INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

Proyecto: Avance 2

Des. de Aplic. Web client and backend development frameworks

Alumno:
ANGELO MIHAELLE OJEDA GÓMEZ

Profesor:
M. en C. Jose Asuncion ENRIQUEZ ZARATE

November 25, 2024

Contents

1 Introducción

La página web actual de TEOH enfrenta problemas de diseño, navegación y rendimiento que afectan la experiencia del usuario y la efectividad para atraer clientes potenciales. Este proyecto propone un rediseño integral y la implementación de un backend robusto con APIs para gestionar entidades clave.

2 Desarrollo

2.1 Entidades y APIs

2.2 Usuario

La entidad **Usuario** representa a las personas que interactúan con el sitio web, incluyendo tanto a los clientes como a los administradores. Su propósito es gestionar la información del usuario, como el nombre, correo electrónico y preferencias. Esta gestión facilita la autenticación y personalización de la experiencia del usuario en la plataforma, permitiendo el acceso a la tienda, el historial de compras y otras funcionalidades.

2.2.1 Modelo

```java
package com.teoh.api.model;

import jakarta.persistence.*;
import lombok.Data;
import io.swagger.v3.oas.annotations.media.Schema;

@Data
@Entity
@Table(name = "usuarios")
@Schema(description = "Modelo que representa a un usuario en TEOH, incluyendo tanto a
↪  clientes como a administradores")
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "ID único del usuario", example = "1")
    private Integer id;

    @Schema(description = "Nombre completo del usuario", example = "Juan Pérez")
    private String nombre;

    @Column(unique = true, nullable = false)
    @Schema(description = "Correo electrónico único del usuario", example =
    ↪  "juan.perez@teoh.com")
    private String correoElectronico;

    @Schema(description = "Contraseña del usuario (encriptada)", example = "********")
    private String contrasena;

    @Enumerated(EnumType.STRING)
    @Schema(description = "Rol del usuario en el sistema", example = "CLIENTE")
    private Rol rol = Rol.CLIENTE;

    @Lob
```

```
    @Schema(description = "Historial de compras del usuario", example = "Compra 1: Producto
    ↪  A, Compra 2: Producto B")
    private String historialCompras;

    @Lob
    @Schema(description = "Preferencias del usuario", example = "Preferencia 1: Entregas por
    ↪  la mañana, Preferencia 2: Descuento en productos para el cuidado de heridas")
    private String preferencias;

    public enum Rol {
        CLIENTE, ADMIN
    }
}
```

2.2.2 Controlador

```
package com.teoh.api.controller;

import com.teoh.api.model.Usuario;
import com.teoh.api.service.UsuarioService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/usuarios")
@Tag(name = "Usuarios", description = "Operaciones relacionadas con la gestión de usuarios")
public class UsuarioController {

    private final UsuarioService usuarioService;

    public UsuarioController(UsuarioService usuarioService) {
        this.usuarioService = usuarioService;
    }

    @GetMapping
    @Operation(summary = "Obtener todos los usuarios",
               description = "Devuelve una lista de todos los usuarios registrados en el
               ↪  sistema.")
    public List<Usuario> getAllUsuarios() {
        System.out.println("Fetching all usuarios...");
        return usuarioService.findAll();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Obtener un usuario por ID",
               description = "Devuelve los detalles de un usuario específico identificado
               ↪  por su ID.",
               parameters = @Parameter(name = "id", description = "ID del usuario", required
               ↪  = true))
    public Usuario getUsuarioById(@PathVariable Integer id) {
        return usuarioService.findById(id);
    }
```

```
    @PostMapping
    @Operation(summary = "Crear un nuevo usuario",
               description = "Permite crear un nuevo usuario proporcionando los datos en el
               ↪  cuerpo de la solicitud.")
    public Usuario createUsuario(@RequestBody Usuario usuario) {
        return usuarioService.save(usuario);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Actualizar un usuario existente",
               description = "Actualiza la información de un usuario identificado por su
               ↪  ID.",
               parameters = @Parameter(name = "id", description = "ID del usuario", required
               ↪  = true))
    public Usuario updateUsuario(@PathVariable Integer id, @RequestBody Usuario usuario) {
        usuario.setId(id);
        return usuarioService.save(usuario);
    }

    @DeleteMapping("/{id}")
    @Operation(summary = "Eliminar un usuario",
               description = "Elimina un usuario específico identificado por su ID.",
               parameters = @Parameter(name = "id", description = "ID del usuario", required
               ↪  = true))
    public void deleteUsuario(@PathVariable Integer id) {
        usuarioService.deleteById(id);
    }
}
```

2.3 Producto

La entidad **Producto** abarca todos los artículos disponibles para la compra en la tienda de TEOH, como suministros para el tratamiento de heridas y ostomías. Su propósito es proporcionar información detallada sobre cada producto, incluyendo su nombre, descripción, precio y disponibilidad. Esta información permite a los usuarios buscar y adquirir productos específicos de manera eficiente.

