
    return f'Nombre: {self.name}\nCategoria: {self.category}\n\nPeso (kg): {self.weight}\nPrecio (MXN): $ {self.price}\n\nCaducidad: {self.expiry}\nStock: {self.stock}'

def read_csv(file_path: str) → List[Register]:
    """Leer datos desde un archivo CSV y devolver una lista de registros."""
    try:
        with open(file_path, mode='r', encoding='utf-8') as file:
            csv_reader = DictReader(file)
            return [Register(**row) for row in csv_reader]
    except FileNotFoundError:
        print(f"El archivo {file_path} no se encuentra.")
        return []

def write_csv(file_path: str, registers: List[Register]) → None:
    """Escribir los registros actualizados en un archivo CSV."""
    with open(file_path, mode='w', newline='', encoding='utf-8') as csvfile:
        fieldnames = ['id_reg', 'name', 'category', 'weight',
                      'price', 'expiry', 'stock']

        writer = DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()
        for r in registers:
            writer.writerow(r.getattribs_dict())

def to_DataTable(registers: List[Register]) → List[ft.DataRow]:
    """Convertir los registros en filas para la tabla de datos en Flet."""
    return [
        ftDataRow(
```

```

        cells=[
            ft.DataCell(ft.Text(str(rr.id_reg))),
            ft.DataCell(ft.Text(rr.name)),
            ft.DataCell(ft.Text(rr.category)),
            ft.DataCell(ft.Text(str(rr.weight))),
            ft.DataCell(ft.Text(f"${rr.price}")),
            ft.DataCell(ft.Text(rr.expiry)),
            ft.DataCell(ft.Text(str(rr.stock)))
        ]
    ) for rr in registers
]

def search_register(registers: List[Register], id_reg: int) → Union[Register, bool]:
    """Buscar un registro por su ID. Devolver el registro o False si no existe."""
    return False if id_reg > len(registers) or id_reg ≤ 0 else
    registers[id_reg - 1]

```

Interfaz gráfica

Para la creación de la interfaz gráfica del sistema se ocupó **felt** que es un framework que permite crear aplicaciones web utilizando Python, sin necesidad de escribir código HTML, CSS o JavaScript. Utiliza un modelo similar a Flutter (de Google), donde puedes crear interfaces de usuario mediante la programación de componentes en Python. Es útil para construir aplicaciones interactivas y modernas, con soporte para widgets, notificaciones y funcionalidades más avanzadas de forma sencilla.

El código que se desarrolló fue el siguiente:

app.py

```

import flet as ft
from core import read_csv, write_csv, to_DataTable, search_register

# Ruta del archivo CSV
DATA_FILE = "./data/wildfork.csv"

# Leer los datos iniciales
data = read_csv(DATA_FILE)
rows = to_DataTable(data)

def alert(error: str) → ft.AlertDialog:
    """Mostrar un diálogo de alerta con un mensaje de error."""
    return ft.AlertDialog(title=ft.Text(error))

def main(page: ft.Page):

```

```

# Configuración inicial de la página
page.title = "Tarea 4 - Jorge Juarez"
page.theme_mode = ft.ThemeMode.LIGHT
page.scroll = ft.ScrollMode.AUTO
page.vertical_alignment = ft.MainAxisAlignment.CENTER
page.horizontal_alignment = ft.CrossAxisAlignment.CENTER

# AppBar y título
title_appbar = ft.Text("Productos disponibles",
theme_style=ft.TextThemeStyle.HEADLINE_SMALL)
appbar = ft.AppBar(
    title=title_appbar,
    center_title=True,
    leading=ft.Image(src="images/wild-fork-logo.png"),
    leading_width=180
)

# Componentes de la interfaz de usuario
id_register = ft.Text("", theme_style=ft.TextThemeStyle.BODY_LARGE)
id_search = ft.TextField(label="ID del producto", icon=ft.icons.GRID_3X3)
data_register = ft.Text("", weight=ft.FontWeight.BOLD)
# Función para buscar un registro
def go_register(e):
    try:
        product_id = int(id_search.value)
        reg_info = search_register(data, product_id)
        if reg_info:
            mostrar_detalles_producto(reg_info)
        else:
            page.open(alert("No se encuentra un producto con el ID
ingresado o el producto está agotado"))
    except ValueError:
        page.open(alert("Valor ingresado incorrectamente"))
    except Exception as ex:
        print(f"Error inesperado: {ex}")

def mostrar_detalles_producto(reg_info):
    """Muestra los detalles del producto seleccionado."""
    title_appbar.value = "Compra de producto"
    id_register.value = f"Detalles del producto: {id_search.value}"
    data_register.value = str(reg_info)
    page.bottom_appbar = register_bottombar
    page.scroll = None
    page.controls.pop()
    page.add(register)
    page.update()

