Trabajo final

Fundamentos de Informática

integrantes del grupo: Maria Victoria Blumenthal, Lola Jamuy, Lucas Santivañez y Martina Unamunzaga

UNIVERSIDAD DEL CEMA
UCEMA

Contenidos

"La nueva tecnología para el mundo de la moda en Argentina"

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

}

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

- Desorden
- Desorganización
- Errores
- Problemas
- Robos
- Dificultades
- Pérdidas
- Estrés Laboral

}

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

1. **Algunas tiendas carecen de sistemas internos para gestionar eficientemente la información sobre sus productos.**

- Desorden
- Desorganización
- Errores
- Problemas
- Robos
- Dificultades
- Pérdidas
- Estrés Laboral

}

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

1. Algunas tiendas carecen de sistemas internos para gestionar eficientemente la información sobre sus productos.
2. **Gran cantidad de productos, modelos, colores y tallas generan caos en la gestión de inventario.**

- Desorden
- Desorganización
- Errores
- Problemas
- Robos
- Dificultades
- Pérdidas
- Estrés Laboral

}

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

1. Algunas tiendas carecen de sistemas internos para gestionar eficientemente la información sobre sus productos.
2. Gran cantidad de productos, modelos, colores y tallas generan caos en la gestión de inventario.
3. **Falta de organización: robos, pérdidas y dificultades para rastrear productos.**

- Desorden
- Desorganización
- Errores
- Problemas
- Robos
- Dificultades
- Pérdidas
- Estrés Laboral

}

Problemática encontrada

- Complejidad Organizativa en Tiendas de Ropa en Flores, Buenos Aires:

1. Algunas tiendas carecen de sistemas internos para gestionar eficientemente la información sobre sus productos.
2. Gran cantidad de productos, modelos, colores y tallas generan caos en la gestión de inventario.
3. Falta de organización: robos, pérdidas y dificultades para rastrear productos.
4. **El desorden impacta negativamente en la capacidad de las tiendas para satisfacer la demanda y controlar el inventario.**

- Desorden
- Desorganización
- Errores
- Problemas
- Robos
- Dificultades
- Pérdidas
- Estrés Laboral

}

IDEA - Solución Propuesta

Desarrollo de un Sistema Interno para Organizar el Inventario de los locales de ropa en Flores:

- Inclusión de características clave: como ID único para cada producto, detalles de precio, stock, material, color y tipo de tela.
- Esto hara optimizar a los locales en las fechas más importantes que sean mas eficiente y logren mayores beneficios.

}

IDEA - Solución Propuesta

Desarrollo de un Sistema Interno para Organizar el Inventario de los locales de ropa en Flores:

- Inclusión de características clave: como ID único para cada producto, detalles de precio, stock, material, color y tipo de tela.
- Esto hara optimizar a los locales en las fechas más importantes que sean mas eficiente y logren mayores beneficios.

- Re-Estructuración.
- Revisión de errores.
- Corrección.
- Órden
- Enfoque en vender más
- Mayores beneficios

}

OBJETIVOS

Objetivo principal: Mejora en la Eficiencia y Rentabilidad de las Tiendas

}

OBJETIVOS

Objetivo principal: Mejora en la Eficiencia y Rentabilidad de las Tiendas

- **Facilitar el acceso a información clave para empleados, simplificando la gestión de inventario.**

}

OBJETIVOS

Objetivo principal: Mejora en la Eficiencia y Rentabilidad de las Tiendas

- Facilitar el acceso a información clave para empleados, simplificando la gestión de inventario.

- **Reducción de pérdidas, robos y mejor organización para que las tiendas puedan centrarse en aumentar las ventas y optimizar operaciones.**

}

OBJETIVOS

Objetivo principal: Mejora en la Eficiencia y Rentabilidad de las Tiendas

- Facilitar el acceso a información clave para empleados, simplificando la gestión de inventario.

- Reducción de pérdidas, robos y mejor organización para que las tiendas puedan centrarse en aumentar las ventas y optimizar operaciones.

