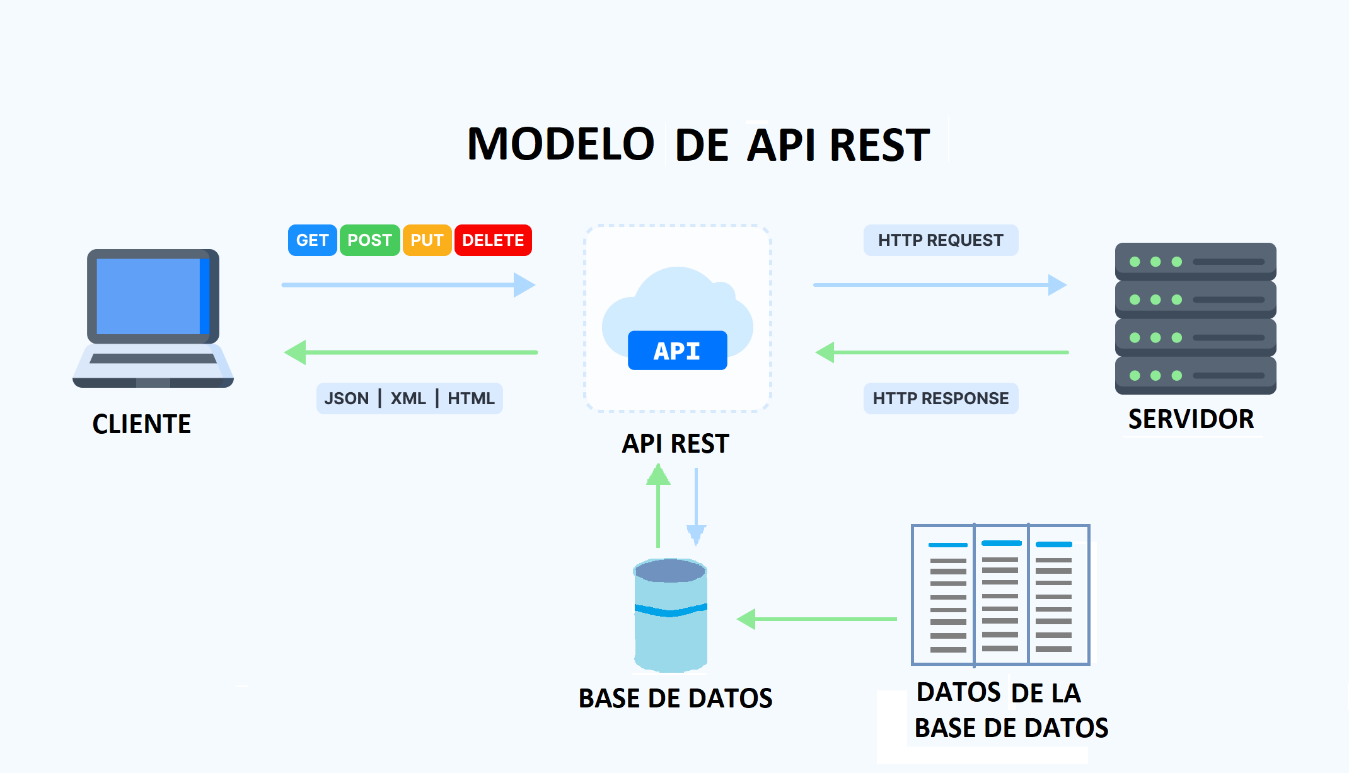
**TUTORIAL API REST DORY**

En este tutorial se explicará cómo construir el API REST de la aplicación web Dory.



A continuación se definirá api, servicio rest y api rest, para entender el modelo de api rest que se realizará.

**API** (Interfaz de Programación de Aplicaciones), es un conjunto de reglas, protocolos y herramientas que se utilizan para construir y acceder a aplicaciones y servicios web. un API permite que diferentes aplicaciones y sistemas interactúen y compartan información entre sí. Un API se puede utilizar para acceder a los datos y funcionalidades de una aplicación o servicio de terceros, sin necesidad de conocer todos los detalles internos de cómo se implementa esa funcionalidad.