2.3.1 Modelo

```
package com.teoh.api.model;

import jakarta.persistence.*;
import lombok.Data;
import io.swagger.v3.oas.annotations.media.Schema;

@Data
@Entity
@Table(name = "productos")
@Schema(description = "Modelo que representa un producto disponible para compra en TEOH")
public class Producto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "ID único del producto", example = "1")
    private Integer id;
```

```
    @Schema(description = "Nombre del producto", example = "Venda para heridas")
    private String nombre;

    @Lob
    @Schema(description = "Descripción detallada del producto", example = "Esta venda es
    ↪  ideal para el tratamiento de heridas menores...")
    private String descripcion;

    @Schema(description = "Categoría del producto", example = "Suministros médicos")
    private String categoria;

    @Schema(description = "Precio del producto", example = "29.99")
    private Double precio;

    @Schema(description = "Cantidad de stock disponible", example = "150")
    private Integer stock;

    @Schema(description = "Imagen del producto", example = "base64_encoded_image_data_here")
    private String imagen;

    @Schema(description = "Valoración promedio del producto por los usuarios", example =
    ↪  "4.5")
    private Float valoracion;
}
```

2.3.2 Controlador

```
package com.teoh.api.controller;

import com.teoh.api.model.Producto;
import com.teoh.api.service.ProductoService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/productos")
@Tag(name = "Productos", description = "Operaciones relacionadas con la gestión de
↪  productos")
public class ProductoController {

    private final ProductoService productoService;

    public ProductoController(ProductoService productoService) {
        this.productoService = productoService;
    }

    @GetMapping
    @Operation(summary = "Obtener todos los productos",
               description = "Devuelve una lista de todos los productos disponibles en la
               ↪  tienda.")
    public List<Producto> getAllProductos() {
```

```
        return productoService.findAll();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Obtener un producto por ID",
               description = "Devuelve los detalles de un producto específico identificado
               ↪  por su ID.",
               parameters = @Parameter(name = "id", description = "ID del producto",
               ↪  required = true))
    public Producto getProductoById(@PathVariable Integer id) {
        return productoService.findById(id);
    }

    @PostMapping
    @Operation(summary = "Crear un nuevo producto",
               description = "Permite crear un nuevo producto proporcionando los detalles en
               ↪  el cuerpo de la solicitud.")
    public Producto createProducto(@RequestBody Producto producto) {
        return productoService.save(producto);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Actualizar un producto existente",
               description = "Actualiza los detalles de un producto identificado por su
               ↪  ID.",
               parameters = @Parameter(name = "id", description = "ID del producto",
               ↪  required = true))
    public Producto updateProducto(@PathVariable Integer id, @RequestBody Producto producto)
    ↪  {
        producto.setId(id);
        return productoService.save(producto);
    }

    @DeleteMapping("/{id}")
    @Operation(summary = "Eliminar un producto",
               description = "Elimina un producto identificado por su ID.",
               parameters = @Parameter(name = "id", description = "ID del producto",
               ↪  required = true))
    public void deleteProducto(@PathVariable Integer id) {
        productoService.deleteById(id);
    }
}
```

2.4 Pedido

La entidad **Pedido** representa los pedidos realizados por los usuarios al comprar productos a través del sitio web. Su propósito es registrar y gestionar la información relacionada con cada pedido, como el estado, la fecha y los detalles de envío. Esta gestión permite un seguimiento efectivo de las transacciones y mejora la experiencia del cliente al ofrecer información sobre su compra.