- **Beneficios Esperados:**

1. **Mayor control sobre el inventario.**
2. **Incremento de las ganancias al minimizar pérdidas.**
3. **Mejora en la experiencia del cliente al facilitar la disponibilidad de productos.**
4. **Mayor eficiencia operativa y reducción de dolores de cabeza relacionados con la gestión del inventario.**

Planificación {

DEFINICION DE LA CLASE

CLASE_ROPA.PY

Define la clase Book, sus atributos y sus métodos

TABLA Y BASE DE DATOS

DB_ROPA.PY

Crea la tabla en la base de datos

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma, aquí vemos la consulta de los datos a través de los distinos métodos

API DE TERCEROS

EXCHANGE_RATE.PY

Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.

}

Planificación {

DEFINICION DE LA CLASE

CLASE_ROPA.PY

Define la clase Book, sus atributos y sus métodos

TABLA Y BASE DE DATOS

DB_ROPA.PY

Crea la tabla en la base de datos

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma, aquí vemos la consulta de los datos a través de los distinos métodos

API DE TERCEROS

EXCHANGE_RATE.PY

Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.

}

Planificación {

DEFINICION DE LA CLASE

CLASE_ROPA.PY

Define la clase Book, sus atributos y sus métodos

TABLA Y BASE DE DATOS

DB_ROPA.PY

Crea la tabla en la base de datos

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma, aquí vemos la consulta de los datos a través de los distinos métodos

API DE TERCEROS

EXCHANGE_RATE.PY

Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.

}

Planificación {

DEFINICION DE LA CLASE

CLASE_ROPA.PY

Define la clase Book, sus atributos y sus métodos

TABLA Y BASE DE DATOS

DB_ROPA.PY

Crea la tabla en la base de datos

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma, aquí vemos la consulta de los datos a través de los distinos métodos

API DE TERCEROS

EXCHANGE_RATE.PY

Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.

}

Planificación {

DEFINICION DE LA CLASE

CLASE_ROPA.PY

Define la clase Book, sus atributos y sus métodos

TABLA Y BASE DE DATOS

DB_ROPA.PY

Crea la tabla en la base de datos

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma, aquí vemos la consulta de los datos a través de los distinos métodos

API DE TERCEROS

EXCHANGE_RATE.PY

Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.