# Barra inferior en la página de inicio
home_bottombar = ft.BottomAppBar(

```

```

        content=ft.Row(
            controls=[
                id_search,
                ft.FilledButton(text="Detalles",
icon=ft.icons.SAVED_SEARCH_ROUNDED, on_click=go_register)
            ],
            vertical_alignment=ft.CrossAxisAlignment.CENTER,
            alignment=ft.MainAxisAlignment.SPACE_EVENLY
        )
    )

# Función para regresar a la página principal
def go_home(e):
    actualizar_datos()
    title_appbar.value = "Productos disponibles"
    page.scroll = ft.ScrollMode.AUTO
    page.bottom_appbar = home_bottombar
    page.controls.pop()
    page.add(data_table)
    page.update()

# Función para actualizar los datos de la tabla
def actualizar_datos():
    global data, rows
    data = read_csv(DATA_FILE)
    rows = to_DataTable(data)
    data_table.rows = rows

# Función para procesar la compra de un producto
def shop(e):
    try:
        product_id = int(id_search.value) - 1
        if int(data[product_id].stock) > 0:
            data[product_id].stock = int(data[product_id].stock) - 1
            write_csv(DATA_FILE, data)
            page.open(ft.AlertDialog(title=ft.Text(f"Producto
'{data[product_id].name}' comprado con éxito"), on_dismiss=go_home))
        else:
            page.open(alert("No hay suficiente stock del producto."))
    except Exception as ex:
        page.open(alert(f"Error en la compra: {ex}"))

# Barra inferior en la página de detalles
register_bottombar = ft.BottomAppBar(
    content=ft.Row(
        controls=[
            ft.FilledButton(text="Comprar",
icon=ft.icons.SHOPPING_BAG_OUTLINED, on_click=shop),
            ft.FilledButton(text="Regresar", icon=ft.icons.ARROW_BACK,

```

```

on_click=go_home)
    ],
    vertical_alignment=ft.CrossAxisAlignment.CENTER,
    alignment=ft.MainAxisAlignment.SPACE_EVENLY
)
)

# Tabla de productos
data_table = ft.DataTable(
    columns=[
        ft DataColumn(ft.Text("ID"), numeric=True),
        ft DataColumn(ft.Text("Nombre")),
        ft DataColumn(ft.Text("Categoria")),
        ft DataColumn(ft.Text("Peso (kg)"), numeric=True),
        ft DataColumn(ft.Text("Precio (MXN)"), numeric=True),
        ft DataColumn(ft.Text("Caducidad")),
        ft DataColumn(ft.Text("Stock"), numeric=True),
    ],
    rows=rows
)

# Contenedor de detalles del registro
register = ft.Container(
    content=ft.Column(
        controls=[
            id_register,
            ft.Container(
                content=data_register,
                padding=ft.padding.all(5),
                border=ft.border.all(2.5, "0"),
                border_radius=ft.border_radius.all(15)
            )
        ]
    )
)

# Configuración inicial de la página
page.appbar = appBar
page.bottom_appbar = home_bottombar
page.add(data_table)

ft.app(
    main,
    assets_dir="assets"
)

```

¡Este código permite crear la siguiente interfaz con las siguientes vistas:

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	26
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5
9	Salsa de tomate	Salsas	0.5	\$30	2024-12-31	12

ID del producto

🔍 Detalles



Memoria compartida

El desarrollo de memoria compartida en sistemas concurrentes permite que múltiples procesos accedan y modifiquen datos de manera simultánea, utilizando una región de memoria común. En Python, esto se logra mediante el uso de `multiprocessing.Manager()`, que facilita la creación de estructuras compartidas como listas o diccionarios. Para garantizar la consistencia de los datos y evitar condiciones de carrera, se emplean mecanismos de sincronización como los locks, que aseguran que solo un proceso pueda modificar una sección crítica de la memoria a la vez. Esto es fundamental en aplicaciones donde se requiere coordinar el acceso a recursos compartidos entre procesos concurrentes.

Para lograr esto, entonces se tuvo que modificar el script de `app.py` para que permita trabajar de manera concurrente y con la lista de registros como lista compartida en las dos instancias del programa. Dando como resultado el siguiente:

`main.py`