2.4.1 Modelo

```
package com.teoh.api.model;

import jakarta.persistence.*;
```

```
import lombok.Data;
import io.swagger.v3.oas.annotations.media.Schema;
import java.util.Date;

@Data
@Entity
@Table(name = "pedidos")
@Schema(description = "Modelo que representa un pedido realizado por un usuario en TEOH")
public class Pedido {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "ID único del pedido", example = "1")
    private Integer id;

    @ManyToOne
    @Schema(description = "Usuario que realiza el pedido", implementation = Usuario.class)
    private Usuario usuario;

    @Temporal(TemporalType.DATE)
    @Schema(description = "Fecha en que se realizó el pedido", example = "2024-11-24")
    private Date fechaPedido;

    @Enumerated(EnumType.STRING)
    @Schema(description = "Estado actual del pedido", example = "EN_PROCESO")
    private EstadoPedido estadoPedido = EstadoPedido.EN_PROCESO;

    @Schema(description = "Total del pedido en valor monetario", example = "150.75")
    private Double totalPedido;

    @Lob
    @Schema(description = "Detalles adicionales de envío, como dirección, instrucciones
    ↪  especiales", example = "Dirección: Calle Ficticia 123, Instrucciones: Entregar por
    ↪  la tarde.")
    private String detallesEnvio;

    public enum EstadoPedido {
        EN_PROCESO, ENVIADO, ENTREGADO
    }
}
```

2.4.2 Controlador

```
package com.teoh.api.controller;

import com.teoh.api.model.Pedido;
import com.teoh.api.service.PedidoService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/pedidos")
```

```java
@Tag(name = "Pedidos", description = "Operaciones relacionadas con la gestión de pedidos")
public class PedidoController {

    private final PedidoService pedidoService;

    public PedidoController(PedidoService pedidoService) {
        this.pedidoService = pedidoService;
    }

    @GetMapping
    @Operation(summary = "Obtener todos los pedidos",
               description = "Devuelve una lista de todos los pedidos realizados.")
    public List<Pedido> getAllPedidos() {
        return pedidoService.findAll();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Obtener un pedido por ID",
               description = "Devuelve los detalles de un pedido específico identificado por
               ↪  su ID.",
               parameters = @Parameter(name = "id", description = "ID del pedido", required
               ↪  = true))
    public Pedido getPedidoById(@PathVariable Integer id) {
        return pedidoService.findById(id);
    }

    @PostMapping
    @Operation(summary = "Crear un nuevo pedido",
               description = "Permite crear un nuevo pedido proporcionando los detalles
               ↪  necesarios en el cuerpo de la solicitud.")
    public Pedido createPedido(@RequestBody Pedido pedido) {
        return pedidoService.save(pedido);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Actualizar un pedido existente",
               description = "Actualiza la información de un pedido específico identificado
               ↪  por su ID.",
               parameters = @Parameter(name = "id", description = "ID del pedido", required
               ↪  = true))
    public Pedido updatePedido(@PathVariable Integer id, @RequestBody Pedido pedido) {
        pedido.setId(id);
        return pedidoService.save(pedido);
    }

    @DeleteMapping("/{id}")
    @Operation(summary = "Eliminar un pedido",
               description = "Elimina un pedido específico identificado por su ID.",
               parameters = @Parameter(name = "id", description = "ID del pedido", required
               ↪  = true))
    public void deletePedido(@PathVariable Integer id) {
        pedidoService.deleteById(id);
    }
}
```

2.5 Caso

La entidad **Caso** documenta casos de éxito y testimonios de pacientes que han recibido tratamiento con los productos ofrecidos en el sitio. Su propósito es proporcionar evidencia social y mostrar la efectividad de los productos. Esto ayuda a construir confianza y credibilidad en la marca, al mismo tiempo que ofrece a los visitantes información valiosa sobre experiencias de otros usuarios.

2.5.1 Modelo

```java
package com.teoh.api.model;

import jakarta.persistence.*;
import lombok.Data;
import io.swagger.v3.oas.annotations.media.Schema;
import java.util.Date;

@Data
@Entity
@Schema(description = "Modelo que representa los casos de éxito en el sitio TEOH")
public class Caso {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "ID único del caso", example = "1")
    private Integer id;

    @Schema(description = "Título del caso", example = "Recuperación exitosa de una herida
    ↪  severa")
    private String titulo;

    @Lob
    @Schema(description = "Descripción detallada del caso", example = "Este caso muestra el
    ↪  tratamiento exitoso de una herida severa...")
    private String descripcion;

    @Temporal(TemporalType.DATE)
    @Schema(description = "Fecha en la que se documentó el caso", example = "2024-11-24")
    private Date fecha;

    @Lob
    @Schema(description = "Imágenes relacionadas con el caso, pueden ser múltiples", example
    ↪  = "base64_encoded_image_data_here")
    private String imagenes;

    @Lob
    @Schema(description = "Testimonio del paciente o persona involucrada en el caso",
    ↪  example = "Gracias al tratamiento de TEOH, pude recuperar completamente mi
    ↪  bienestar...")
    private String testimonio;

    @ManyToOne
    @Schema(description = "Usuario asociado con el caso (paciente o cuidador)",
    ↪  implementation = Usuario.class)
    private Usuario usuario;
}
```