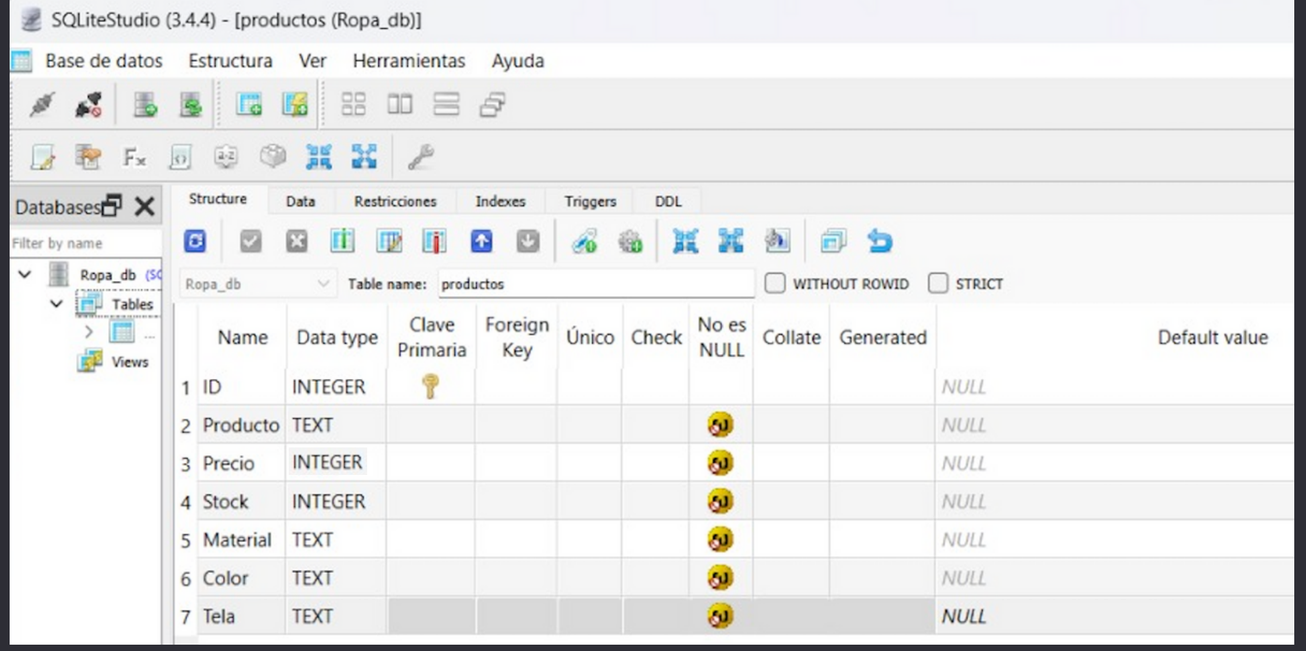
}

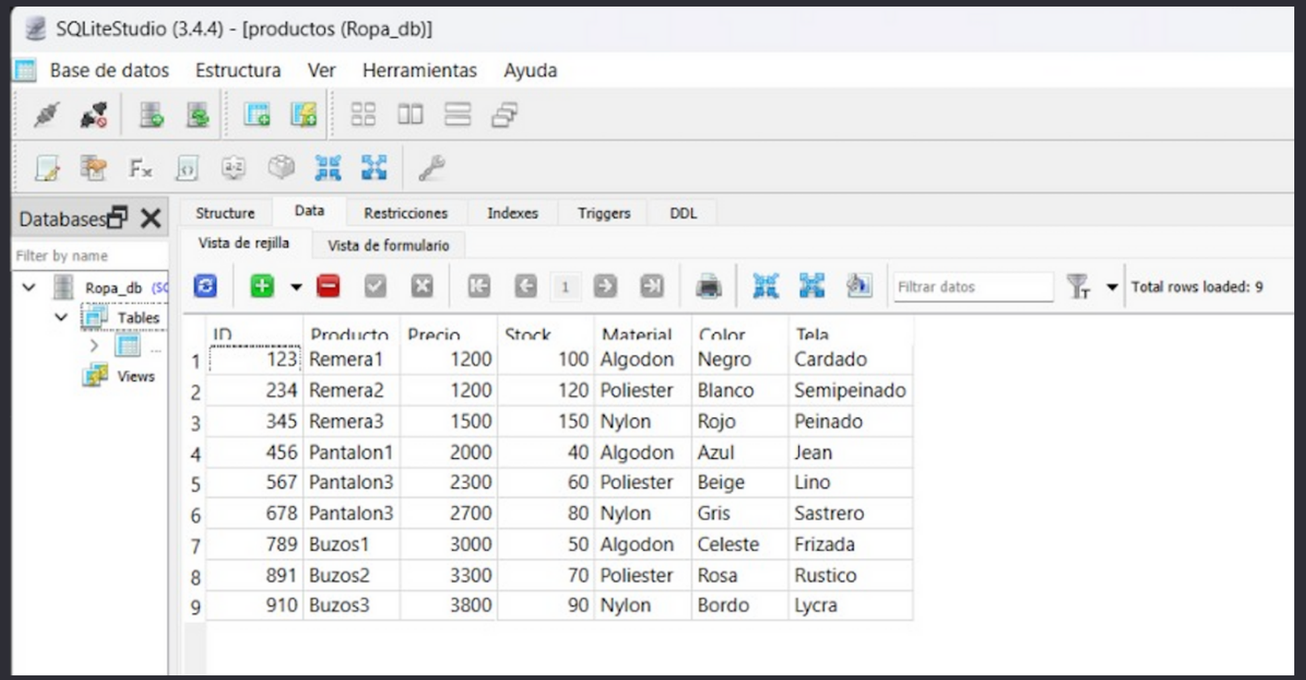
Creación de Base de Datos con SQLite Studio