```
from multiprocessing import Manager, Process
from typing import List
import flet as ft
from core import Register, read_csv, write_csv, to_DataTable
```

```

# Ruta del archivo CSV
DATA_FILE = "./data/wildfork.csv"

# Crear memoria compartida para los productos con todos los atributos
def create_shared_product_list():
    data = read_csv(DATA_FILE)
    # Incluimos todos los atributos de cada registro
    return [
        {
            'id_reg': reg.id_reg,
            'name': reg.name,
            'category': reg.category,
            'weight': reg.weight,
            'price': reg.price,
            'expiry': reg.expiry,
            'stock': reg.stock,
            'is_locked': False # Bandera para indicar si está bloqueado
        }
        for reg in data
    ]

# Función para acceder a los detalles del producto con bloqueo
def access_product(products: List, lock, product_id):
    with lock:
        product = products[product_id]
        if product['is_locked']:
            return False # El producto está bloqueado por otra instancia
        else:
            lock_state = product.copy()
            lock_state['is_locked'] = True
            products[product_id] = lock_state # Bloquea el producto
            return product # Devuelve el producto para mostrar los detalles

# Función para desbloquear un producto después de la compra o regreso
def unlock_product(products, lock, product_id):
    with lock:
        product = products[product_id]
        lock_state = product.copy()
        lock_state['is_locked'] = False
        products[product_id] = lock_state # Desbloquea el producto

def alert(error: str) → ft.AlertDialog:
    """Mostrar un diálogo de alerta con un mensaje de error."""
    return ft.AlertDialog(title=ft.Text(error))

def main(page: ft.Page, products: List, lock):
    # Configuración inicial de la página
    page.title = "Tarea 4 - Jorge Juarez"

```

```
page.theme_mode = ft.ThemeMode.LIGHT
page.scroll = ft.ScrollMode.AUTO
page.vertical_alignment = ft.MainAxisAlignment.CENTER
page.horizontal_alignment = ft.CrossAxisAlignment.CENTER

# AppBar y título
title_appbar = ft.Text("Productos disponibles",
theme_style=ft.TextThemeStyle.HEADLINE_SMALL)
appbar = ft.AppBar(
    title=title_appbar,
    center_title=True,
    leading=ft.Image(src="images/wild-fork-logo.png"),
    leading_width=180
)

# Componentes de la interfaz de usuario
id_register = ft.Text("", theme_style=ft.TextThemeStyle.BODY_LARGE)
id_search = ft.TextField(label="ID del producto", icon=ft.icons.GRID_3X3)
data_register = ft.Text("", weight=ft.FontWeight.BOLD)

# Función para buscar un registro
def go_register(e):
    try:
        product_id = int(id_search.value) - 1
        # Intentamos acceder al producto con el bloqueo
        # print("Estado del bloqueo: ", products[product_id]['is_locked'])
        product = access_product(products, lock, product_id)
        # print("Estado del bloqueo despues: ", products[product_id]
['is_locked'])
        if product:
            mostrar_detalles_producto(product, product_id)
        else:
            page.open(alert("El producto está siendo accedido por otra
instancia"))
    except ValueError:
        page.open(alert("Valor ingresado incorrectamente"))
    except IndexError:
        page.open(alert(f"Producto con el ID: {id_search.value} no
encontrado"))
    except Exception as ex:
        page.open(alert(f"Error inesperado: {ex}"))

def mostrar_detalles_producto(product, product_id):
    """Muestra los detalles del producto seleccionado."""
    title_appbar.value = "Compra de producto"
    id_register.value = f"Detalles del producto: {product['id_reg']}"
    data_register.value = f"""
Nombre: {product['name']}
Categoría: {product['category']}
```