2.5.2 Controlador

```java
package com.teoh.api.controller;

import com.teoh.api.model.Caso;
import com.teoh.api.service.CasoService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/casos")
@Tag(name = "Casos", description = "Operaciones relacionadas con casos de éxito y
↪  testimonios")
public class CasoController {

    private final CasoService casoService;

    public CasoController(CasoService casoService) {
        this.casoService = casoService;
    }

    @GetMapping
    @Operation(summary = "Obtener todos los casos",
               description = "Devuelve una lista de todos los casos de éxito y testimonios
               ↪  almacenados.")
    public List<Caso> getAllCasos() {
        return casoService.findAll();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Obtener un caso por ID",
               description = "Devuelve los detalles de un caso específico identificado por
               ↪  su ID.",
               parameters = @Parameter(name = "id", description = "ID del caso", required =
               ↪  true))
    public Caso getCasoById(@PathVariable Integer id) {
        return casoService.findById(id);
    }

    @PostMapping
    @Operation(summary = "Crear un nuevo caso",
               description = "Permite agregar un nuevo caso de éxito o testimonio
               ↪  proporcionando los detalles en el cuerpo de la solicitud.")
    public Caso createCaso(@RequestBody Caso caso) {
        return casoService.save(caso);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Actualizar un caso existente",
               description = "Actualiza los detalles de un caso específico identificado por
               ↪  su ID.",
```

```
            parameters = @Parameter(name = "id", description = "ID del caso", required =
            ↪ true))
    public Caso updateCaso(@PathVariable Integer id, @RequestBody Caso caso) {
        caso.setId(id);
        return casoService.save(caso);
    }

    @DeleteMapping("/{id}")
    @Operation(summary = "Eliminar un caso",
            description = "Elimina un caso de éxito o testimonio identificado por su
            ↪ ID.",
            parameters = @Parameter(name = "id", description = "ID del caso", required =
            ↪ true))
    public void deleteCaso(@PathVariable Integer id) {
        casoService.deleteById(id);
    }
}
```

2.6 Artículo del Blog

La entidad **Artículo del Blog** almacena artículos y publicaciones que abordan temas relacionados con la salud, el cuidado de ostomías y heridas, así como consejos prácticos. Su propósito es educar y proporcionar contenido útil a los usuarios, mejorando su conocimiento sobre los productos y tratamientos disponibles. Los artículos del blog también contribuyen a la optimización en motores de búsqueda (SEO) y atraen tráfico al sitio web.

2.6.1 Modelo

```
package com.teoh.api.model;

import jakarta.persistence.*;
import lombok.Data;
import io.swagger.v3.oas.annotations.media.Schema;
import java.util.Date;

@Data
@Entity
@Table(name = "articulos_del_blog")
@Schema(description = "Modelo que representa los artículos del blog del sitio TEOH")
public class ArticuloDelBlog {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "ID único del artículo del blog", example = "1")
    private Integer id;

    @Schema(description = "Título del artículo", example = "Tratamiento de heridas de
    ↪ ostomía")
    private String titulo;

    @Lob
    @Schema(description = "Contenido completo del artículo", example = "El artículo describe
    ↪ el proceso de cuidado de las heridas...")
    private String contenido;
```

```
    @Temporal(TemporalType.DATE)
    @Schema(description = "Fecha de publicación del artículo", example = "2024-11-24")
    private Date fechaPublicacion;

    @ManyToOne
    @Schema(description = "Autor del artículo del blog", implementation = Usuario.class)
    private Usuario autor;

    @Schema(description = "Categoría a la que pertenece el artículo", example = "Salud")
    private String categoria;

    @Lob
    @Schema(description = "Etiquetas relacionadas con el artículo", example = "cuidado,
    ↪  ostomía, heridas")
    private String etiquetas;
}
```