- Utilizando los métodos y atributos correspondientes, comenzamos a crear nuestra base de datos

- Utilizamos los atributos: ID, Producto, Precio, Stock, Material, Color y Tela

- De esta manera, el contenido de nuestra tabla contiene tres tipos de productos (Remera, Pantalón y Buzo)





DEFINICION DE LA CLASE

CLASE_ROPA.PY

Definimos nuestra clase con los atributos mencionados anteriormente

Muestra fragmento de código de la exportación de datos.

Definimos funciones de serialize y serialize_details; a fin de destacar nuestros atributos más importantes y luego agruparlos

```
5 usages  Vicky Blumenthal
1  class Ropa:
2
       Vicky Blumenthal
3      def _init_(self, id, producto, precio, stock, material, color, tela) -> None:
4          self.id = id
5          self.producto = producto
6          self.precio = precio
7          self.stock = stock
8          self.material = material
9          self.color = color
10         self.tela = tela
11
       1 usage (1 dynamic)  Vicky Blumenthal
12     def serialize(self):
13         return {
14             'id': self.id,
15             'producto': self.producto,
16             'precio': self.precio,
17             'stock': self.stock
       1 usage  Vicky Blumenthal
20     def serialize_details(self):
21         return {
22             'id': self.id,
23             'producto': self.producto,
24             'precio': self.precio,
25             'stock': self.stock,
26             'material': self.material,
27             'color': self.color,
28             'tela': self.tela,
29         }
```

TABLA Y BASE DE DATOS

DB_ROPA.PY

Importando nuestra base de datos denominada “ropa.db”, creamos la tabla en la base de datos

```python
import sqlite3
DATABASE_NAME = "ropa.db"


8 usages  Vicky Blumenthal
def get_db():
    conn = sqlite3.connect(DATABASE_NAME)
    return conn


2 usages  Vicky Blumenthal
def create_tables():
    tables = [
        """CREATE TABLE IF NOT EXISTS ropa(
                ID INTEGER PRIMARY KEY,
                producto TEXT NOT NULL,
                precio INTEGER NOT NULL,
                stock INTEGER NOT NULL,
                material TEXT NOT NULL,
                color TEXT NOT NULL,
                tela TEXT NOT NULL
            )
        """
    ]
    db = get_db()
    cursor = db.cursor()
    for table in tables:
        cursor.execute(table)
```

}

create_tables() › for table in tables

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Contiene todas las funciones para realizar las distintas operaciones en la base de datos.

Aquí se definen dichas funciones a fin de facilitar el trabajo de nuestra empleadora si desea actaualizar su base de datos con productos nuevos que ha agregado

```
2 usages  Vicky Blumenthal
30  def insert_ropa(id, producto, precio, stock, material, color, tela):
31      db = get_db()
32      cursor = db.cursor()
33      statement = "INSERT INTO ropas (id, producto, precio, stock, material, color, tela) \
34      VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?)"
35      cursor.execute(statement, __parameters: [id, producto, precio, stock, material, color, tela])
36      db.commit()
37      return True
38
    1 usage  Vicky Blumenthal
39  def update_ropa(id, producto, precio, stock, material, color, tela):
40      db = get_db()
41      cursor = db.cursor()
42      statement = "UPDATE ropas SET id = ?, producto = ?, stock= ?, material= ?, color= ?, tela= ?, \
43      WHERE id = ?"
44      cursor.execute(statement, __parameters: [id, producto, precio, stock, material, color, tela])
45      db.commit()
46      return True
47
48
    2 usages  Vicky Blumenthal
49  def delete_ropa(id):
50      db = get_db()
51      cursor = db.cursor()
52      statement = "DELETE FROM ropas WHERE id = ?"
53      cursor.execute(statement, __parameters: [id])
54      db.commit()
55      return True
```