**REST** (transferencia de estado representativo), se refiere a un estilo de arquitectura para el diseño de servicios web. En un servicio REST, los recursos (como datos, documentos, imágenes, etc.) son expuestos a través de una interfaz web utilizando URIs (Identificadores de recursos uniformes) como identificadores únicos. Cada recurso puede ser accedido o modificado a través de las operaciones estándar del protocolo HTTP, como GET, POST, PUT, DELETE, entre otros. Los servicios REST permiten una alta escalabilidad, ya que se basan en una arquitectura "stateless" (sin estado), lo que significa que cada solicitud se procesa de forma independiente, sin mantener información de estado entre solicitudes. Además, el uso de estándares abiertos y bien conocidos, como HTTP y JSON, hace que los servicios REST sean fáciles de implementar y consumir.

Un **API REST** es una interfaz de programación de aplicaciones que se basa en el protocolo HTTP para permitir la comunicación entre diferentes sistemas y aplicaciones. En un API REST, las diferentes operaciones o acciones se representan mediante verbos HTTP, como GET, POST, PUT, DELETE, etc. Cada una de estas acciones tiene una respuesta asociada en formato JSON, XML, HTML, entre otros. Las API REST se utilizan comúnmente en la creación de servicios web y aplicaciones móviles, permitiendo la comunicación entre clientes y servidores a través de internet utilizando el protocolo HTTP.

**DESCRIPCIÓN DE LAS TECNOLOGÍAS USADAS PARA IMPLEMENTAR EL API REST DORY**

Las tecnologías que se utilizaron para crear el API REST del sitio web Dory son:

**Visual Studio Code (VS Code):** Es un editor de código fuente gratuito muy popular entre los desarrolladores que se utiliza para escribir y depurar el código de la aplicación. Está disponible en diferentes plataformas como Windows, macOS y Linux.

Url de descarga: https://code.visualstudio.com/download

**MySQL:** Es un sistema de gestión de bases de datos relacionales(es decir, los datos se organizan en tablas, donde cada tabla representa una entidad o relación, y cada fila de la tabla representa una instancia o registro de esa entidad). Permite almacenar y gestionar los datos de la aplicación.

MySQL es libre y gratuita para su uso, modificación y distribución, sujeta a los términos de la licencia pública general de GNU (GPL). Sin embargo, si se desea utilizar MySQL como parte de una solución empresarial que requiere soporte técnico y actualizaciones de seguridad, es posible adquirir una licencia comercial a través de Oracle Corporation, la cual ofrece diferentes opciones de licencia y soporte, incluyendo licencias por servidor, por núcleo o por usuario.

El costo de las licencias comerciales de MySQL varía en función del tipo de licencia y los servicios de soporte y mantenimiento que se requieran. Es recomendable consultar con un representante de ventas de Oracle para obtener información precisa sobre los precios y las opciones de licencia disponibles.

**Node.js:** Es un conjunto de herramientas, recursos y servicios que permiten a los desarrolladores o usuarios desarrollar y ejecutar aplicaciones y software en el lenguaje de programación Javascript en el lado del servidor. Permite crear aplicaciones web y api rest.

Las limitaciones de node.js son:

* Rendimiento limitado en aplicaciones con mucho cálculo, ya que solo puede utilizar un núcleo de CPU a la vez.
* No es adecuado para aplicaciones de escritorio o móviles: Node.js se centra en el desarrollo de aplicaciones web y de servidor.
* No es una solución de base de datos completa en sí mismo.
* Node.js tiene un sistema de manejo de errores un tanto complejo que puede resultar difícil de manejar.
* Necesidad de utilizar paquetes de terceros para muchas funcionalidades comunes, lo que puede resultar en problemas de compatibilidad o seguridad.

Node.js es mantenido por una comunidad global de desarrolladores y liderado por un equipo central de desarrollo conocido como **comité técnico central de Node.js**, mientras que la **fundación Node.js** se encarga de apoyar y promover el desarrollo del proyecto.