2.6.2 Controlador

```
package com.teoh.api.controller;

import com.teoh.api.model.ArticuloDelBlog;
import com.teoh.api.service.ArticuloDelBlogService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/articulos")
@Tag(name = "Artículos del Blog", description = "Operaciones relacionadas con los artículos
↪  del blog")
public class ArticuloDelBlogController {

    private final ArticuloDelBlogService articuloDelBlogService;

    public ArticuloDelBlogController(ArticuloDelBlogService articuloDelBlogService) {
        this.articuloDelBlogService = articuloDelBlogService;
    }

    @GetMapping
    @Operation(summary = "Obtener todos los artículos",
               description = "Devuelve una lista de todos los artículos del blog
               ↪  disponibles.")
    public List<ArticuloDelBlog> getAllArticulos() {
        return articuloDelBlogService.findAll();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Obtener un artículo por ID",
               description = "Devuelve un artículo específico del blog identificado por su
               ↪  ID.",
               parameters = @Parameter(name = "id", description = "ID del artículo",
               ↪  required = true))
```

```
    public ArticuloDelBlog getArticuloById(@PathVariable Integer id) {
        return articuloDelBlogService.findById(id);
    }

    @PostMapping
    @Operation(summary = "Crear un nuevo artículo",
               description = "Crea un nuevo artículo del blog con la información
               ↪  proporcionada en el cuerpo de la solicitud.")
    public ArticuloDelBlog createArticulo(@RequestBody ArticuloDelBlog articulo) {
        return articuloDelBlogService.save(articulo);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Actualizar un artículo existente",
               description = "Actualiza un artículo del blog existente identificado por su
               ↪  ID con los nuevos datos proporcionados.",
               parameters = @Parameter(name = "id", description = "ID del artículo",
               ↪  required = true))
    public ArticuloDelBlog updateArticulo(@PathVariable Integer id, @RequestBody
    ↪  ArticuloDelBlog articulo) {
        articulo.setId(id);
        return articuloDelBlogService.save(articulo);
    }

    @DeleteMapping("/{id}")
    @Operation(summary = "Eliminar un artículo",
               description = "Elimina un artículo del blog identificado por su ID.",
               parameters = @Parameter(name = "id", description = "ID del artículo",
               ↪  required = true))
    public void deleteArticulo(@PathVariable Integer id) {
        articuloDelBlogService.deleteById(id);
    }
}
```

2.7 Verificación del Funcionamiento

A continuación se incluyen capturas de pantalla de las pruebas realizadas con Postman para verificar el correcto funcionamiento de cada API:

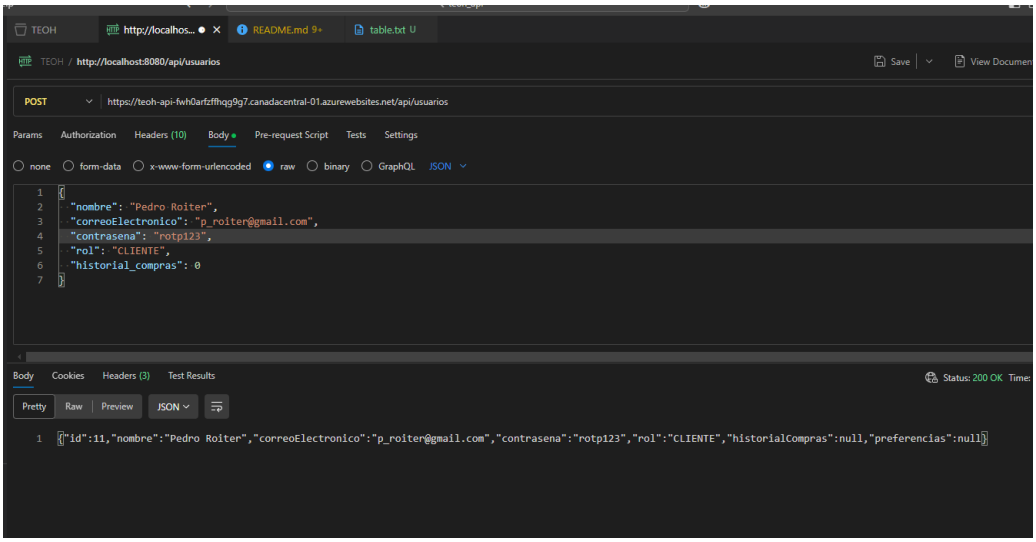

Figure 1: Verificación del endpoint para Usuarios (POST).

