**Tipo de artículo: Artículo original**

**Simple CRUD Nodejs + Express + PostgreSQL**

**Simple CRUD Nodejs + Express + PostgreSQL**

**Nombre y apellidos: Joaquin Rivas Sanchez**

1 afiliación institucional completa (Departamento XX, Facultad 4, Universidad UCI). joaquiners@uci.estudiantes.cu

**\* Autor para correspondencia:** joaquiners@uci.estudiantes.cu

**Resumen**

En este artículo, se presenta la implementación de una API CRUD (Crear, Leer, Actualizar, Borrar) utilizando Node.js, Express y PostgreSQL. El objetivo principal del trabajo es proporcionar a los lectores una guía paso a paso sobre cómo construir una API básica con estas tecnologías. Se describe la metodología utilizada para la creación de la API, se presentan los resultados obtenidos y se discuten las implicaciones. La API resultante es un punto de partida para el desarrollo de aplicaciones web más avanzadas.

**Palabras clave:** Node.js, Express, PostgreSQL, API, CRUD

***Abstract***

*In this article, we present the implementation of a CRUD API (Create, Read, Update, Delete) using Node.js, Express, and PostgreSQL. The main goal of the work is to provide readers with a step-by-step guide on how to build a basic API with these technologies. The methodology used for creating the API is described, the results are presented, and their implications are discussed. The resulting API serves as a starting point for the development of more advanced web applications.*

***Keywords:*** *Keywords: Node.js, Express, PostgreSQL, API, CRUD*

**Recibido: dd/mm/2021**

**Aceptado: dd/mm/2021**

**En línea: 01/mm/2021**

**Introducción**

En el mundo del desarrollo web, la creación de APIs es una tarea esencial para interactuar con bases de datos y permitir que las aplicaciones se comuniquen con los servidores. Node.js, junto con el marco de desarrollo Express, se ha convertido en una elección popular para construir estas APIs de manera eficiente y efectiva. En este artículo, exploraremos cómo implementar una API CRUD simple utilizando Node.js y Express, y cómo conectarla a una base de datos PostgreSQL.

El desarrollo de APIs CRUD es una parte fundamental en la creación de aplicaciones web, ya que permite realizar operaciones básicas sobre los datos, como la creación, lectura, actualización y eliminación de registros. PostgreSQL, una robusta base de datos relacional, se utilizará como el sistema de gestión de bases de datos en este proyecto.

El objetivo principal de este trabajo es proporcionar a los desarrolladores una guía detallada sobre cómo construir una API CRUD utilizando estas tecnologías. Comenzaremos por configurar el entorno y la base de datos, luego crearemos las rutas y controladores necesarios para las operaciones CRUD. Además, se abordarán temas como la validación de datos de entrada y el manejo de errores.

A medida que avancemos en el desarrollo de la API, se describirán los pasos y se presentarán ejemplos prácticos para ilustrar cada parte del proceso. Al final, tendremos una API totalmente funcional que servirá como punto de partida para la creación de aplicaciones web más complejas.

En resumen, este artículo se centra en la implementación de una API CRUD en Node.js con Express, conectada a PostgreSQL, con el propósito de brindar a los desarrolladores una comprensión clara de cómo construir aplicaciones web robustas y escalables. La metodología utilizada se basa en las mejores prácticas de desarrollo web y se presenta de manera accesible para aquellos que deseen adentrarse en este campo.

**Materiales y métodos**

En esta sección, describiremos en detalle los materiales y métodos utilizados para la implementación de la API CRUD en Node.js con Express y la conexión a la base de datos PostgreSQL. Esto incluye la configuración del entorno de desarrollo, la creación de la base de datos y las herramientas empleadas.

**Entorno de Desarrollo**:

- **Node.js y npm:** Para comenzar, se instaló Node.js, que proporciona el entorno de ejecución para JavaScript en el servidor, y npm (Node Package Manager) para gestionar las dependencias del proyecto.

- **Editor de código:** Se utilizó Visual Studio Code, para escribir y editar el código fuente.

- **Configuración de la Base de Datos**:

- **PostgreSQL + Docker**: Se configuró un servidor PostgreSQL para almacenar los datos de la aplicación. PostgreSQL se eligió por su robustez y capacidad para manejar aplicaciones de alta concurrencia.