```

        Peso: {product['weight']} kg
        Precio: {product['price']} MXN
        Caducidad: {product['expiry']}
        Stock: {product['stock']}
    """
    page.bottom_appbar = register_bottombar(product_id)
    page.scroll = None
    page.controls.pop()
    page.add(register)
    page.update()

# Barra inferior en la página de inicio
home_bottombar = ft.BottomAppBar(
    content=ft.Row(
        controls=[
            id_search,
            ft.FilledButton(text="Detalles",
icon=ft.icons.SAVED_SEARCH_ROUNDED, on_click=go_register),
        ],
        vertical_alignment=ft.CrossAxisAlignment.CENTER,
        alignment=ft.MainAxisAlignment.SPACE_EVENLY
    )
)

# Función para regresar a la página principal
def go_home(product_id=None):
    if product_id is not None:
        unlock_product(products, lock, product_id) # Desbloqueamos el
producto
        actualizar_datos()
        title_appbar.value = "Productos disponibles"
        page.scroll = ft.ScrollMode.AUTO
        page.bottom_appbar = home_bottombar
        page.controls.pop()
        page.add(data_table)
        page.update()

# Función para actualizar los datos de la tabla
def actualizar_datos():
    """Actualiza la tabla de productos con los datos en memoria
compartida."""
    # Convertimos cada diccionario de producto a un objeto Register
    data_table.rows = to_DataTable([Register(**p) for p in products])

# Función para procesar la compra de un producto
def shop(product_id):
    with lock:
        if int(products[product_id]['stock']) > 0:
            product_copy = products[product_id].copy()

```

```

        product_copy['stock'] = int(product_copy['stock']) - 1
        products[product_id] = product_copy
        # Actualizamos el archivo CSV
        write_csv(DATA_FILE, [Register(**p) for p in products])
        page.open(ft.AlertDialog(title=ft.Text(f"Producto
'{products[product_id]['name']}' comprado con éxito"), on_dismiss=lambda e:
go_home(product_id)))
    else:
        page.open(alert("No hay suficiente stock del producto."))
    # Barra inferior en la página de detalles
    def register_bottombar(product_id):
        return ft.BottomAppBar(
            content=ft.Row(
                controls=[
                    ft.FilledButton(text="Comprar",
icon=ft.icons.SHOPPING_BAG_OUTLINED, on_click=lambda e: shop(product_id)),
                    ft.FilledButton(text="Regresar", icon=ft.icons.ARROW_BACK,
on_click=lambda e: go_home(product_id))
                ],
                vertical_alignment=ft.CrossAxisAlignment.CENTER,
                alignment=ft.MainAxisAlignment.SPACE_EVENLY
            )
        )
    # Tabla de productos
    data_table = ft.DataTable(
        columns=[
            ft DataColumn(ft.Text("ID"), numeric=True),
            ft DataColumn(ft.Text("Nombre")),
            ft DataColumn(ft.Text("Categoría")),
            ft DataColumn(ft.Text("Peso (kg)"), numeric=True),
            ft DataColumn(ft.Text("Precio (MXN)"), numeric=True),
            ft DataColumn(ft.Text("Caducidad")),
            ft DataColumn(ft.Text("Stock"), numeric=True)
        ]
    )
    # Contenedor de detalles del registro
    register = ft.Container(
        content=ft.Column(
            controls=[
                id_register,
                ft.Container(
                    content=data_register,
                    padding=ft.padding.all(5),
                    border=ft.border.all(2.5, "0"),
                    border_radius=ft.border_radius.all(15)
                )
            ]
        )
    )

```

```
        ]
    )
)

# Configuración inicial de la página
actualizar_datos()
page.appbar = appBar
page.bottom_appbar = home_bottombar
page.add(data_table)

def run_flet_app(products, lock):
    ft.app(target=lambda page: main(page, products, lock), assets_dir="assets")

if __name__ == "__main__":
    manager = Manager()
    products = manager.list(create_shared_product_list())
    lock = manager.Lock()

    # Crear dos procesos que corran la aplicación Flet
    p1 = Process(target=run_flet_app, args=(products, lock))
    p2 = Process(target=run_flet_app, args=(products, lock))

