

UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

FACULTAD DE INGENIERÍA

INGENIERÍA EN SISTEMAS

SEDE EL NARANJO

DESARROLLO WEB



ANA LAURA MERCEDES SANTIZO DOMÍNGUEZ 9490-20-12280

ELVIS ESTUARDO DE LEÓN MORALES 9490-20-17682

Índice

Introducción	3
Objetivos	3
Requisitos previos	3
Instalación de aplicación	4
Endpoints.....	7
Registro de usuarios	7
LOGIN	7
Gestión de perfil.....	8
Catálogo de productos.....	8
Gestión de productos.....	9
Carrito de compra.....	10
Módulo de compra	11
Términos y Condiciones de Uso	12
PANTALLAS DEL PROCESO PARA UTILIZACIÓN DE LOS ENDPOINTS	12
Registrar usuario:	12
Iniciar sesión recibe token:.....	13
Obtener usuario por dpi con bearer token para autenticar.....	13
Se modifica el perfil del usuario, se modificó el nombre y apellido	14
Se eliminó usuario y el token se elimina tambien	14
Se crea producto nuevo	15
Se obtienen todos los productos	15
Se modifica producto por id se cambia descuento y stock.....	16
Se elimina producto.....	16
Se crea carrito de compras	17
Obtiene carrito de compras	17
Borra producto del carrito	18
Genera compra y crea bitácora de compra para usuario	18

Introducción

En el presente manual encontrarás todo lo que necesitas saber para sacar el máximo provecho de nuestra plataforma, que te permite acceder a nuestras APIs de manera eficiente. Con un sencillo proceso de registro de sesión, podrás tener un sistema de comercio electrónico con un login, un catálogo de productos y un módulo de compra. Nuestra interfaz intuitiva y amigable te guiará a través de la gestión de tus tareas, ayudándote a ser más productivo que nunca.

Objetivos

- Implementar la autenticación de usuarios utilizando JWT para garantizar la seguridad de las peticiones.
- Diseñar y desarrollar módulos de registro de usuario, login y gestión de perfiles con actualizaciones y eliminaciones.
- Exponer una API que permita acceder al catálogo de productos disponibles y realizar búsquedas por nombre y categoría.
- Desarrollar un módulo de carrito de compra que permita a los usuarios agregar productos y finalizar compras.
- Crear un módulo administrativo que permita a los administradores gestionar productos, inventarios y ventas.
- Establecer la comunicación con una base de datos en línea, utilizando MongoDB para almacenar y recuperar datos.
- Autenticación: Explica cómo autenticarse correctamente para acceder a la API, si es necesario (por ejemplo, mediante tokens de API, OAuth, etc.).

Requisitos previos

Tener instalados todos los paquetes y librerías necesarias para poder ejecutarlo.

Node.js: Asegúrate de tener Node.js instalado en tu sistema y de que esté configurado correctamente.

Express.js: Asegúrate de que Express.js esté instalado como una dependencia en tu proyecto.

Módulos NPM.

Middleware Passport: Asegúrate de que passport y passportMiddleware estén configurados correctamente para autenticar las solicitudes entrantes.

Rutas: Asegúrate de que estas rutas estén definidas y configuradas correctamente en sus respectivos archivos. Esto incluye la definición de rutas y controladores asociados.

Puerto de Escucha: Puerto 8080.

Logger (Morgan).

CORS.

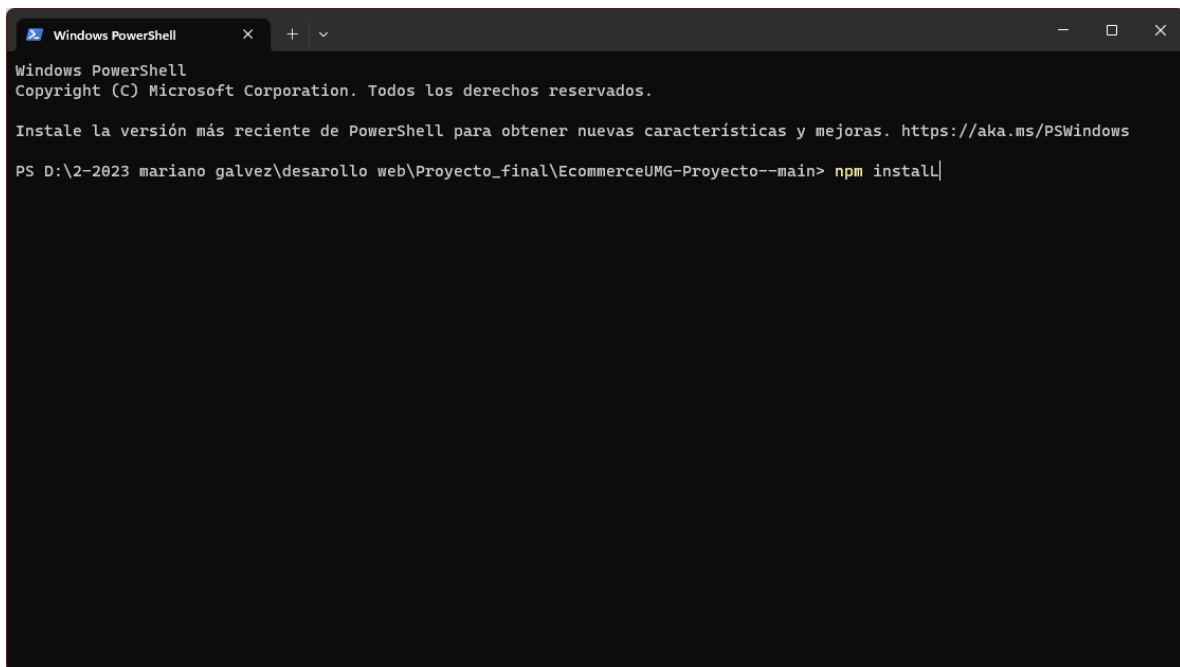
Parseo de Cuerpo: `express.urlencoded` y `express.json`.

