Integracion de Aplicaciones - NRC 2753

EVALUACIÓN FINAL

ISIL

2024-10

**Profesor:**

**Castañeda Alban, Jorge Eduardo**

**Desarrollo de Software**

**Alumnos**:

**Molfino Caletti, Piero 70366399**

Indices de contenidos

[Introducción: 2](#_Toc172639458)

[Resumen: 3](#_Toc172639459)

[Marco Teórico: 4](#_Toc172639460)

[Descripción del caso de estudio: 4](#_Toc172639461)

[Objetivos 7](#_Toc172639462)

[Desarrollo 8](#_Toc172639463)

[i. Especificaciones del Requerimiento de Software 8](#_Toc172639464)

[ii. Análisis de Software 11](#_Toc172639465)

[Recomendaciones 19](#_Toc172639466)

[Conclusiones 20](#_Toc172639467)

[Bibliografía 20](#_Toc172639468)

# Introducción:

La empresa "TechStore", busca modernizar su infraestructura tecnológica para gestionar sus operaciones diarias de manera más eficiente. Con este objetivo en mente, se ha decidido implementar un API RESTful que permita automatizar y simplificar la gestión de productos, clientes, órdenes y categorías.

El API desarrollado proporcionará a TechStore una serie de funcionalidades clave:

1. **Gestión de Productos**: Permitirá añadir nuevos productos, consultar detalles de productos específicos, actualizar la información de productos existentes y eliminar aquellos que ya no estén disponibles o hayan sido descontinuados.
2. **Gestión de Clientes**: Facilitará el registro de nuevos clientes, la consulta de información detallada de clientes, la actualización de sus datos y la eliminación de registros de clientes que ya no realicen compras en la tienda.
3. **Gestión de Órdenes**: Permitirá registrar nuevas órdenes de compra, consultar detalles de órdenes específicas, actualizar el estado de las órdenes y eliminar registros de órdenes canceladas o erróneas.
4. **Gestión de Categorías**: Permitirá añadir nuevas categorías de productos, consultar todas las categorías disponibles, actualizar la información de categorías existentes y eliminar aquellas que ya no se utilizan.

Este API está diseñado para integrarse de manera flexible y escalable con las operaciones de TechStore, garantizando así una mejora significativa en la eficiencia y efectividad de sus procesos. El presente informe detalla el desarrollo del API, incluyendo las especificaciones de cada funcionalidad, las tecnologías utilizadas y las pruebas realizadas para asegurar su correcto funcionamiento.

# Resumen:

En el presente informe se documenta el desarrollo de un API RESTful para "TechStore", una empresa especializada en la venta de productos electrónicos como computadoras, teléfonos móviles, tabletas y accesorios. Con el objetivo de modernizar su infraestructura tecnológica y mejorar la eficiencia de sus operaciones diarias, se ha implementado un sistema que permite gestionar productos, clientes, órdenes y categorías.

El API proporciona las siguientes funcionalidades principales:

1. **Gestión de Productos**:
   * Creación de nuevos productos con detalles como nombre, descripción, precio, cantidad en stock y categoría.
   * Consulta de información detallada de productos específicos por su ID.
   * Actualización de información de productos existentes, incluyendo ajustes en el precio y cantidad en stock.
   * Eliminación de productos descontinuados o no disponibles.
2. **Gestión de Clientes**:
   * Registro de nuevos clientes con información como nombre, dirección de correo electrónico, número de teléfono y dirección de envío.
   * Consulta de detalles de clientes específicos por su ID.
   * Actualización de información de clientes, como dirección de envío y número de teléfono.
   * Eliminación de registros de clientes inactivos.
3. **Gestión de Órdenes**:
   * Registro de nuevas órdenes de compra con el ID del cliente, productos comprados, cantidades y total de la orden.
   * Consulta de información de órdenes específicas por su ID.
   * Actualización del estado de las órdenes (por ejemplo, de "pendiente" a "enviado" o "entregado").
   * Eliminación de órdenes canceladas o erróneas.
4. **Gestión de Categorías**:
   * Creación de nuevas categorías de productos.
   * Consulta de todas las categorías disponibles.
   * Actualización de nombre y descripción de categorías existentes.
   * Eliminación de categorías en desuso.