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Operaciones CRUD

- **insert_ropa**: *Inserta* un nuevo producto de ropa en la mesa ropas.
- **update_ropa**: *Actualiza* un producto existente en la tabla basado en su ID
- **delete_ropa**: *Elimina* un producto de ropa de la tabla basándose en su ID.
- get_by_id: Recupera un producto de ropa de la tabla basándose en su ID.
- get_ropas: Recupera una lista de todos los productos de ropa de la tabla

```
2 usages  Vicky Blumenthal
30 def insert_ropa(id, producto, precio, stock, material, color, tela):
31     db = get_db()
32     cursor = db.cursor()
33     statement = "INSERT INTO ropas (id, producto, precio, stock, material, color, tela) \
34     VALUES ( ?, ?, ?, ?, ?, ?, ?, ?, ?)"
35     cursor.execute(statement, __parameters: [id, producto, precio, stock, material, color, tela])
36     db.commit()
37     return True
38
1 usage  Vicky Blumenthal
39 def update_ropa(id, producto, precio, stock, material, color, tela):
40     db = get_db()
41     cursor = db.cursor()
42     statement = "UPDATE ropas SET id = ?, producto = ?, stock= ?, material= ?, color= ?, tela= ?, \
43     WHERE id = ?"
44     cursor.execute(statement, __parameters: [id, producto, precio, stock, material, color, tela])
45     db.commit()
46     return True
47
48
2 usages  Vicky Blumenthal
49 def delete_ropa(id):
50     db = get_db()
51     cursor = db.cursor()
52     statement = "DELETE FROM ropas WHERE id = ?"
53     cursor.execute(statement, __parameters: [id])
54     db.commit()
55     return True
```

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

Operaciones CRUD

- insert_ropa: Inserta un nuevo producto de ropa en la mesa ropas.
- update_ropa: Actualiza un producto existente en la tabla basado en su ID
- delete_ropa: Elimina un producto de ropa de la tabla basándose en su ID.
- **get_by_id**: *Recupera* un producto de ropa de la tabla basándose en su ID.
- **get_ropas**: *Recupera* una lista de todos los productos de ropa de la tabla

```
2 usages  Vicky Blumenthal
def get_by_id(id):
    db = get_db()
    cursor = db.cursor()
    statement = "SELECT id, producto, precio, stock, material, color, tela FROM ropa WHERE id = ?"
    cursor.execute(statement, __parameters: [id])
    single_ropa = cursor.fetchone()
    id = single_ropa[0]
    producto = single_ropa[1]
    precio = single_ropa[2]
    stock = single_ropa[3]
    material = single_ropa[4]
    color = single_ropa[5]
    tela = single_ropa[6]
    ropa = Ropa(id, producto, precio, stock, material, color, tela)
    return ropa.serialize_details()
```

```
def get_ropas():
    db = get_db()
    cursor = db.cursor()
    query = "SELECT id, producto, precio, stock, material, color, tela FROM ropas"
    cursor.execute(query)
    ropa_list = cursor.fetchall()
    list_of_ropas=[]
    for ropa in ropa_list:
        id = ropa[0]
        producto = ropa[1]
        precio = ropa[2]
        stock = ropa[3]
        material = ropa[4]
        color = ropa[5]
        tela = ropa[6]
        ropa_to_add = Ropa(id, producto, precio, stock, material, color, tela)
        list_of_ropas.append(ropa_to_add)
    return list_of_ropas
```

FUNCIONES Y OPERACIONES PARA LA BASE DE DATOS

ROPA_CONTROLLER_POO.PY

MENU

- Proporciona un sencillo menú de línea de comandos para interactuar con la base de datos.
- Las opciones incluyen cargar datos de un archivo, agregar un producto, eliminar un producto, buscar un producto por ID, enumerar todos los productos y salir del menú.

```
def menu():
    print('******************Menu*****************')
    print()
    print('ingrese 0 para cargar desde archivo de texto')
    print('ingrese 1 para agregar un producto a la base de datos')
    print ('ingrese 2 para eliminar un producto de la base de datos')
    print ('ingrese 3 para buscar un producto por su id')
    print('ingrese 4 para listar todos los productos')
    print ('ingrese 5 para salir del menu')
```