- **Creación de la Base de Datos y Tablas:** Se creó una base de datos llamada "articulosdb" y se definieron las tablas necesarias para la API, incluyendo la estructura de campos requeridos.

**Desarrollo de la API**

- **Express.js:** Se utilizó el marco de desarrollo Express.js para la creación de la API. Express simplifica la gestión de rutas y controladores, lo que facilita el desarrollo de la aplicación.

- **Rutas y Controladores:** Se definieron rutas para las operaciones CRUD, como la creación, lectura, actualización y eliminación de registros. Cada ruta se asoció con un controlador específico que maneja las solicitudes HTTP y las interacciones con la base de datos.

- **Validación de Datos:** Se utilizo la librería **Zod** para garantizar la integridad de los datos, se implementó una validación de datos de entrada en las rutas, asegurando que los datos cumplen con los requisitos establecidos.

- **Manejo de Errores**: Se implementaron mecanismos de manejo de errores para capturar y gestionar posibles errores en las solicitudes, proporcionando respuestas adecuadas al cliente.

**Pruebas**:

- **RapidAPI**: Para probar la API durante el desarrollo, se utilizó la herramienta **RapidAPI**. Se realizaron pruebas en todas las rutas y se verificó que las operaciones CRUD funcionaran correctamente.

En esta sección, hemos descrito los materiales y métodos utilizados para la implementación de la API CRUD en Node.js con Express y la conexión a PostgreSQL. La configuración del entorno, la creación de la base de datos y la metodología de desarrollo son componentes fundamentales para comprender cómo se lograron los resultados presentados en este artículo.

**Resultados y discusión**

Resultados y Discusión

Resultados de la Implementación:

1. Creación de Registros:

   - Se logró implementar con éxito la creación de registros en la base de datos utilizando la ruta correspondiente.

   - Se verificó que los nuevos registros se almacenaran correctamente en la tabla de la base de datos.

2. Lectura de Registros:

   - La operación de lectura de registros se implementó de manera efectiva, permitiendo la recuperación de datos almacenados.

   - Se comprobó que la API proporcionara datos precisos y actualizados.

3. Actualización de Registros:

   - La actualización de registros se realizó satisfactoriamente a través de la API.

   - Se verificó que los registros se actualizaran con precisión en la base de datos.

4. Eliminación de Registros:

   - La operación de eliminación de registros se implementó con éxito.

   - Se confirmó que los registros se eliminaran de la base de datos sin problemas.

Discusión de los Resultados:

5. Eficiencia y Rendimiento:

   - Se observó que la API CRUD desarrollada demostró un rendimiento eficiente en términos de velocidad de respuesta y uso de recursos.

   - La implementación fue capaz de manejar un alto número de solicitudes de manera óptima.

6. Validación y Seguridad:

   - La validación de datos implementada en la API contribuyó a garantizar la integridad de los datos almacenados.

   - Se aplicaron medidas de seguridad para proteger la API de posibles amenazas, como la inyección de SQL.

7. Escalabilidad y Mantenimiento:

   - La estructura modular de la API permite una fácil escalabilidad y mantenimiento a medida que los requisitos cambian o se agregan nuevas funcionalidades.

   - Se utilizaron prácticas de desarrollo sostenible para garantizar la longevidad y la capacidad de mantenimiento de la API.

8. Posibilidades de Futuro Desarrollo:

   - La API CRUD proporciona una base sólida para futuras expansiones y proyectos relacionados.

   - Se discuten posibles direcciones para el desarrollo futuro, como la adición de autenticación de usuarios y la implementación de características avanzadas.

**Conclusiones**

En resumen, la implementación exitosa de una API CRUD con Node.js, Express y PostgreSQL ha demostrado el cumplimiento de los objetivos del proyecto, así como su contribución significativa al campo de desarrollo web. Este trabajo destaca la utilidad de estas tecnologías modernas para crear aplicaciones web eficientes y seguras. Además, sienta las bases para futuros proyectos al sugerir áreas de investigación y desarrollo, como la autenticación de usuarios y la optimización de rendimiento. En conjunto, esta investigación proporciona una guía valiosa y accesible para desarrolladores y promueve prácticas seguras y sostenibles en el desarrollo de aplicaciones web