El desarrollo de este API busca proporcionar a TechStore una herramienta flexible, escalable y eficiente para la gestión de sus operaciones comerciales, asegurando una mejor integración y automatización de sus procesos internos.

# Marco Teórico:

1. **API RESTful**
   1. **Escalabilidad:** Gracias a su arquitectura basada en recursos y operaciones, permite un fácil escalado horizontal.
   2. **Flexibilidad:** Se adapta a diversos tipos de clientes y dispositivos, incluyendo navegadores web y aplicaciones móviles.
   3. **Interoperabilidad:** Al utilizar estándares web como HTTP, JSON o XML, facilita la integración con otros sistemas y servicios
2. **Gestión de Recursos**
   1. **Productos**
   2. **Clientes**
   3. **Ordenes**
   4. **Categorías**
3. **Tecnologías Utilizadas:** 
   1. **Node.js y Express.js**
   2. **JWT (Json Web Tokens)**
   3. **Sequalize**
   4. **Postman**
   5. **Phpmyadmin**
   6. **Visual Studio Code**
4. **Buenas Prácticas en el desarrollo de APIs RESTful**
   1. **Uniformidad**
   2. **Documentacion**
   3. **Seguridad**

En conclusión, El desarrollo de un API RESTful para TechStore se fundamenta en principios y tecnologías modernas que buscan mejorar la eficiencia operativa y la experiencia del cliente. Este marco teórico proporciona una base sólida para entender el diseño y la implementación del sistema, asegurando que cumpla con los objetivos de la empresa y responda a las necesidades de sus usuarios

# Descripción del caso de estudio:

**Contexto**

"TechStore" es una empresa comercial especializada en la venta de productos electrónicos. En su búsqueda por mejorar la eficiencia y efectividad de sus operaciones diarias, TechStore ha identificado la necesidad de modernizar su infraestructura tecnológica. La empresa ha decidido implementar un API RESTful para automatizar y optimizar la gestión de sus productos, clientes, órdenes y categorías.

**Objetivos del Proyecto**

El objetivo principal del proyecto es desarrollar un API RESTful que permita a TechStore gestionar sus operaciones de manera más eficiente y eficaz. Los objetivos específicos incluyen:

1. **Mejorar la gestión de inventarios**: Proporcionar una forma eficiente de añadir, actualizar, consultar y eliminar productos.
2. **Optimizar la gestión de clientes**: Facilitar el registro, actualización, consulta y eliminación de clientes.
3. **Automatizar la gestión de órdenes**: Permitir el registro, actualización, consulta y eliminación de órdenes de compra.
4. **Organizar las categorías de productos**: Proveer herramientas para la gestión de categorías, incluyendo la creación, actualización, consulta y eliminación de categorías.

**Alcance del Proyecto**

El proyecto abarca el diseño, desarrollo, implementación y prueba de un API RESTful que cubra las siguientes funcionalidades:

1. **Gestión de Productos**:
   * **Crear Producto**: Permitir a los administradores añadir nuevos productos al inventario con detalles como nombre, descripción, precio, cantidad en stock y categoría.
   * **Consultar Producto**: Proveer la capacidad de obtener información detallada de un producto específico utilizando su ID.
   * **Actualizar Producto**: Facilitar la modificación de la información de un producto existente, incluyendo ajustes en el precio y la cantidad en stock.
   * **Eliminar Producto**: Permitir la eliminación de productos que ya no estén disponibles o que hayan sido descontinuados.
2. **Gestión de Clientes**:
   * **Registrar Cliente**: Añadir nuevos clientes con información detallada como nombre, dirección de correo electrónico, número de teléfono y dirección de envío.
   * **Consultar Cliente**: Obtener detalles de un cliente específico utilizando su ID.
   * **Actualizar Cliente**: Permitir la actualización de la información del cliente, incluyendo la dirección de envío y el número de teléfono.
   * **Eliminar Cliente**: Eliminar registros de clientes que ya no realicen compras en la tienda.
3. **Gestión de Órdenes**:
   * **Crear Orden**: Registrar una nueva orden de compra, incluyendo el ID del cliente, los productos comprados, las cantidades y el total de la orden.
   * **Consultar Orden**: Obtener información detallada de una orden específica utilizando su ID.
   * **Actualizar Estado de Orden**: Modificar el estado de una orden, por ejemplo, de "pendiente" a "enviado" o "entregado".
   * **Eliminar Orden**: Eliminar registros de órdenes canceladas o erróneas.
4. **Gestión de Categorías**:
   * **Crear Categoría**: Añadir nuevas categorías de productos.
   * **Consultar Categorías**: Obtener una lista de todas las categorías disponibles.
   * **Actualizar Categoría**: Modificar el nombre o la descripción de una categoría existente.
   * **Eliminar Categoría**: Eliminar categorías que ya no se utilizan.