Enrutamiento: (`user.routes.js`, `product.routes.js`, etc.) estén configuradas.

Instalación de aplicación

Abrir un terminal dentro la carpeta donde se encuentra la aplicación y colocar

- **Npm install**

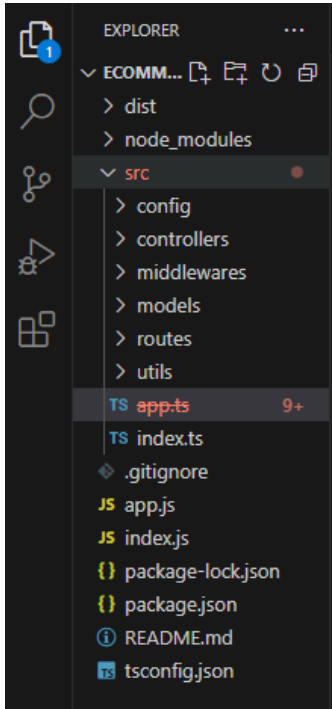


```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS D:\2-2023 mariano galvez\desarrollo web\Proyecto_final\EcommerceUMG-Proyecto--main> npm install
```

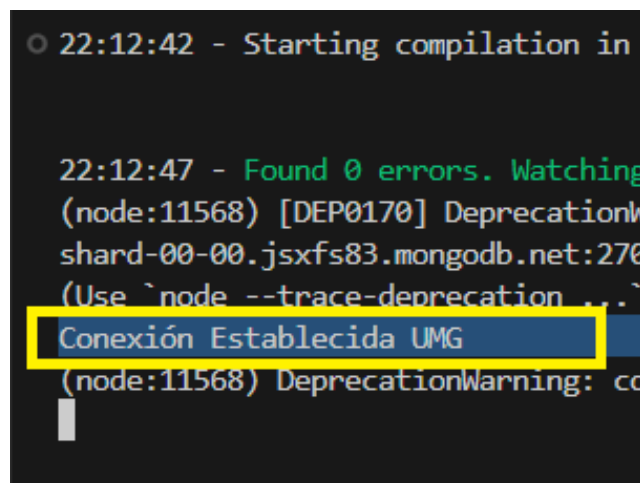
A partir de acá se realizó la instalación de las dependencias necesarias para poder construir el proyecto y se va a crear la carpeta “node_modules” dentro de la carpeta raíz del proyecto similar a la siguiente ilustración:



Ahora el ambiente se encuentra listo para poder realizar la compilación en local utilizando el siguiente comando:

- Npm run dev

Si la compilación fue correcta, va a compilar y mostrar la ruta de compilación, similar a la siguiente ilustración:



```
© 22:12:42 - Starting compilation in watch mode...

22:12:47 - Found 0 errors. Watching for file changes.
(node:11568) [DEP0170] DeprecationWarning: The URL mongodb://Bataman:Morales5616@ac-2ecrz9i-shard-00-01.jsxfs83.mongodb.net:27017,ac-2ecrz9i-shard-00-00.jsxfs83.mongodb.net:27017/?authSource=admin&replicaSet=atlas-1vgmz8-shard-0&ssl=true is invalid. Future versions of Node.js will throw an error.
(Use "node --trace-deprecation ..." to show where the warning was created)
Conexión Establecida UMG
(node:11568) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Si no logra conectarse a la base de datos o algún otro conveniente recibirá el siguiente error.