    # Iniciar ambos procesos
    p1.start()
    p2.start()

    # Esperar a que ambos procesos terminen
    p1.join()
    p2.join()
```

Funcionamiento (pantallazos)

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:19

Tarea 4 - Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py...

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

ID del producto Detalles

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:20

Tarea 4 - Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py...

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Valor ingresado incorrectamente ID del producto Detalles

error

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:21

es

(Tarea4) jorgejrzz@fedora: /Documentos/Tarea4\$ python main.py
package:media_kit_linux registered.
package:media_kit_linux registered.

Tarea 4 - Jorge Juarez

Wild Fork Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-11-15	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Producto con el ID: 10000000 no encontrado

ID del producto Detalles

Detalles

fedora

The screenshot shows a Fedora OS desktop environment. In the top-left corner, there's a terminal window titled "jorgejrzz@fedora:~/Documentos/Tarea4" displaying command-line output related to Python and package registration. The main workspace features two overlapping windows from the "Wild Fork" application. The top window is titled "Productos disponibles" and lists various food items with columns for ID, Nombre, Categoría, Peso (kg), Precio (MXN), Caducidad, and Stock. The bottom window is also titled "Productos disponibles" and includes a search bar with the placeholder "ID del producto" and a value "1000000", along with a "Detalles" button. The desktop background is a blue and green abstract design.