Node.js no tiene licencia GNU, está disponible bajo la Licencia MIT, que es una licencia de software libre y de código abierto que permite a los usuarios modificar, distribuir y utilizar el software sin restricciones. La Licencia MIT es una de las licencias de software libre más permisivas y es utilizada por muchos proyectos de código abierto, incluyendo jQuery, AngularJS y Rails.

Url de descarga: https://nodejs.org/en

**Express:** Es una libreria de Node.js que facilita la creación de servidores web y APIs REST. Permite manejar rutas, peticiones y respuestas HTTP de manera sencilla.

**Postman:** Es una herramienta muy popular en el mundo del desarrollo web y de APIs REST. Se utiliza principalmente para probar, depurar y documentar APIs REST de una manera fácil y eficiente. Con Postman, puedes enviar peticiones HTTP a una API REST y recibir las respuestas correspondientes. La herramienta te permite configurar diferentes tipos de peticiones HTTP, como GET, POST, PUT, DELETE, entre otros, y enviar parámetros y datos en diferentes formatos, como JSON, XML y formularios.

**Git:** Es un sistema de control de versiones ampliamente utilizado para el desarrollo de software. Se utiliza para rastrear los cambios en el código fuente y coordinar el trabajo entre varios desarrolladores en un proyecto de software.

**Piezas de software utilizadas:**

**NPM (Node Package Manager):** Es el gestor de paquetes oficial de Node.js. Permite instalar y gestionar de manera sencilla y eficiente los paquetes y dependencias que se utilizan en una aplicación de Node.js.

Para instalar un módulo local en un proyecto de Node.js, se debe utilizar el comando **npm install <nombre\_del\_modulo>**. Para instalar un módulo globalmente, se debe utilizar el comando **npm install -g <nombre\_del\_modulo>**. Los módulos locales se guardan en la carpeta node\_modules del proyecto, mientras que los módulos globales se instalan en un directorio global de Node.js.

La diferencia principal entre ambos es que los módulos globales se pueden utilizar desde cualquier proyecto, mientras que los módulos locales sólo están disponibles para el proyecto en el que se instalaron.

Los comandos locales están instalados en el proyecto y se encuentran en el archivo package.json dentro de la sección "scripts". Para ejecutar uno de estos comandos, se puede utilizar el comando **npm run <nombre\_del\_comando>.** Por ejemplo, si se tiene un comando llamado "start" en el archivo package.json, se puede ejecutar el comando npm run start en la terminal.

Si se desea ejecutar un comando local que no esté dentro de un proyecto de Node.js, se debe estar en el directorio que contiene el archivo que define el comando. Por ejemplo, si se tiene un archivo index.js que define un comando, se puede ejecutar el comando node index.js en la terminal desde el directorio que contiene el archivo.

Los comandos globales pueden ser utilizados desde cualquier directorio. Para ejecutar un comando global, se puede simplemente escribir el nombre del comando en la terminal. Por ejemplo, si se tiene instalado globalmente el comando nodemon, se puede ejecutar simplemente escribiendo nodemon en la terminal.

**mysql2:** Es un paquete de Node.js que se utiliza para conectarse a una base de datos MySQL. Permite enviar consultas y recibir resultados de manera eficiente.

**Nodemon:** Es una herramienta que se utiliza para reiniciar automáticamente una aplicación Node.js cuando se detectan cambios en el código fuente. Esto evita tener que reiniciar manualmente la aplicación cada vez que se realizan cambios.

**Body-parser:** Es una librería de Express que se utiliza para parsear el cuerpo de una petición HTTP en un objeto JavaScript que se puede manipular fácilmente. Esto es útil para procesar datos que se envían en formato JSON o en formularios HTML. **Parsear** es el proceso de analizar y descomponer una cadena de texto o un archivo en sus componentes estructurales (objetos, arrays, strings, etc.) para poder procesarlos y manipularlos de manera programática.