```
C:\WINDOWS\system32\cmd. X + v
de_modules\mongoose\lib\index.js:1149:10)
    at Mongoose.connect (D:\2-2023 mariano galvez\desarollo web\Proyecto_final\EcommerceUMG-Proyecto--main\node_modules\mongoose\lib\index.js:350:20)
    at Object.<anonymous> (D:\2-2023 mariano galvez\desarollo web\Proyecto_final\EcommerceUMG-Proyecto--main\dist\config\database.js:12:20)
    at Module._compile (node:internal/modules/cjs/loader:1256:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1310:10)
    at Module.load (node:internal/modules/cjs/loader:1119:32)
    at Module._load (node:internal/modules/cjs/loader:960:12)
    at Module.require (node:internal/modules/cjs/loader:1143:19)
    at require (node:internal/modules/cjs/helpers:121:18)
    at Object.<anonymous> (D:\2-2023 mariano galvez\desarollo web\Proyecto_final\EcommerceUMG-Proyecto--main\dist\index.js:7:1)
    at Module._compile (node:internal/modules/cjs/loader:1256:14) {
  reason: TopologyDescription {
    type: 'Single',
    setName: null,
    maxSetVersion: null,
    maxElectionId: null,
    servers: Map(1) { 'localhost:27017' => [ServerDescription] },
    stale: false,
    compatible: true,
    compatibilityError: null,
    logicalSessionTimeoutMinutes: null,
    heartbeatFrequencyMS: 10000,
    localThresholdMS: 15,
    commonWireVersion: null