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panaderia	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones iumbo	Mariscos	0.5	\$180	2024-11-15	5

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:23

Tarea 4 - Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media_kit.libs_linux registered.

package:media_kit.libs_linux registered.

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Detalles del producto: 8

Nombre: Camarones jumbo
Categoría: Mariscos
Peso: 0.5 kg
Precio: 180 MXN
Caducidad: 2024-11-15
Stock: 5

Comprar Regresar

ID del producto: 1000000

Detalles

CTRL DERECHA

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:23

Tarea 4 - Jorge Juarez

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media_kit.libs_linux registered.

package:media_kit.libs_linux registered.

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filete de res Angus	Carne	1.2	\$250	2024-12-31	20
5	Helado de vainilla	Helados	1.5	\$100	2024-09-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

Detalles del producto: 8

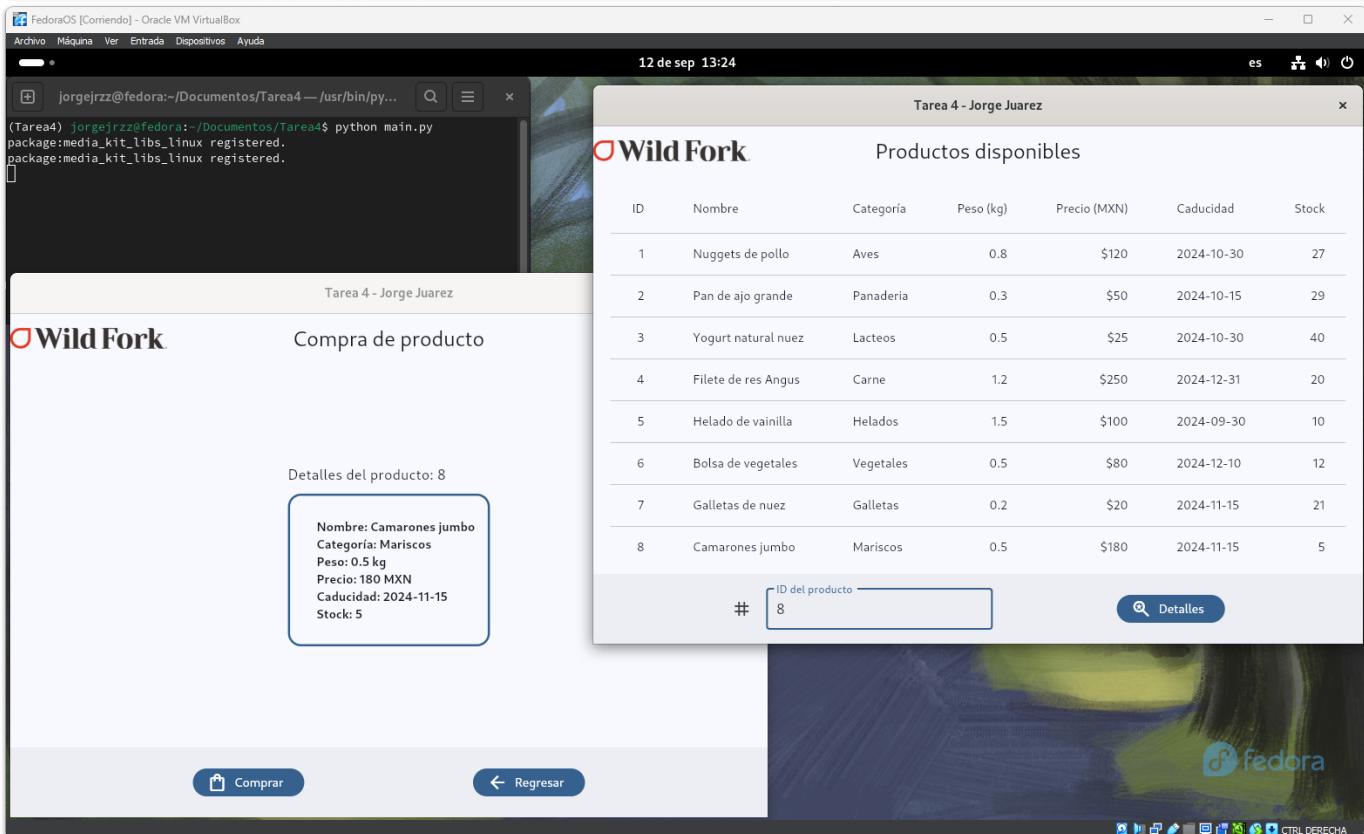
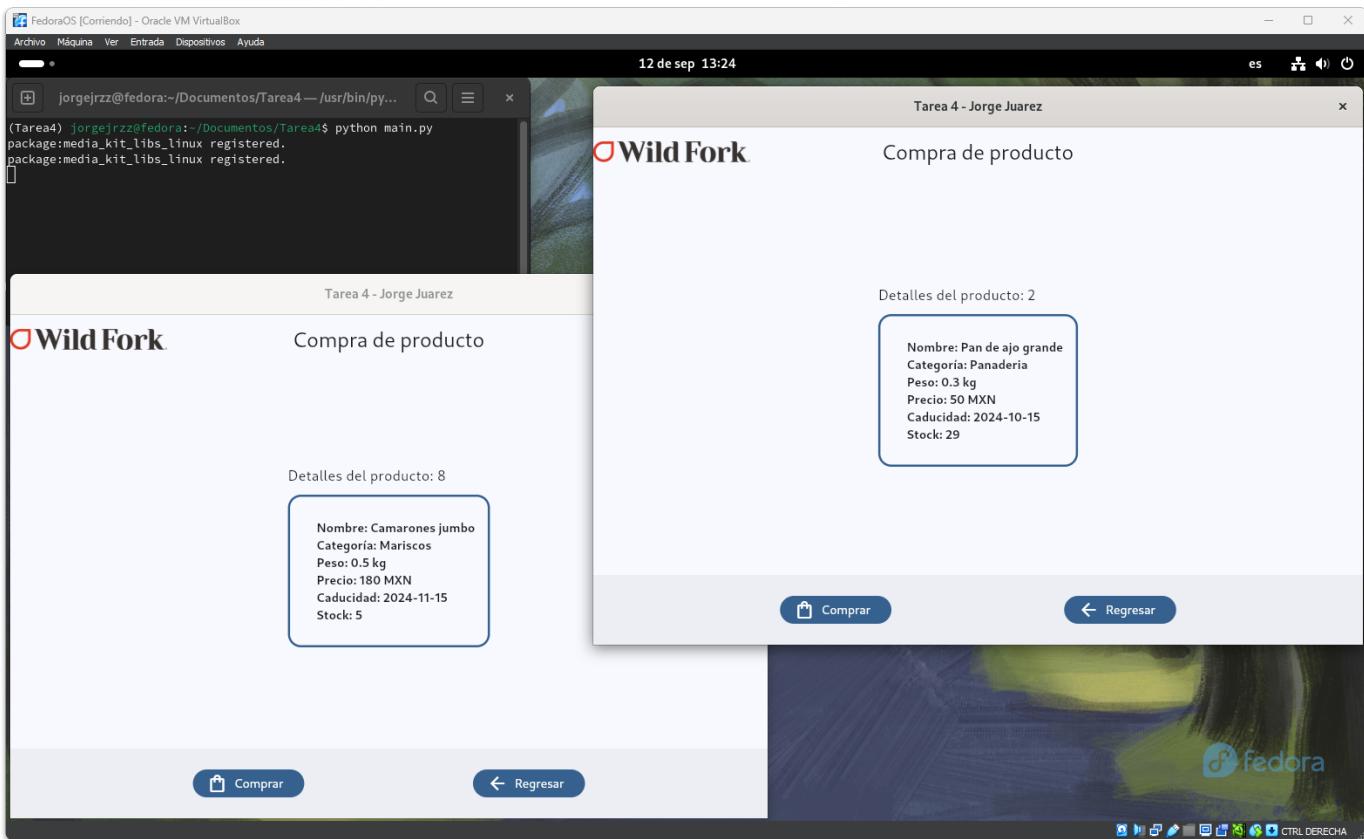
Nombre: Camarones jumbo
Categoría: Mariscos
Peso: 0.5 kg
Precio: 180 MXN
Caducidad: 2024-11-15
Stock: 5

Comprar Regresar

ID del producto: 2

Detalles

CTRL DERECHA



FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:24

jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media_kit.libs_linux registered.

package:media_kit.libs_linux registered.