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma

Definición de Rutas

- **/ropa(GET):** Recupera la lista de productos de ropa y devuelve una respuesta JSON
- **/ropa/create(POST):** Inserta un nuevo producto de ropa utilizando los datos proporcionados en el cuerpo de la solicitud JSON.

```python
from flask import Flask, jsonify, request
from db_ropa import create_tables
import ropa_controller_poo

app = Flask(__name__)


Vicky Blumenthal
@app.route('/ropa', methods=["GET"])
def get_ropa():
    ropas = ropa_controller_poo.get_ropas()
    ropas_list = []
    for ropa in ropas:
        elem = ropa.serialize()
        ropas_list.append(elem)
    return jsonify(ropas_list)

Vicky Blumenthal
@app.route("/ropa/create", methods=["POST"])
def insert_ropa():
    ropa_details = request.get_json()
    id = ropa_details["id"]
    producto = ropa_details["producto"]
    precio = ropa_details["precio"]
    stock = ropa_details["stock"]
    material = ropa_details["material"]
    color = ropa_details["color"]
    tela = ropa_details["tela"]
    result = ropa_controller_poo.insert_ropa(id, producto, precio, stock, material, color, tela)
```

CREACIÓN DE API

SERVER_ROPA_POO.PY

Es la API en sí misma

Definición de Rutas

- **/ropa/modify(PUT):** Actualiza un producto de ropa existente basándose en los datos proporcionados en el cuerpo de la solicitud JSON.
- **/ropa/eliminate/<id>(DELETE):** Elimina un producto de ropa según su ID proporcionado en la URL.
- **/ropa/<id>(GET):** Recupera un producto de ropa específico basándose en su ID proporcionado en la URL.

```
    return jsonify(result)

Vicky Blumenthal
@app.route("/ropa/modify", methods=["PUT"])
def update_ropa():
    ropa_details = request.get_json()
    id = ropa_details["id"]
    producto = ropa_details["producto"]
    precio = ropa_details["precio"]
    stock = ropa_details["stock"]
    material = ropa_details["material"]
    color = ropa_details["color"]
    tela = ropa_details["tela"]
    result = ropa_controller_poo.update_ropa(id, producto, precio, stock, material, color, tela)
    return jsonify(result)

Vicky Blumenthal
@app.route("/ropa/eliminate/<id>", methods=["DELETE"])
def delete_ropa(id):
    result = ropa_controller_poo.delete_ropa(id)
    return jsonify(result)

Vicky Blumenthal
@app.route("/ropa/<id>", methods=["GET"])
def get_ropa_by_id(id):
    ropa = ropa_controller_poo.get_by_id(id)
    return jsonify(ropa)
```

```
if __name__ == '__main__':
    create_tables()
    app.run()
```

API DE TERCEROS

EXCHANGE_RATE.PY

- Esta API devuelve varios de los tipos de cambio que hay en Argentina hoy en día.
- Con el objetivo de que nuestra dueña del local pueda ver en tiempo real las modificaciones del dólar y en base a eso que ésta decida cómo proseguir con su negocio