  }
}
```

ENDPOINTS

Es la forma en la que vas a poder hacer uso de las distintas funciones de la api y lo que necesita para su funcionamiento a continuación se te presentara como esta hecho cada endpoints y sus características de cada uno.

Registro de usuarios

Modulo	Registro de Usuarios
Método	POST
Ruta	/api/registro/:DPI
Cuerpo	{ "Nombres":... "Apellidos":... "FechaNacimiento": "Clave": "ValidacionClave": "DireccionEntrega":..... "NIT":..... "NúmeroTelefonico":..... "CorreoElectronico":..... }
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error" }

- Aceptará solo correos con dominios válidos.
- Validara que no admita repetidos según NIT y DPI.
- La contraseña debe tener al menos 8 caracteres y cumplir con requisitos de seguridad (mayúsculas, minúsculas, números, caracteres especiales, etc.).
- Validara la Clave con la Validación clave.
- No permitirá campos vacíos.

LOGIN

Modulo	Login
Método	POST
Ruta	/api/login
Cuerpo	{ "CorreoElectronico":..... "Clave": }
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error", "Token": }

Si la autenticación es exitosa devolverá el respectivo Token.

Gestión de perfil

Modulo	Gestión de Perfil
Método	GET, POST, DELETE
Ruta	/api/perfil/:DPI
Encabezado	{ "Token":... }
Cuerpo	{ " }
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error", }