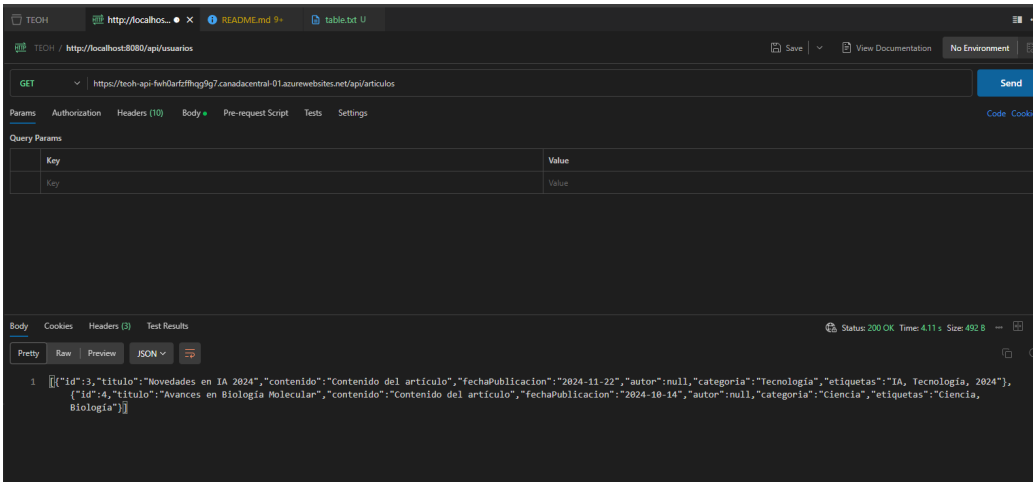

Figure 2: Verificación del endpoint para Articulos (GET).

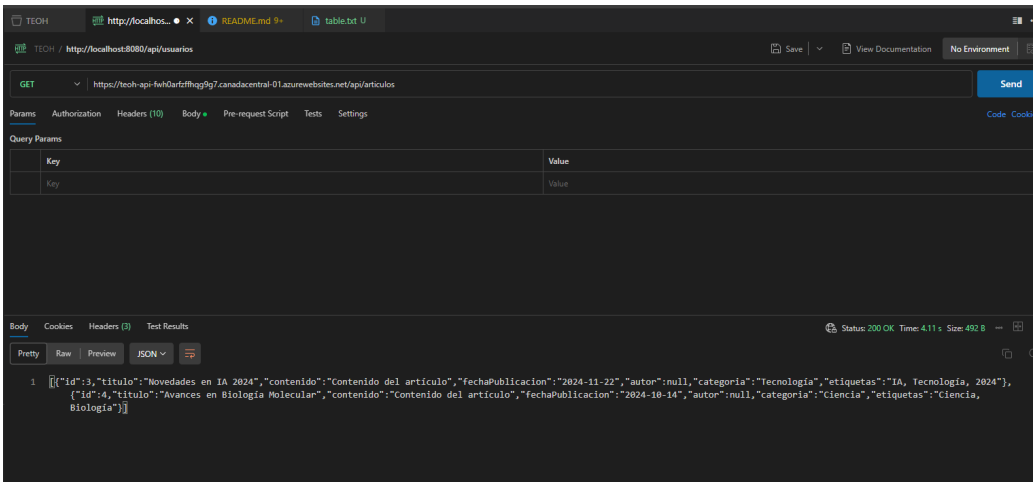

Figure 3: Verificación del endpoint para Casos (GET by ID).

2.8 Enlaces de Revisión

- URL del Repositorio del Proyecto: `https://github.com/AngeloMihaelle/teoh_api`
- URL del Api publicada: `https://teoh-api-fwh0arfzffhqg9g7.canadacentral-01.azurewebsites.net/`
- URL de la Base de Datos: `https://supabase.com/dashboard/project/nwzkjueefemwekbwttge`
  - Usuario: angelo.mihaelle.com@gmail.com
  - Clave: Hola_123_sopa
  - Nombre de la base de datos: teoh_db
  - Clave de la Base de datos: Hola_123_sopa
- URL de la Documentación Swagger: `https://teoh-api-fwh0arfzffhqg9g7.canadacentral-01.azurewebsites.net/swagger-ui/swagger-ui/index.html`

3 Conclusiones

El desarrollo de este proyecto tiene un impacto significativo en la mejora de la experiencia de usuario y en la optimización del sitio web de TEOH. Al implementar un diseño renovado y APIs robustas, se logra:

- Mejorar la accesibilidad y navegación del sitio.
- Aumentar la eficiencia en la gestión de datos críticos como usuarios, productos y pedidos.
- Fortalecer la confianza de los clientes al ofrecer contenido útil y una plataforma confiable.

Este enfoque integral asegura un servicio más profesional, posicionando a TEOH como un referente en su sector.