**Metodología**

Para llevar a cabo el desarrollo del API, se adoptó una metodología ágil que permite iteraciones rápidas y ajustes continuos basados en la retroalimentación. El proceso incluye:

1. **Recolección de Requisitos**: Reuniones con los stakeholders para definir las necesidades y expectativas.
2. **Diseño del Sistema**: Creación de diagramas de arquitectura y modelos de datos.
3. **Desarrollo**: Implementación del API utilizando tecnologías como Node.js, Express.js, PhPMyAdmin y JWT para la autenticación.
4. **Pruebas**: Realización de pruebas unitarias y de integración para asegurar la funcionalidad y fiabilidad del API.
5. **Despliegue**: Implementación del API en un entorno de producción.
6. **Mantenimiento**: Monitoreo y ajustes continuos para mejorar el rendimiento y resolver problemas.

**Resultados Esperados**

Con la implementación del API RESTful, TechStore espera alcanzar los siguientes resultados:

1. **Mayor eficiencia operativa**: Reducción de errores manuales y tiempos de procesamiento.
2. **Mejor experiencia del cliente**: Servicio más rápido y preciso gracias a una mejor gestión de productos y órdenes.
3. **Escalabilidad**: Capacidad para manejar un mayor volumen de transacciones y datos a medida que la empresa crece.
4. **Integración fácil**: Posibilidad de integrar el API con otras aplicaciones y sistemas internos o externos.

En resumen, el desarrollo de un API RESTful para TechStore representa un paso significativo hacia la modernización de sus operaciones, permitiendo una gestión más eficiente y efectiva de sus recursos y mejorando la experiencia general tanto para los empleados como para los clientes.

# Objetivos

**Objetivo General**

Desarrollar un API RESTful para TechStore que permita gestionar eficientemente las operaciones diarias de la empresa, optimizando la gestión de productos, clientes, órdenes y categorías, y mejorando así la eficacia operativa y la experiencia del cliente.

**Objetivos Específicos**

1. **Mejorar la Gestión de Productos**
   * Implementar funcionalidades para añadir nuevos productos con detalles específicos como nombre, descripción, precio, cantidad en stock y categoría.
   * Proporcionar mecanismos para consultar información detallada de productos individuales mediante su ID.
   * Facilitar la actualización de la información de productos existentes, incluyendo ajustes en el precio y la cantidad en stock.
   * Permitir la eliminación de productos descontinuados o no disponibles en el inventario.
2. **Optimizar la Gestión de Clientes**
   * Desarrollar funcionalidades para registrar nuevos clientes con información completa, incluyendo nombre, dirección de correo electrónico, número de teléfono y dirección de envío.
   * Implementar consultas para obtener detalles de clientes específicos mediante su ID.
   * Permitir la actualización de la información de clientes, como la dirección de envío y el número de teléfono.
   * Facilitar la eliminación de registros de clientes que ya no compran en la tienda.
3. **Automatizar la Gestión de Órdenes**
   * Crear funcionalidades para registrar nuevas órdenes de compra, incluyendo el ID del cliente, los productos comprados, las cantidades y el total de la orden.
   * Proporcionar mecanismos para consultar información detallada de órdenes específicas mediante su ID.
   * Permitir la actualización del estado de las órdenes, por ejemplo, de "pendiente" a "enviado" o "entregado".
   * Facilitar la eliminación de registros de órdenes canceladas o erróneas.
4. **Organizar la Gestión de Categorías**
   * Desarrollar funcionalidades para crear nuevas categorías de productos.
   * Proporcionar mecanismos para consultar todas las categorías disponibles.
   * Facilitar la actualización del nombre y la descripción de categorías existentes.
   * Permitir la eliminación de categorías que ya no se utilizan.
5. **Implementar Seguridad y Autenticación**
   * Incorporar mecanismos de autenticación y autorización, utilizando JWT (JSON Web Tokens) para asegurar que solo los usuarios autorizados puedan acceder y manipular los recursos del API.
   * Asegurar la protección de los datos sensibles mediante la implementación de buenas prácticas de seguridad.
6. **Garantizar la Escalabilidad y Mantenibilidad del Sistema**
   * Diseñar el API de manera que sea fácilmente escalable para manejar un aumento en el volumen de transacciones y datos.
   * Asegurar que el código sea mantenible y extensible, permitiendo futuras actualizaciones y mejoras sin afectar la funcionalidad existente.