- Validara la veracidad del Token.
- No se podrá actualizar con campos vacíos.
- No se podrá duplicar el NIT, DPI o Correo.
- Validara el formato del correo al actualizar.
- En el método GET devolverá toda la información del usuario.
- Al eliminar el usuario ya no será válido el Token.

Catálogo de productos

Modulo	Catálogo de Productos
Método	GET
Ruta	/api/productos
Encabezado	{ "Token":... }
Cuerpo	[{ "Identificador":.... "Nombre":... "Marca":... "Disponibilidad":..... "Descuento":..... "PrecioDescuento":..... "Imagen":....., "Descripcion" "Categorias": ["Categoria1", "Categoria2", "Categorias3"] }
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error", }

Validar la veracidad del Token.

Gestión de productos

Modulo	Catálogo de Productos
Método	GET
Ruta	/api/productos
Encabezado	{ "Token":.... }
Cuerpo	[{ "Identificador":.... "Nombre":... "Marca":... "Disponibilidad":..... "Descuento":..... "PrecioDescuento":..... "Imagen":....., "Descripcion" "Categorias": ["Categoria1", "Categoria2", "Categorias3"] }]
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error", }

- En el método GET devolverá toda la información del producto.
- Al crear, actualizar o eliminar el producto se debe de validar el Token con el respectivo Rol del Usuario (Únicamente Administradores pueden realizar esta gestión).