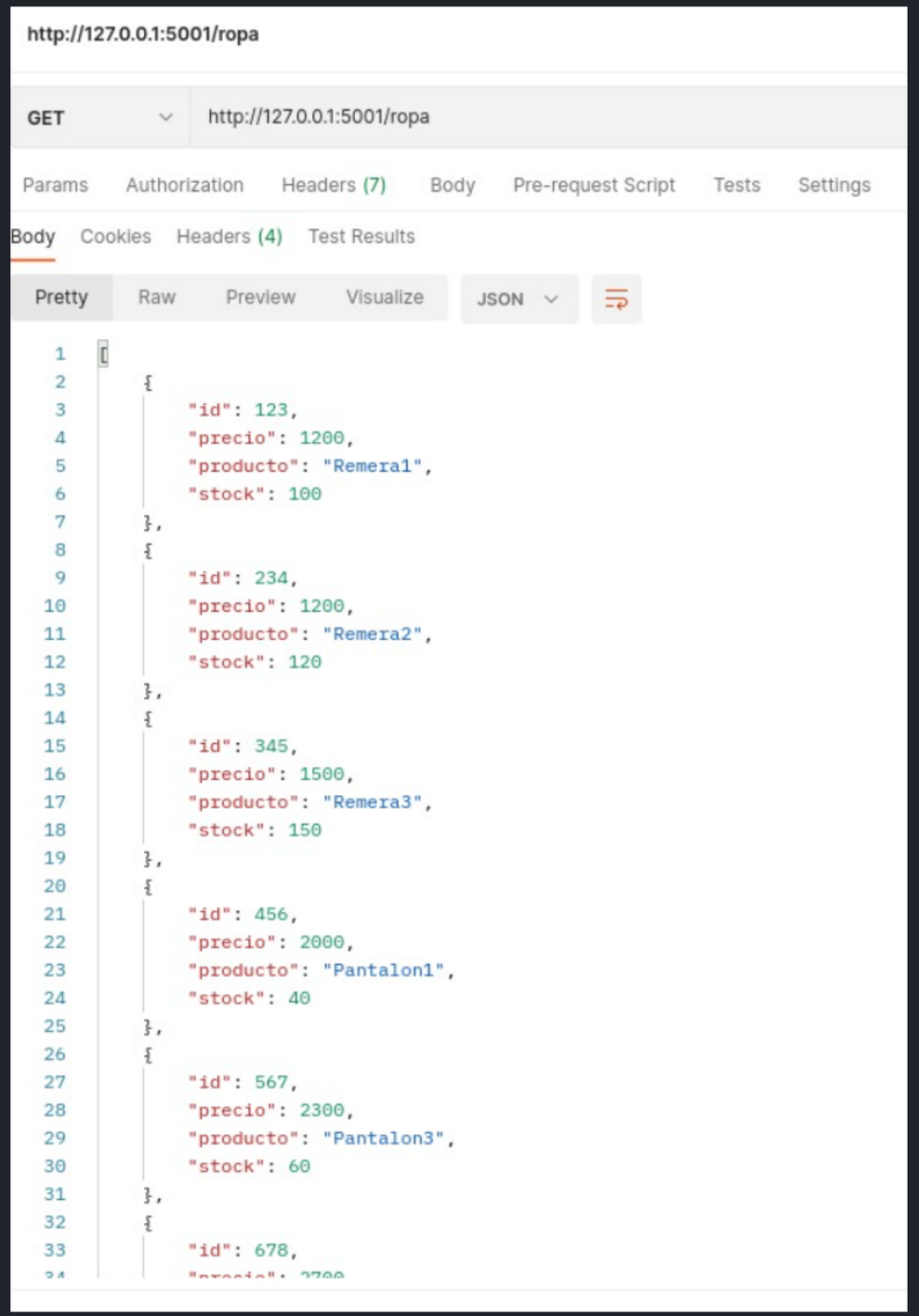
```python
import json
import requests

Vicky Blumenthal
def get_xr():
    url = 'https://api.bluelytics.com.ar/v2/latest'
    r = requests.get(url)
    data=json.loads(r.text)
    x_rate = data['oficial']['value_sell']
    return x_rate
```