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filetes de pescado					20
5	Helado vainilla				2024-10-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

El producto está siendo accedido por otra instancia

ID del producto: 8

Detalles

Nombre: Camarones jumbo
Categoría: Mariscos
Peso: 0.5 kg
Precio: 180 MXN
Caducidad: 2024-11-15
Stock: 5

Comprar Regresar

Tarea 4 - Jorge Juarez

Wild Fork

Compra de producto

FedoraOS [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

12 de sep 13:25

jorgejrzz@fedora:~/Documentos/Tarea4 — /usr/bin/py... x

(Tarea4) jorgejrzz@fedora:~/Documentos/Tarea4\$ python main.py

package:media_kit.libs_linux registered.

package:media_kit.libs_linux registered.

Tarea 4 - Jorge Juarez

Wild Fork

Productos disponibles

ID	Nombre	Categoría	Peso (kg)	Precio (MXN)	Caducidad	Stock
1	Nuggets de pollo	Aves	0.8	\$120	2024-10-30	27
2	Pan de ajo grande	Panadería	0.3	\$50	2024-10-15	29
3	Yogurt natural nuez	Lacteos	0.5	\$25	2024-10-30	40
4	Filetes de pescado					20
5	Helado vainilla				2024-10-30	10
6	Bolsa de vegetales	Vegetales	0.5	\$80	2024-12-10	12
7	Galletas de nuez	Galletas	0.2	\$20	2024-11-15	21
8	Camarones jumbo	Mariscos	0.5	\$180	2024-11-15	5

El producto está siendo accedido por otra instancia

ID del producto: 8

Detalles

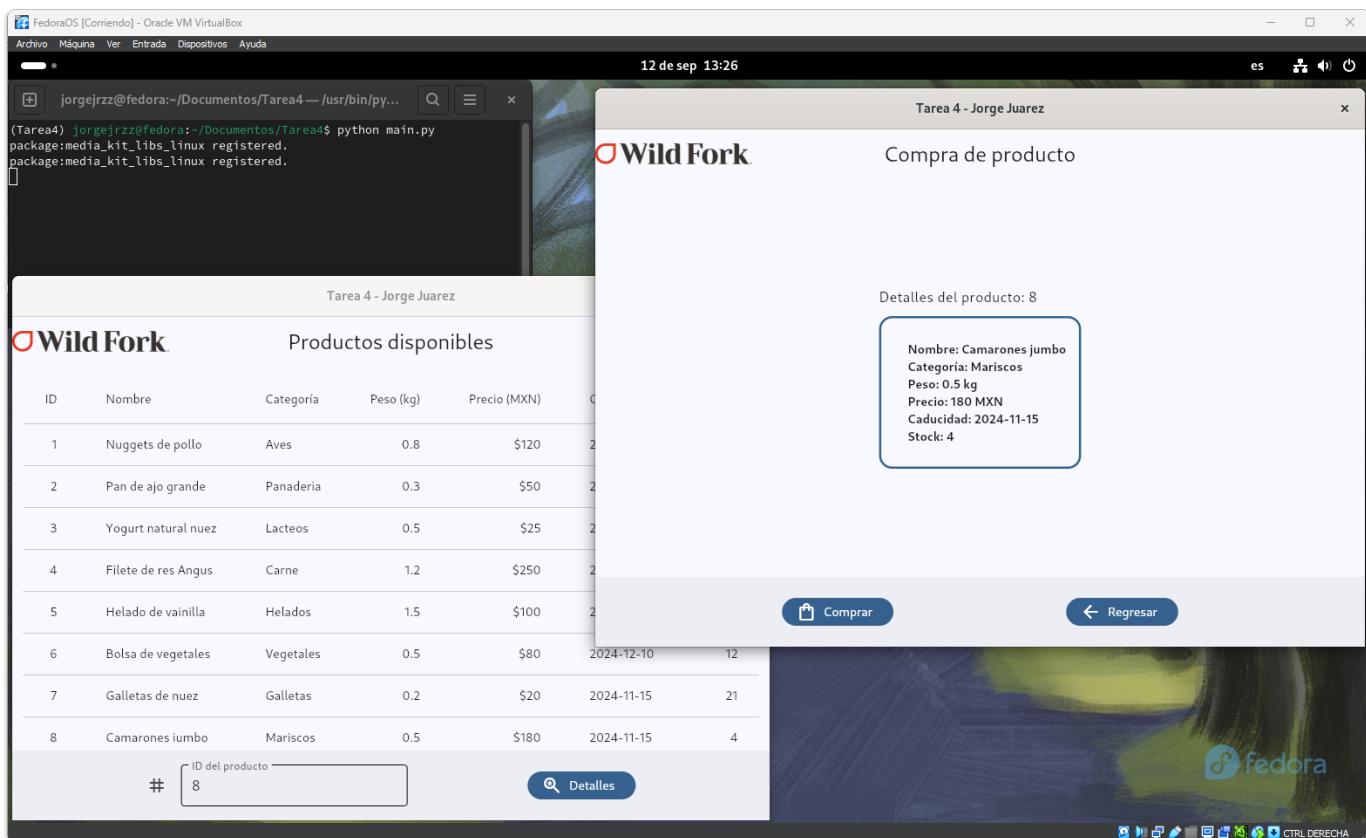
Nombre: Camarones jumbo
Categoría: Mariscos
Peso: 0.5 kg
Precio: 180 MXN
Caducidad: 2024-11-15
Stock: 5

Comprar Regresar

Tarea 4 - Jorge Juarez

Wild Fork

Compra de producto



Conclusión