- Al eliminar el producto se debe cambiar el estado del producto para activar nuevamente con la modificación.
- El Precio de Descuento se debe calcular con el Precio y el descuento.
- No pueden quedar campos vacíos.
- Un producto puede manejar múltiples categorías.
- Al realizar una compra debe de descontar de la cantidad de productos disponibles.

Carrito de compra

Modulo	Carrito de Compra
Método	GET, POST, DELETE
Ruta	/api/carrito
Encabezado	{ "Token":... }
Cuerpo	{...}
Respuesta	{ "Productos":[{ "Identificador":.... "Nombre":... "Marca":... "Disponibilidad":..... "Descuento":..... "PrecioDescuento":..... "Imagen":....., "Descripcion", "Cantidad":.... }], "Total":....., "Mensaje": "Mensaje de respuesta ó Error", }

- El método GET permitirá devolver toda la información del carrito de compra por el respectivo usuario.
- El método POST permitirá actualizar únicamente la "Cantidad" de productos a solicitar, validando la disponibilidad del producto.
- El método DELETE Permitirá eliminar los productos del carrito de compra, actualizando el "Total" a pagar.
- Cuando un producto se encuentra en el carrito de compra del cliente se debe cambiar el estado a reservado, descontando del inventario general dicha cantidad de producto.

Módulo de compra

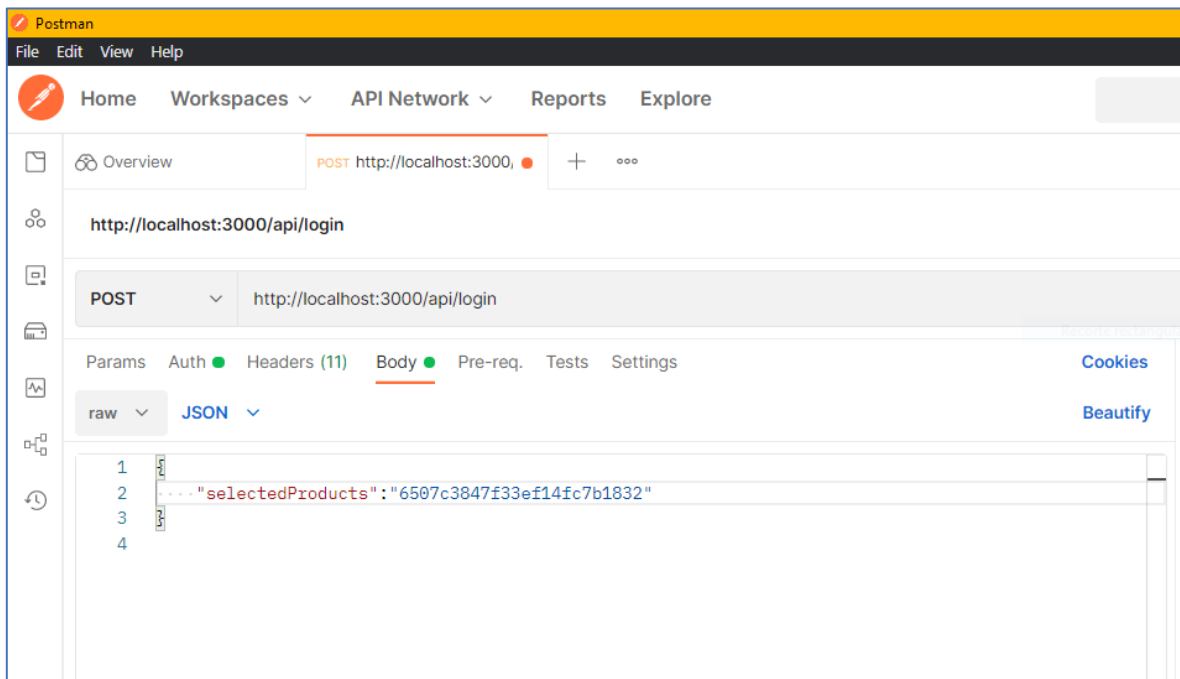
Modulo	Compra