EJECUTANDO EN POSTMAN

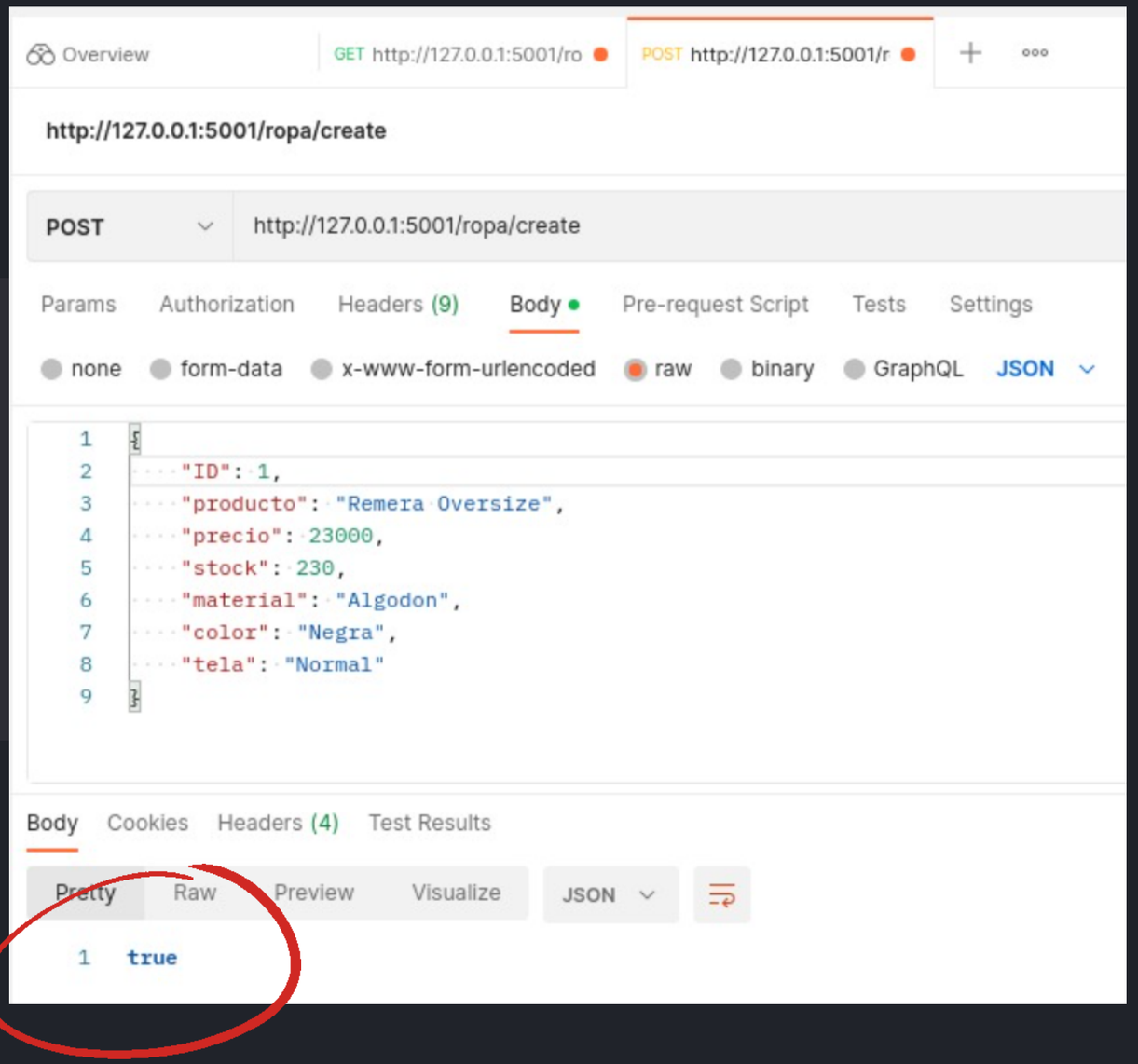
- Plataforma de colaboración para el desarrollo de API

- Postman proporciona un entorno amigable para que los desarrolladores puedan crear, probar y documentar APIs.



EJECUTANDO EN POSTMAN

- Con nuestra API corriendo en el link http://127.0.0.1:5000
- De esta manera, ¡comprobamos que nuestra API funciona!



¡Muchas gracias!