**Cors:** Es una librería de Express que se utiliza para permitir el acceso a recursos desde otros dominios y construir aplicaciones web que consumen servicios de terceros de manera segura y controlada.

CORS agrega los encabezados HTTP necesarios a las respuestas del servidor para permitir el acceso a los recursos desde un dominio diferente al que sirve la página web. Esto se hace agregando un encabezado Access-Control-Allow-Origin a la respuesta del servidor, que indica qué dominios están autorizados para acceder a los recursos.

Por defecto, Express no permite solicitudes de recursos desde otros dominios, CORS es necesario para habilitar este tipo de acceso a los recursos.

CORS se puede configurar de diferentes maneras, por ejemplo, para permitir solicitudes de cualquier origen (\*), o sólo para dominios específicos. También se pueden especificar otros encabezados HTTP personalizados que se deben agregar a las respuestas del servidor.

**JWT Simple**: Es una librería para Node.js que proporciona una manera simple de generar, firmar y verificar tokens JWT (JSON Web Token) en una aplicación Node.js. Los tokens JWT se utilizan comúnmente para la autenticación y la autorización en aplicaciones web y móviles.

La representación del token es un string codificado en base64 que consta de tres partes separadas por puntos (.). Estas partes son: encabezado, carga y firma.

Header (Encabezado): Es un objeto JSON que describe el tipo de token y el algoritmo de cifrado utilizado para firmar el token. Por ejemplo:

{

"alg": "HS256",

"typ": "JWT"

}

El encabezado se codifica en base64 y se incluye en el token como la primera parte, seguido de un punto.

Payload (Carga): Es un objeto JSON que contiene la información del usuario o la aplicación que se está autenticando. Por ejemplo:

{

"sub": "1234567890",

"name": "John Doe",

"iat": 1516239022

}

El payload también se codifica en base64 y se incluye en el token como la segunda parte, seguido de otro punto.

Signature (Firma): Es una cadena de caracteres que se utiliza para verificar la autenticidad del token. La firma se crea utilizando el encabezado y el payload codificados en base64, junto con una clave secreta que sólo el servidor conoce. Por ejemplo:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

La firma se agrega al final del token, separado por un tercer punto.

**Moment**: Es una librería de manipulación de fechas y horarios para JavaScript, que permite la manipulación, formateo y análisis de fechas y horarios en diferentes formatos.

**Nodemailer**: Es una librería de Node.js que permite enviar correos electrónicos utilizando diferentes servicios de correo electrónico, como SMTP, SendGrid o Mailgun.

**Http-Error:** Es una biblioteca para Node.js que proporciona una manera fácil de manejar errores HTTP en aplicaciones Node.js y Express. Permite crear objetos de error personalizados con códigos de estado HTTP y mensajes personalizados para que puedan ser manejados adecuadamente por la aplicación.

**Dotenv**: Es un paquete de Node.js que se utiliza para cargar variables de entorno desde un archivo .env en una aplicación. El archivo .env es un archivo de texto plano que contiene una lista de variables de entorno y sus valores. Por ejemplo, se puede utilizar un archivo .env para almacenar información confidencial como claves de API, contraseñas de bases de datos, tokens de autenticación, etc.