Método	POST
Ruta	/api/compra
Encabezado	{ "Token":.... }
Cuerpo	{ }
Respuesta	{ "Mensaje": "Mensaje de respuesta ó Error", }

- Validará que el token corresponda al usuario indicado.
- Se descontarán la cantidad de productos en el inventario.
- Se creará una bitácora de compras por usuario.

Para la consulta de los endpoints puedes utilizar el siguiente programa con la siguiente ruta:

Postman

Get: <http://localhost:3000/api/login>

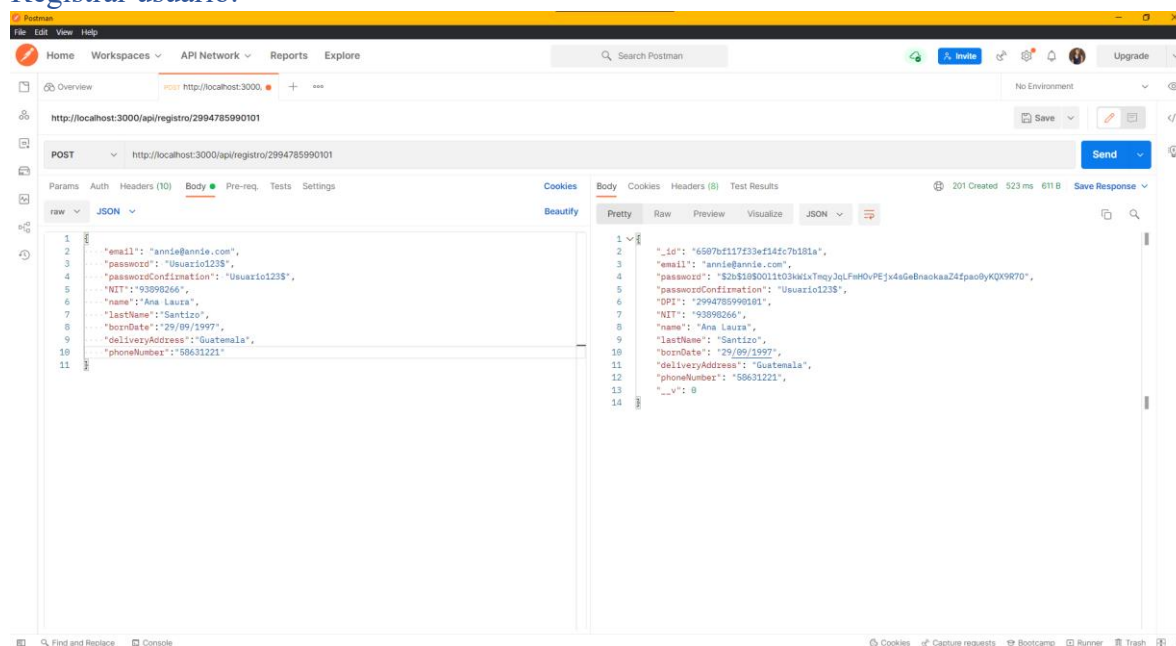


Términos y Condiciones de Uso

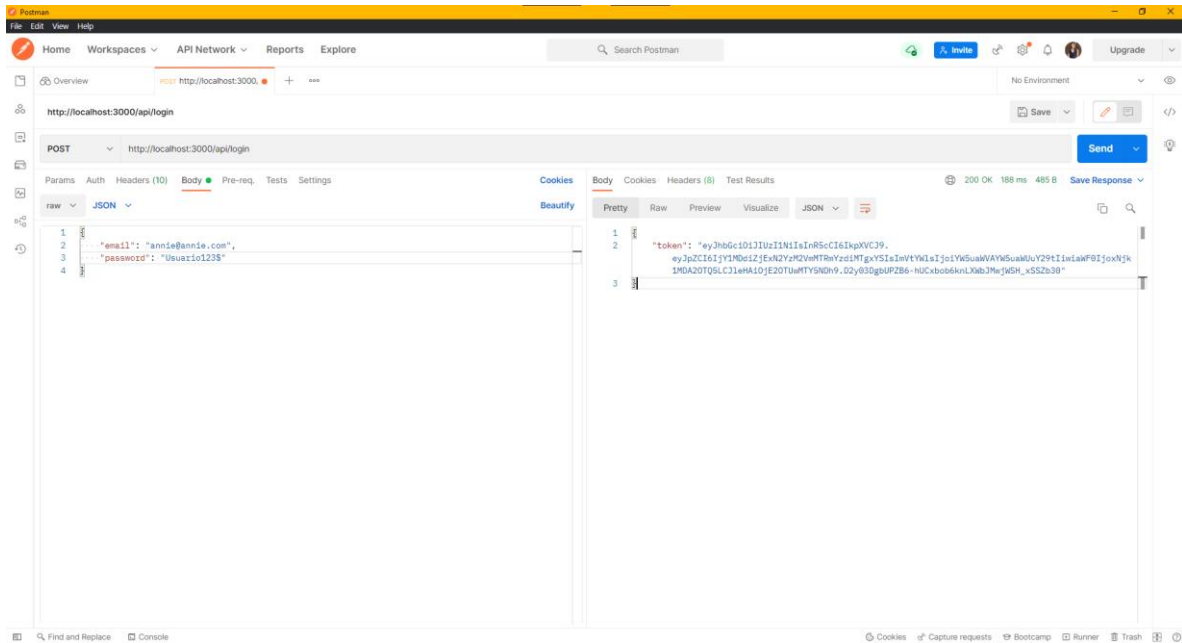
Nos reservamos el derecho de modificar estos Términos en cualquier momento. Cualquier modificación será efectiva inmediatamente después de su publicación en este sitio. Es su responsabilidad revisar periódicamente estos términos para estar al tanto de las modificaciones.

PANTALLAS DEL PROCESO PARA UTILIZACIÓN DE LOS ENDPOINTS

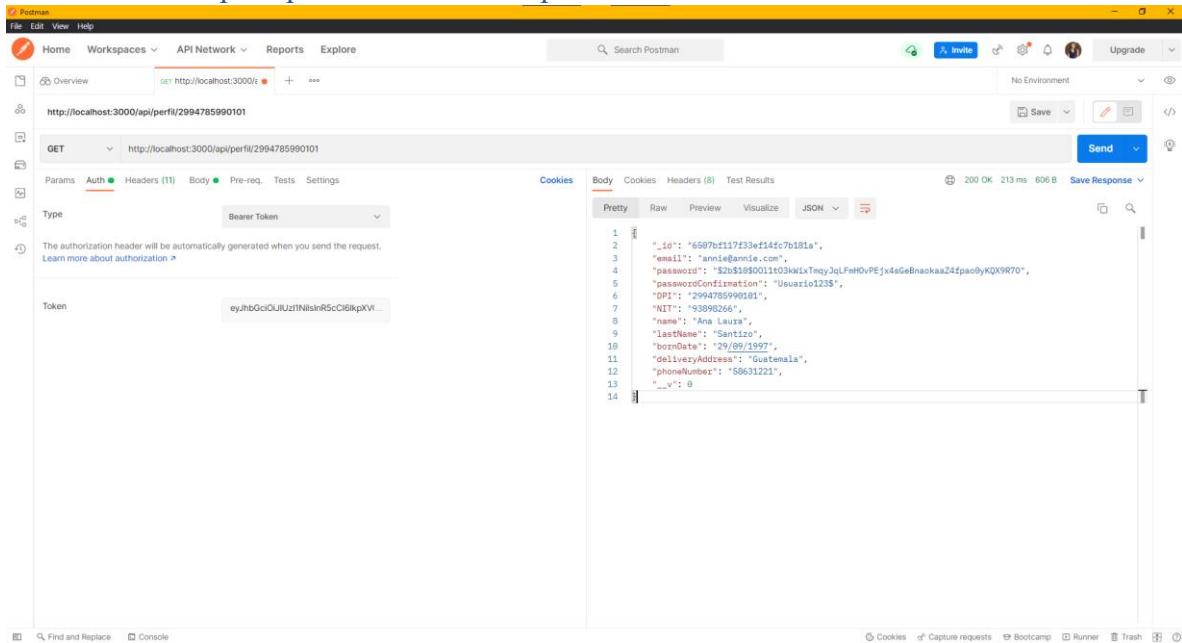
Registrar usuario:



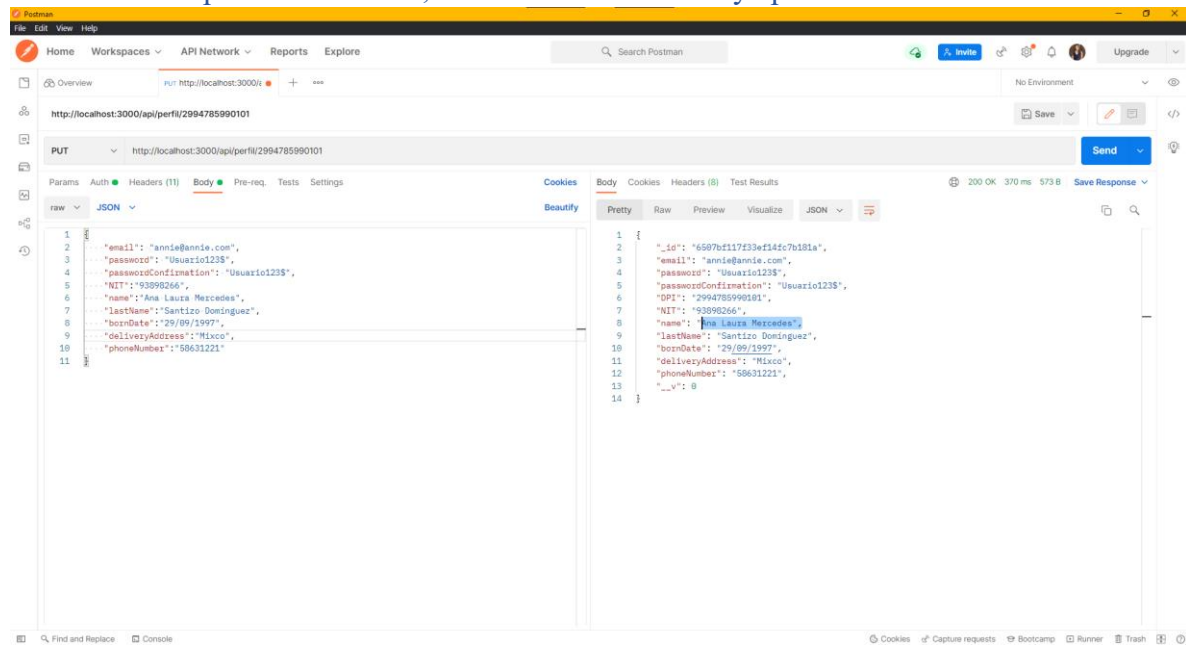
Iniciar sesión recibe token:



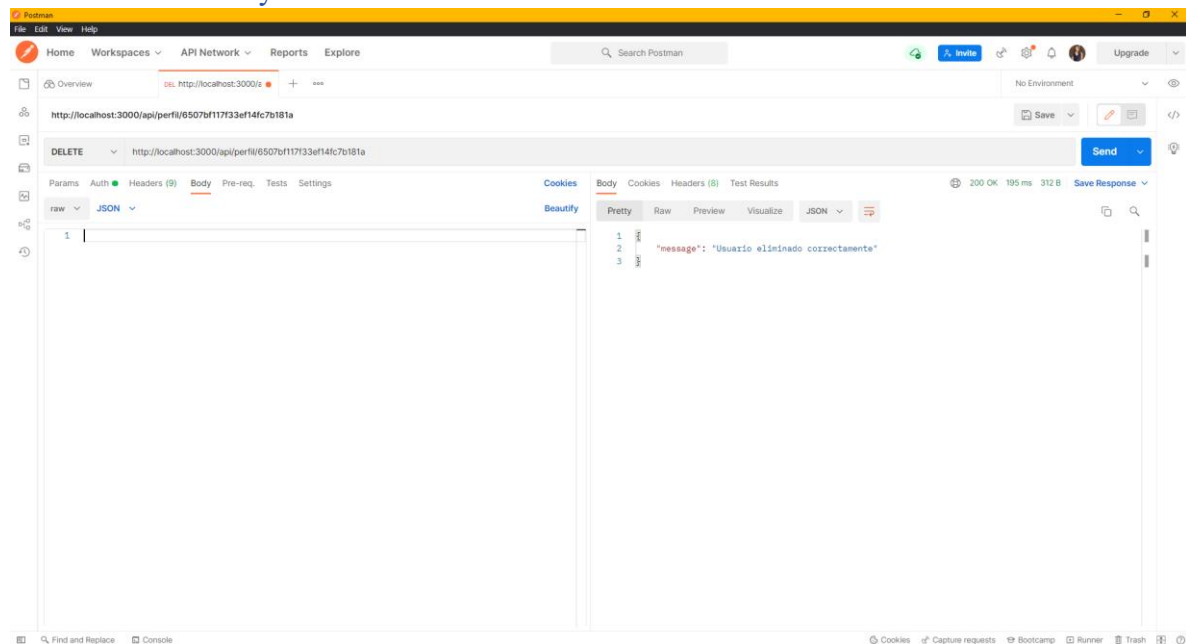
Obtener usuario por dpi con bearer token para autenticar



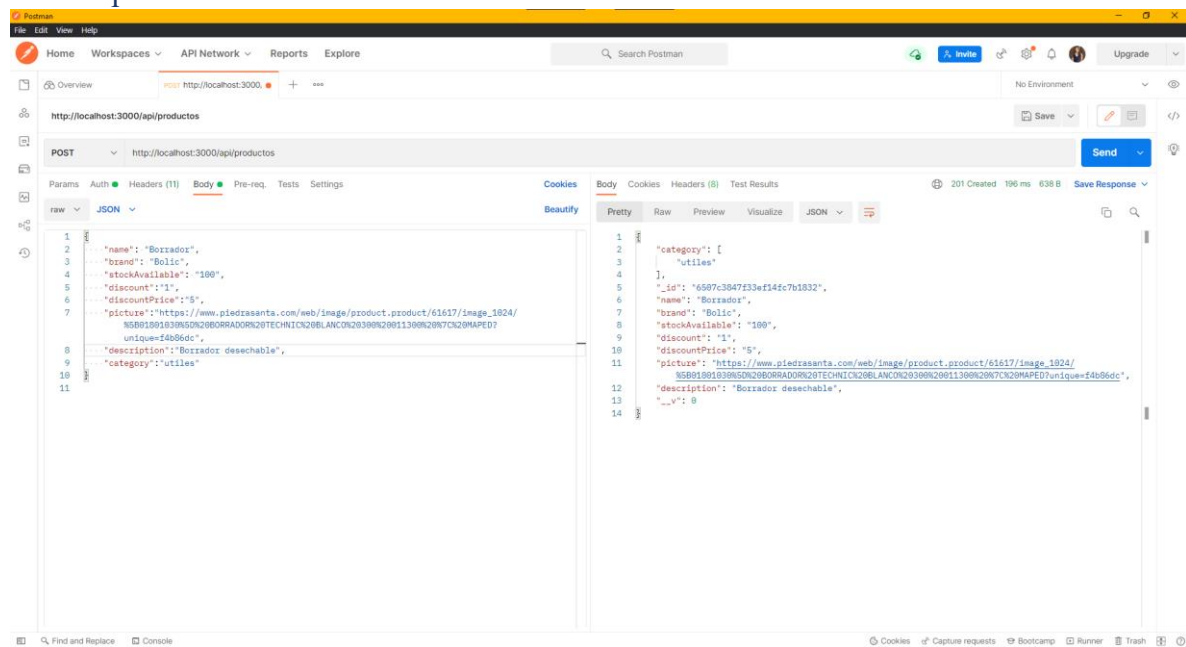
Se modifica el perfil del usuario, se modificó el nombre y apellido



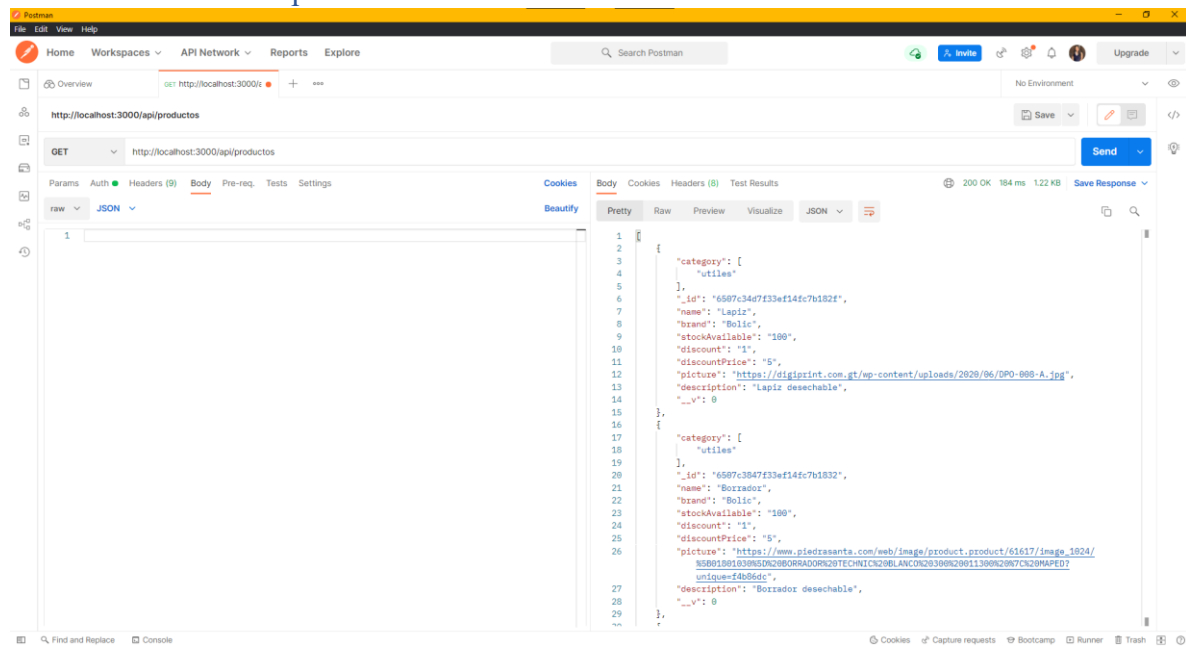
Se eliminó usuario y el token se elimina tambien