Con estos objetivos, el proyecto busca no solo mejorar la eficiencia operativa de TechStore, sino también proporcionar una base tecnológica sólida que permita un crecimiento y adaptación continuos en el dinámico mercado de productos electrónicos.

# Desarrollo

## i. Especificaciones del Requerimiento de Software

**1. Descripción General**

TechStore necesita un API RESTful que facilite la gestión de sus operaciones diarias, incluyendo productos, clientes, órdenes y categorías. El API debe ser seguro, escalable y fácil de integrar con otras aplicaciones y sistemas internos o externos.

**2. Requerimientos Funcionales**

**Gestión de Productos**

* **Crear Producto**:
  + Endpoint: POST /api/productos
  + Descripción: Permite añadir un nuevo producto con detalles como nombre, descripción, precio, cantidad en stock y categoría.
  + Datos de Entrada: JSON con los campos nombre, descripcion, precio, cantidad, categoriaId.
  + Respuesta: Detalles del producto creado, incluyendo su ID.
* **Consultar Producto**:
  + Endpoint: GET /api/productos/{id}
  + Descripción: Obtiene información detallada de un producto específico por su ID.
  + Datos de Entrada: id del producto en la URL.
  + Respuesta: JSON con los detalles del producto.
* **Actualizar Producto**:
  + Endpoint: PUT /api/productos/{id}
  + Descripción: Modifica la información de un producto existente.
  + Datos de Entrada: id del producto en la URL, JSON con los campos a actualizar (nombre, descripcion, precio, cantidad, categoriaId).
  + Respuesta: Detalles del producto actualizado.
* **Eliminar Producto**:
  + Endpoint: DELETE /api/productos/{id}
  + Descripción: Elimina un producto específico por su ID.
  + Datos de Entrada: id del producto en la URL.
  + Respuesta: Confirmación de la eliminación.

**Gestión de Clientes**

* **Registrar Cliente**:
  + Endpoint: POST /api/clientes
  + Descripción: Añade un nuevo cliente con detalles como nombre, dirección de correo electrónico, número de teléfono y dirección de envío.
  + Datos de Entrada: JSON con los campos nombre, email, telefono, direccion.
  + Respuesta: Detalles del cliente creado, incluyendo su ID.
* **Consultar Cliente**:
  + Endpoint: GET /api/clientes/{id}
  + Descripción: Obtiene información detallada de un cliente específico por su ID.
  + Datos de Entrada: id del cliente en la URL.
  + Respuesta: JSON con los detalles del cliente.
* **Actualizar Cliente**:
  + Endpoint: PUT /api/clientes/{id}
  + Descripción: Modifica la información de un cliente existente.
  + Datos de Entrada: id del cliente en la URL, JSON con los campos a actualizar (nombre, email, telefono, direccion).
  + Respuesta: Detalles del cliente actualizado.
* **Eliminar Cliente**:
  + Endpoint: DELETE /api/clientes/{id}
  + Descripción: Elimina un cliente específico por su ID.
  + Datos de Entrada: id del cliente en la URL.
  + Respuesta: Confirmación de la eliminación.

