Universidad ORT Uruguay

# Obligatorio 1
# Diseño de Aplicaciones 2

Ejecución de las pruebas de la API con Postman

Martin Edelman - 263630

Tatiana Poznanski - 221056

Tomas Bañales - 239825

Profesores: Nicolás Fierro, Alexander Wieler, Marco Fiorito

2023

# 1. Link al repositorio

https://github.com/IngSoft-DA2-2023-2/263630_221056_239825

# 2. Pruebas en Postman

Para demostrar la ejecución de los diferentes endpoints y funcionalidades de la API, creamos una colección en Postman con todas las pruebas necesarias. Dentro de esta, organizamos las pruebas en diversas carpetas, correspondientes a cada controlador de nuestro sistema, lo que facilita la búsqueda y comprensión de los casos de prueba específicos. Implementando el uso de Clean Code, asignamos nombres distintivos e intuitivos a cada prueba, de modo que al leer el nombre, se pueda comprender de inmediato a qué se refiere el endpoint en cuestión.

Asimismo, dividimos el proyecto de Postman en 4 sectores (Anexo, Imágenes 1 y 2), las pruebas con (*), los endpoints de usuario, los endpoints de productos y los endpoints de compras. Y, con el fin de evidenciar el correcto funcionamiento, realizamos un video.

Al utilizar Postman, podemos demostrar de manera efectiva la funcionalidad de nuestra API y validar que todos los endpoints funcionen según lo previsto. Esto proporciona una evidencia sólida de que nuestro sistema cumple con los requisitos y está listo para su implementación en un entorno de producción.

# 3. Funcionalidades con (*)

## Post usuarios

**Crear usuario mail incorrecto**

Body:

```
{
 "correoElectronico": "martinedelman.com.uy",
 "direccionEntrega": "string",
 "contrasena": "12345678",
 "rol": 1
}
```

Output: 400 Bad Request

```
"El email es incorrecto"
```

**Crear usuario dirección incorrecta**

Body:

```
{
```

```
 "correoElectronico": "martin@edelman.com.uy",
 "direccionEntrega": "",
 "contrasena": "12345678",
 "rol": 1
}
```

Output: 400 Bad Request

```
"La direccion de entrega no puede ser nula o vacia"
```

**Crear usuario contraseña incorrecta**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "direccionEntrega": "string",
 "contrasena": "12345",
 "rol": 1
}
```

Output: 400 Bad Request

```
"Contraseña no valida"
```

**Crear usuario Ok Cliente**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "direccionEntrega": "string",
 "contrasena": "12345678",
 "rol": 1
}
```

Output: 201 Created

```
{
  "id": 4,
  "correoElectronico": "martin.aaron.edelman@gmail.com",
  "direccionEntrega": "string",
  "compras": [],
  "rol": 1,
  "contrasena": "12345678"
}
```

**Crear usuario Mail Repetido**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "direccionEntrega": "string",
 "contrasena": "87654321",
 "rol": 1
}
```

Output: 400 Bad Request

```
"El email es incorrecto"
```

# Post authentication

LogIn Usuario Mail Incorrecto

```
{
 "correoElectronico": "martin.edelman@gmail.com",
 "contrasena": "12345678"
}
```

Output: 404 Not Found

```
"Credenciales incorrectas"
```

**LogIn Usuario Contraseña Incorrecta**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "contrasena": "87654321"
}
```

Output: 404 Not Found

```
"Credenciales incorrectas"
```

**LogIn Usuario Ok**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "contrasena": "12345678"
}
```

Output: 201 Created

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJiYXNlV2ViQXBpU3ViamVjdCIsImp0aSI6IjgxZWU1ZDFiLWQxOGYtNGRkNS04MDVhLWVhNmE0ZmMxNzIxYSIsImlhdCI6IjUvMTAvMjMgMjI6Mzg6NTciLCJlbWFpbCI6Im1hcnRpbkBlZGVsbWFuLmNvbS51eSIsImlkIjoiMyIsInJvbCI6IkFkbWluaXN0cmFkb3IiLCJleHAiOjE2OTkxMzc1MzcsImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6NzA2MS8iLCJhdWQiOiJodHRwOi8vbG9jYWxob3N0OjcwNjEvIn0.5LA0G-hKRxdwkWRRAf4PoUKBsQYofZoCN9n7nAej9ac

El token creado es aleatorio y único para cada usuario.

# PUT Usuario/{id}

Este endpoint recibe como parametro un header con el token además del body.

Header:

- Key: “token”
- Value: “string” (valor del token asociado al usuario)

**Usuario sin mail**

Body:

```
{
 "correoElectronico": "martin.com",
 "direccionEntrega": "Julio Cesar 1247",
 "rol": 2,
 "contrasena": "12345678"
}
```

Output: 400 Bad Request

```
"El email es incorrecto"
```

**Usuario sin dirección**

Body:

```
{
 "correoElectronico": "martin.aaron.edelman@gmail.com",
 "direccionEntrega": "",
 "rol": 2,
 "contrasena": "12345678"
}
```

Output: 400 Bad Request

```
"La direccion de entrega no puede ser nula o vacia"
```

**Usuario sin contraseña**

Body:

```
{
 "correoElectronico": "martin.aaron.edelman@gmail.com",
 "direccionEntrega": "Julio Cesar 1247",
 "rol": 2,
 "contrasena": "1234"
}
```

Output: 400 Bad Request

```
"Contraseña no valida"
```

**Usuario Token Incorrecto**

Header: Se da un token de otro usuario o un string al azar

Body:

```
{
 "correoElectronico": "martin.aaron.edelman@gmail.com",
 "direccionEntrega": "Julio Cesar 1247",
 "rol": 2,
 "contrasena": "12345678"
}
```

Output: 401 Unauthorized

```
"No Autorizado"
```

**Usuario Ok**

Body:

```
{
 "correoElectronico": "martin@edelman.com.uy",
 "direccionEntrega": "Julio Cesar 1247",
 "rol": 0,
 "contrasena": "123456789"
}
```

Output: 200 Ok

# DELETE usuario/{id}

Este endpoint recibe como parámetro un header con el token pero sin un body.

Header:

- Key: “token”

- Value: “string” (valor del token asociado al usuario)

**Usuario Sin token**

Output: 401 Unauthorized

```
"No Autorizado"
```

**Usuario Sin token**

Output: 401 Unauthorized

```
"No Autorizado"
```

**Usuario No existe**

Output: 404 Not Found

```
"No existe el usuario con la id dada"
```

**Usuario Sin token**

Output: 200 Ok

# GET compras

Este endpoint recibe como parámetro un header con el token pero sin un body.

Header:

- Key: “token”
- Value: “string” (valor del token asociado al usuario)

**Compras Token Incorrecto**

Output: 401 Unauthorized

```
"Attempted to perform an unauthorized operation."
```

**Compras Ok**

Output: 200 Ok


```
[
  {
    "id": 15,
    "productos": [
      8,
      9,
      10
    ],
    "precio": 1200,
```

```
    "nombrePromo": "Se aplico un 50% de descuento en el producto de mayor valor",
    "fechaCompra": "2023-10-05T18:02:20.27818",
    "usuarioId": 3
  },
  {
    "id": 16,
    "productos": [
      8,
      9,
      10
    ],
    "precio": 1200,
    "nombrePromo": "Se aplico un 50% de descuento en el producto de mayor valor",
    "fechaCompra": "2023-10-05T18:02:57.0766475",
    "usuarioId": 3
  }
]
```


# POST usuario/{id}/compras

Este endpoint recibe como parámetro un header con el token además del body.

Header:

- Key: “token”
- Value: “string” (valor del token asociado al usuario)

**Compras Sin Productos**

Body:

```
{
 "idProductos": [
 ]
}
```

Output:

```
"La compra debe tener al menos un producto"
```

**Compras Token Incorrecto**

Output: 401 Unauthorized

```
"No Autorizado"
```

**Compras Total Look**

Body:

```
{
 "idProductos": [
  8,9,10
 ]
}
```

Output: 201 Created

```
{
  "idProductos": [
    8,
    9,
    10
  ]
}
```

**Compras 3x1**

Body:

```
{
 "idProductos": [
  21,22,23
 ]
}
```

Output: 201 Created

```
{
  "idProductos": [
    21,
    22,
    23
  ]
}
```

**Compras 3x2**

Body:

```
{
 "idProductos": [
  14,18,19,20
 ]
}
```

Output: 201 Created

```
{
  "idProductos": [
    14,
    18,
    19,
    20
  ]
}
```

**Compras 20% off**

Body:

```
{
 "idProductos": [
    11,12,13
 ]
}
```

Output: 201 Created

```
{
  "idProductos": [
    11,
    12,
    13
  ]
}
```

# 4. Video de YouTube

En el siguiente video, se encuentra Martín Edelman explicando cómo utilizó las pruebas anteriores, más algunas extra para poder utilizarlas.

https://youtu.be/uf2N-r2v1Dg

# Anexo

## [Imágenes 1 y 2]