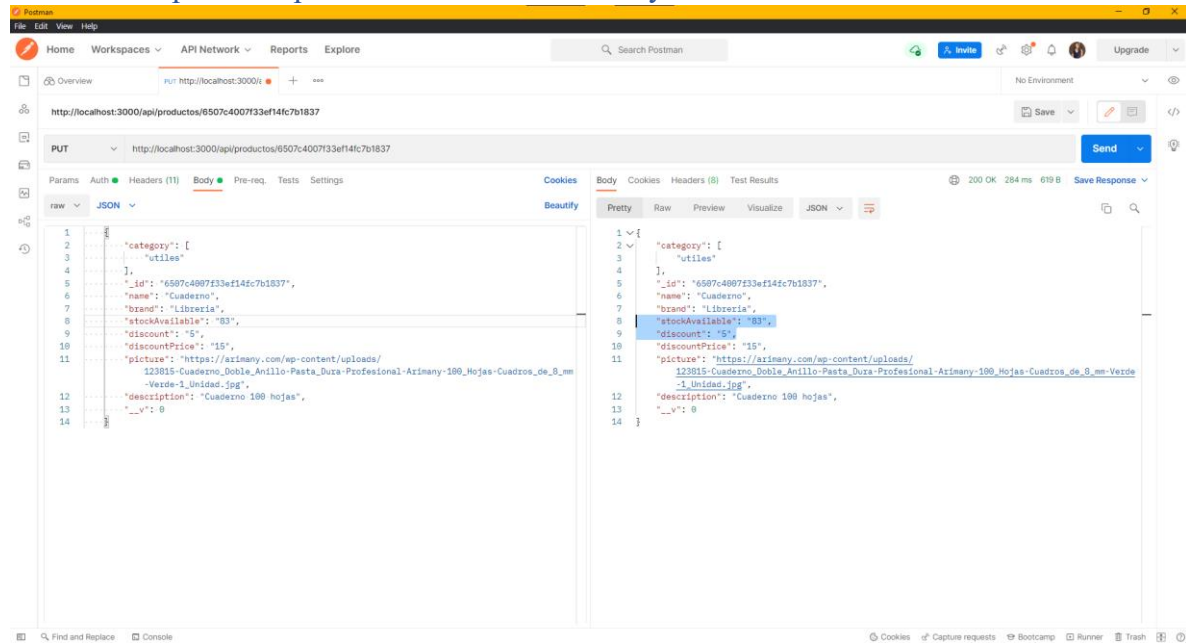
Se crea producto nuevo



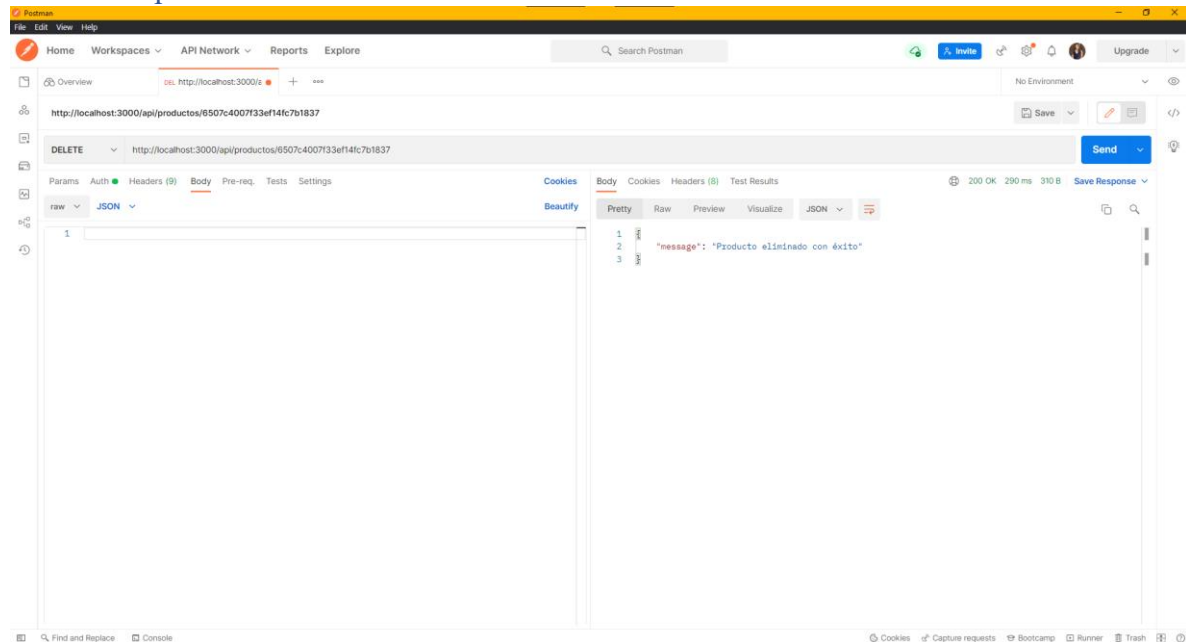
Se obtienen todos los productos



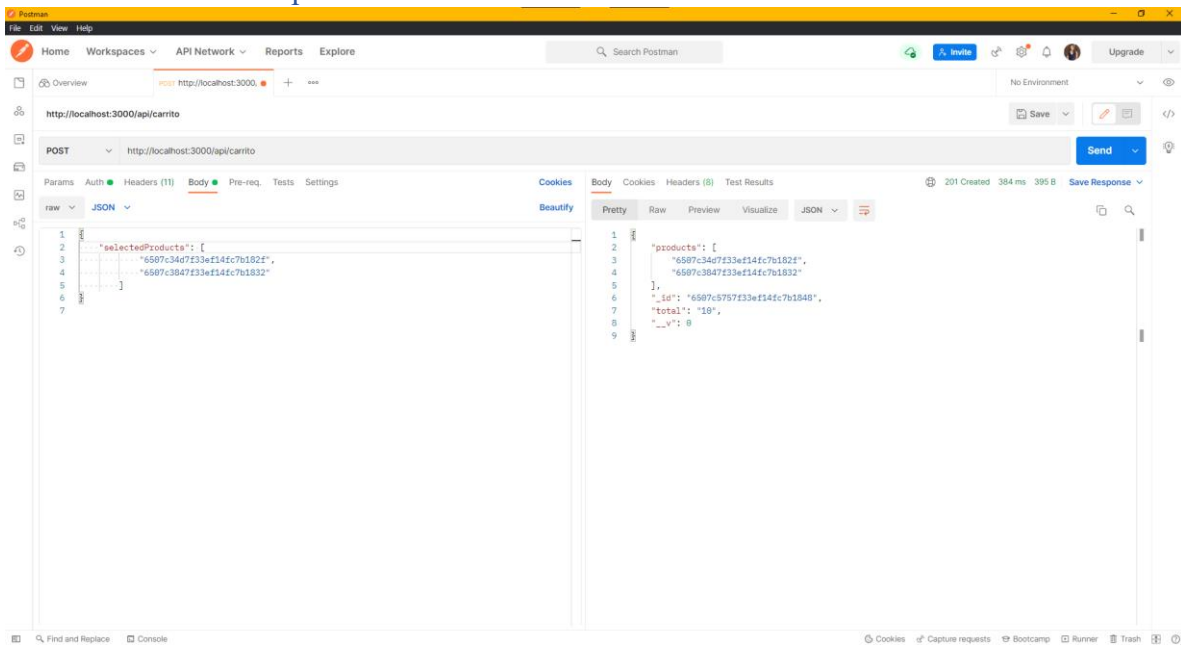
Se modifica producto por id se cambia descuento y stock



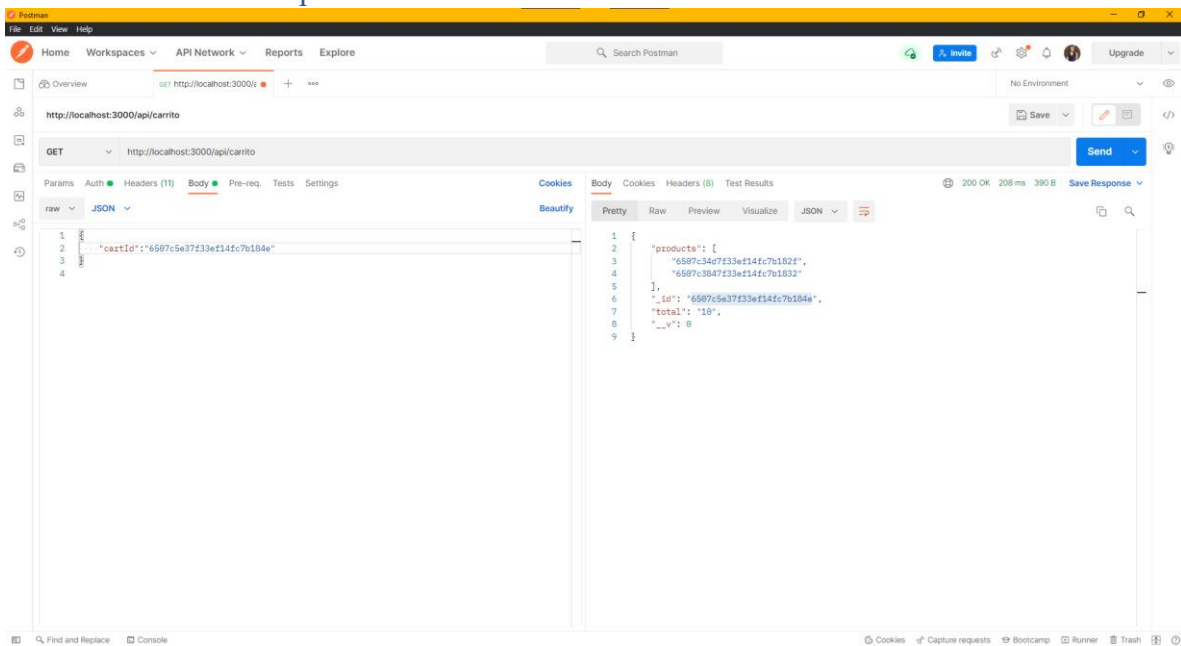
Se elimina producto



Se crea carrito de compras



Obtiene carrito de compras



Borra producto del carrito

Postman interface showing a DELETE request to `http://localhost:3000/api/carrito`. The request body is JSON:

```
{  "cartId": "6597c5e37f33ef14fc7b184e",  "productId": "6597c3407f33ef14fc7b182f"}
```

The response is 200 OK with a JSON body:

```
{  "products": [    {      "id": "6597c3047f33ef14fc7b1832"    }  ],  "total": "5",  "_v": 1}
```

Genera compra y crea bitácora de compra para usuario

Postman interface showing a POST request to `http://localhost:3000/api/compra`. The request body is JSON:

```
{  "selectedProducts": "6597c3047f33ef14fc7b1832"}
```

The response is 200 OK with a JSON body:

```
{  "message": "Compra realizada con éxito",  "purchaseLog": {    "products": [      {        "id": "6597c3047f33ef14fc7b1832"      }    ],    "user": {      "id": "6597c6c6d8a8b28887d9545",      "userId": "6597c2417f33ef14fc7b1828",      "purchaseDate": "2023-09-18T03:40:54.836Z",      "_v": 0    }  } }
```