**Gestión de Órdenes**

* **Crear Orden**:
  + Endpoint: POST /api/ordenes
  + Descripción: Registra una nueva orden de compra.
  + Datos de Entrada: JSON con los campos clienteId, productos (lista de objetos con productoId y cantidad), total.
  + Respuesta: Detalles de la orden creada, incluyendo su ID.
* **Consultar Orden**:
  + Endpoint: GET /api/ordenes/{id}
  + Descripción: Obtiene información detallada de una orden específica por su ID.
  + Datos de Entrada: id de la orden en la URL.
  + Respuesta: JSON con los detalles de la orden.
* **Actualizar Estado de Orden**:
  + Endpoint: PUT /api/ordenes/{id}
  + Descripción: Modifica el estado de una orden.
  + Datos de Entrada: id de la orden en la URL, JSON con el campo estado.
  + Respuesta: Detalles de la orden actualizada.
* **Eliminar Orden**:
  + Endpoint: DELETE /api/ordenes/{id}
  + Descripción: Elimina una orden específica por su ID.
  + Datos de Entrada: id de la orden en la URL.
  + Respuesta: Confirmación de la eliminación.

**Gestión de Categorías**

* **Crear Categoría**:
  + Endpoint: POST /api/categorias
  + Descripción: Añade una nueva categoría de productos.
  + Datos de Entrada: JSON con los campos nombre, descripcion.
  + Respuesta: Detalles de la categoría creada, incluyendo su ID.
* **Consultar Categorías**:
  + Endpoint: GET /api/categorias
  + Descripción: Obtiene una lista de todas las categorías disponibles.
  + Datos de Entrada: Ninguno.
  + Respuesta: JSON con la lista de categorías.
* **Actualizar Categoría**:
  + Endpoint: PUT /api/categorias/{id}
  + Descripción: Modifica el nombre o la descripción de una categoría existente.
  + Datos de Entrada: id de la categoría en la URL, JSON con los campos a actualizar (nombre, descripcion).
  + Respuesta: Detalles de la categoría actualizada.
* **Eliminar Categoría**:
  + Endpoint: DELETE /api/categorias/{id}
  + Descripción: Elimina una categoría específica por su ID.
  + Datos de Entrada: id de la categoría en la URL.
  + Respuesta: Confirmación de la eliminación.

**3. Requerimientos No Funcionales**

* **Seguridad**: Implementar autenticación y autorización mediante JWT para asegurar que solo usuarios autenticados puedan acceder y manipular los recursos.
* **Escalabilidad**: Diseñar el API de manera que pueda manejar un aumento en el volumen de transacciones y datos sin degradar el rendimiento.
* **Mantenibilidad**: Asegurar que el código sea claro, modular y documentado, permitiendo futuras actualizaciones y mejoras.
* **Interoperabilidad**: Utilizar estándares como HTTP, JSON y Swagger para facilitar la integración con otros sistemas y aplicaciones.
* **Rendimiento**: Optimizar consultas y operaciones para garantizar tiempos de respuesta rápidos, incluso bajo carga alta.

**4. Tecnología**

* **Backend**: Node.js con Express.js para manejar las solicitudes HTTP y gestionar la lógica del servidor.
* **Base de Datos**: MongoDB para el almacenamiento de datos, ofreciendo flexibilidad y escalabilidad.
* **Autenticación**: JWT (JSON Web Tokens) para la autenticación y autorización de usuarios.
* **Documentación**: Documentación detallada en un formato accesible, facilitando su uso e integración por parte de otros desarrolladores y sistemas.