En esta actividad se implementó una aplicación de gestión de productos utilizando *Flet* para la interfaz gráfica y Python para la lógica subyacente. El enfoque principal fue manejar concurrencia mediante memoria compartida, permitiendo que múltiples instancias del programa accedan a una lista de productos y sincronicen sus acciones, como visualizar y comprar productos. Para lograr esto, se utilizó el módulo `multiprocessing` de Python, junto con `Manager` y `Lock` para garantizar la integridad de los datos y evitar accesos simultáneos a productos que ya estaban siendo manipulados por otro proceso. Además, se implementó la actualización dinámica del inventario, sincronizando la información entre los procesos y el archivo CSV que actúa como base de datos.

El desarrollo de esta aplicación permitió profundizar en el manejo de interfaces gráficas y el control de procesos concurrentes en Python. A través de la memoria compartida y la sincronización de procesos, se logró una solución eficiente que garantiza la coherencia en el acceso a los recursos compartidos, como el inventario de productos. Es importante aplicar mecanismos de control en sistemas concurrentes para evitar condiciones de carrera y mantener la integridad de los datos, conocimientos cruciales en el desarrollo de aplicaciones que requieren alta concurrencia o distribución de tareas.

Finalmente el saber implementar memoria compartida y manejar procesos concurrentes es esencial en el desarrollo de sistemas y aplicaciones que requieren eficiencia y sincronización en el uso de recursos. Esto es particularmente útil en escenarios donde múltiples procesos o usuarios

necesitan acceder y modificar los mismos datos, como en sistemas de inventario, servidores web, bases de datos, videojuegos multijugador, entre otros.