## ii. Análisis de Software

**Diseño de la Base de Datos**

**Diagrama Entidad-Relación (E-R)**



**Diagrama Lógico**:

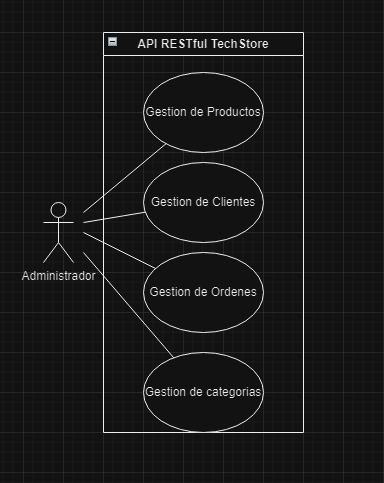


**iii. Diseño del Software**

**Diagrama de Casos de Uso**

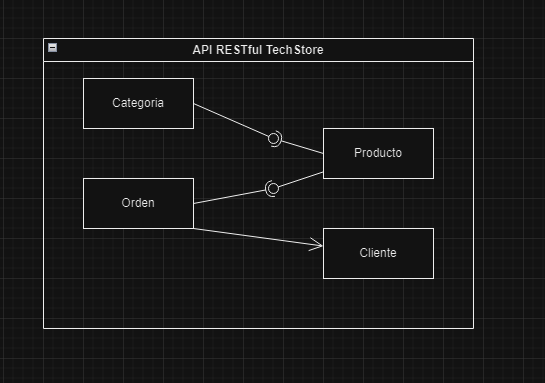
El Diagrama de Casos de Uso describe las interacciones entre el actor (Administrador) y el sistema (API RESTful de TechStore). Los casos de uso principales son:

* **Gestión de Productos**:
  + Crear Producto
  + Consultar Producto
  + Actualizar Producto
  + Eliminar Producto
* **Gestión de Clientes**:
  + Registrar Cliente
  + Consultar Cliente
  + Actualizar Cliente
  + Eliminar Cliente
* **Gestión de Órdenes**:
  + Crear Orden
  + Consultar Orden
  + Actualizar Orden
  + Eliminar Orden
* **Gestión de Categorías**:
  + Crear Categoría
  + Consultar Categorías
  + Actualizar Categoría
  + Eliminar Categoría

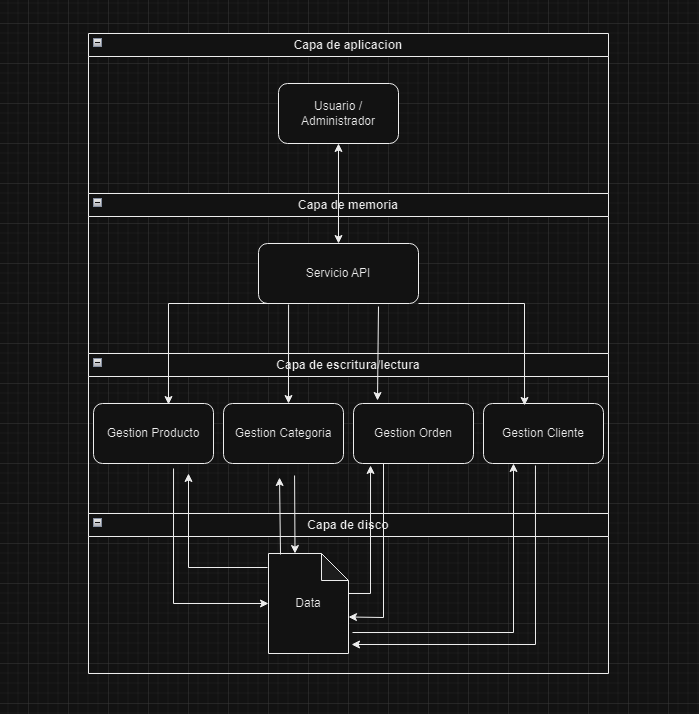


**Diagrama de Componentes**

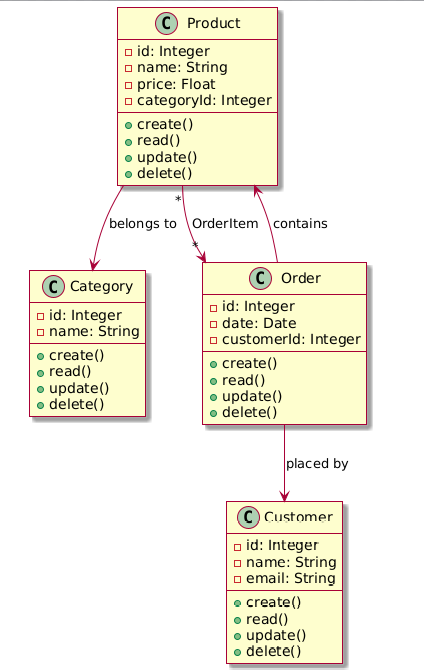
El Diagrama de Componentes muestra la organización de los principales módulos de tu proyecto, incluyendo controladores, modelos, rutas y configuración.



**Diagrama de Arquitectura**



**Diagrama de Clases**



# Recomendaciones

1. **Seguridad y Autenticación:**
   * **Implementar medidas adicionales de seguridad:** Aunque JWT ofrece una capa de autenticación segura, es recomendable implementar la autenticación multifactor (MFA) para añadir una capa extra de protección.
   * **Realizar auditorías de seguridad periódicas:** Llevar a cabo revisiones y pruebas de penetración regulares para identificar y corregir posibles vulnerabilidades en el API.
2. **Documentación y Mantenibilidad:**
   * **Mantener una documentación actualizada y accesible:** Utilizar herramientas como Swagger para generar documentación interactiva y actualizada del API, facilitando su uso y comprensión por parte de otros desarrolladores.
   * **Adoptar buenas prácticas de codificación:** Seguir principios de codificación limpia y modular para asegurar que el código sea fácil de mantener y escalar.
3. **Optimización y Rendimiento:**
   * **Monitorear y optimizar consultas a la base de datos:** Implementar índices adecuados y optimizar las consultas para garantizar un rendimiento óptimo, especialmente en operaciones de lectura intensiva.
   * **Utilizar caché donde sea aplicable:** Considerar la implementación de caché para mejorar la velocidad de respuesta y reducir la carga en la base de datos.
4. **Escalabilidad y Flexibilidad:**
   * **Diseñar para la escalabilidad horizontal:** Asegurarse de que el sistema pueda escalar horizontalmente, permitiendo añadir más servidores según la demanda.
   * **Adoptar una arquitectura basada en microservicios:** Considerar dividir el API en microservicios independientes para mejorar la escalabilidad y facilitar el mantenimiento.
5. **Pruebas y Calidad del Software:**
   * **Implementar pruebas automáticas:** Incluir pruebas unitarias, de integración y end-to-end en el ciclo de desarrollo para asegurar la calidad y fiabilidad del software.
   * **Utilizar herramientas de CI/CD:** Adoptar herramientas de Integración Continua y Despliegue Continuo para automatizar el proceso de pruebas y despliegue, asegurando que los cambios se implementen de manera rápida y segura.

# Conclusiones

La implementación del API RESTful para TechStore representa un avance significativo en la modernización de su infraestructura tecnológica. A través de este proyecto, se han logrado varios objetivos clave:

1. **Mejora en la Gestión de Recursos:** Se han desarrollado funcionalidades robustas para la gestión de productos, clientes, órdenes y categorías, permitiendo una administración más eficiente y precisa de los recursos de TechStore.
2. **Seguridad y Control de Acceso:** La incorporación de JWT para la autenticación y autorización garantiza que solo usuarios autorizados puedan acceder y manipular los recursos, mejorando la seguridad general del sistema.
3. **Escalabilidad y Flexibilidad:** El diseño del API permite un fácil escalado horizontal y una integración fluida con otras aplicaciones y sistemas, asegurando que TechStore pueda crecer y adaptarse a las necesidades cambiantes del mercado.
4. **Eficiencia Operativa:** La automatización de procesos a través del API ha reducido significativamente los errores manuales y los tiempos de procesamiento, mejorando la eficiencia operativa y la experiencia del cliente.
5. **Fundamentos Sólidos para el Futuro:** El proyecto establece una base tecnológica sólida que permitirá futuras mejoras y adaptaciones, asegurando que TechStore se mantenga competitiva en el dinámico mercado de productos electrónicos.

En resumen, el desarrollo del API RESTful ha cumplido con los objetivos planteados y ha proporcionado a TechStore una herramienta poderosa para optimizar sus operaciones diarias, mejorando tanto la eficiencia interna como la satisfacción del cliente.

# Bibliografía

1. **Fielding, R. T. (2000).** "Architectural Styles and the Design of Network-based Software Architectures." Doctoral dissertation, University of California, Irvine.
2. **O'Reilly Media (2013).** "RESTful Web APIs." Richardson, L., & Amundsen, M.
3. **Apress (2017).** "Pro REST API Development with Node.js." Seguin, J.
4. **Addison-Wesley (2009).** "RESTful Web Services Cookbook." Allamaraju, S.
5. **Manning Publications (2018).** "Node.js in Action." Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N.