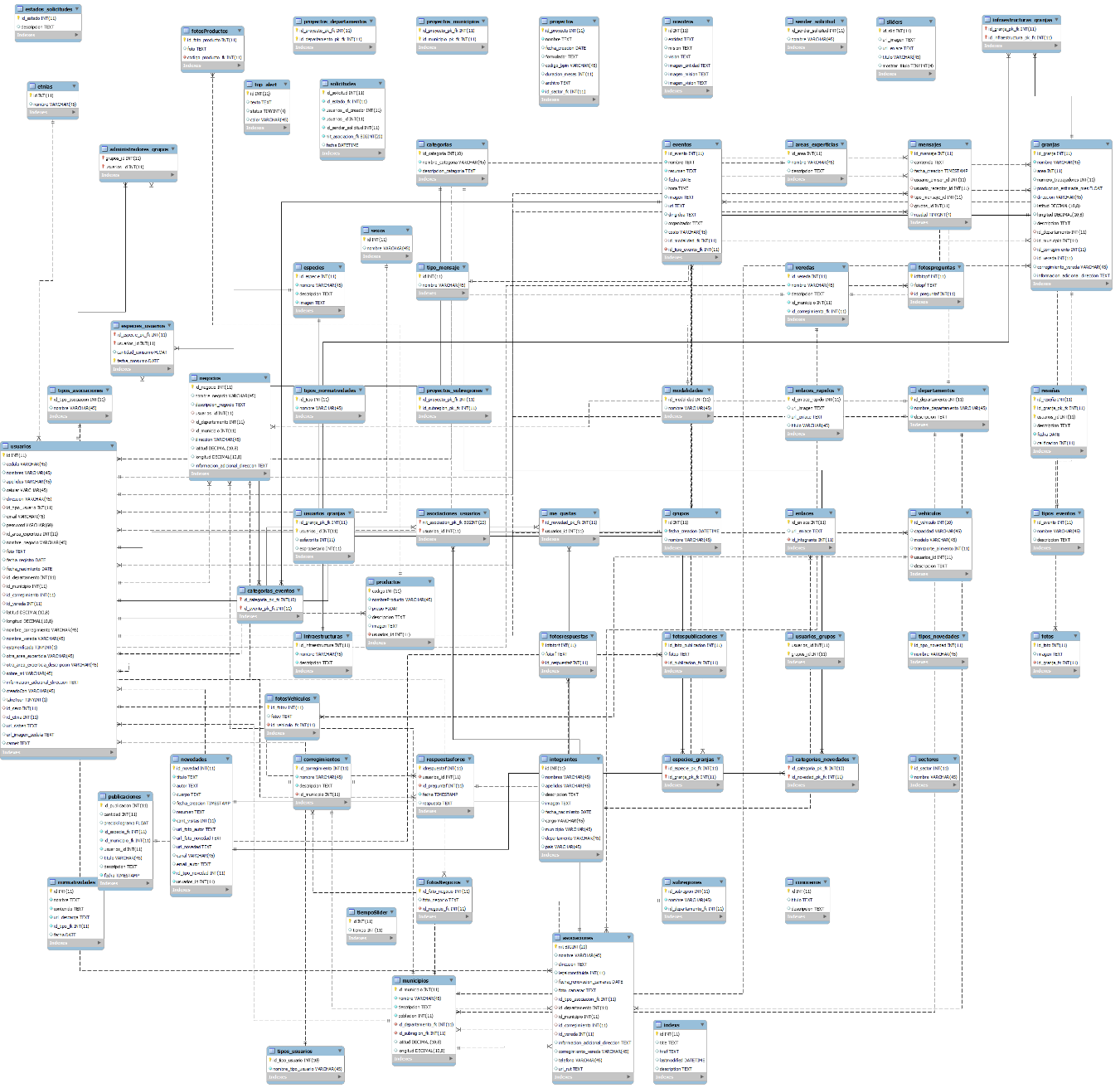
Para utilizar dotenv en una aplicación Node.js, primero se debe instalar utilizando el comando "npm install dotenv". Luego, se debe crear un archivo .env y definir las variables de entorno y sus valores. Por último, se debe cargar las variables de entorno en la aplicación utilizando la función "config()" de dotenv.

**MODELO DE DATOS DE LA BASE DE DATOS**

Planteada la metodología, se empieza a realizar el modelo entidad-relación de los datos con el cual se implementa la base de datos.

En el modelo de datos se estructura cada una de las tablas de la base de datos. En cada tabla se crean los campos, los tipos de datos, el tamaño de los datos, el valor recibido por defecto, se selecciona el campo que tendrá la llave primaria y se colocan las llaves foráneas si son necesarias en dicha tabla, además de usar la función de **auto\_increment** si es imprescindible en el campo de dicha entidad.

Creado el modelado de los datos se empieza a realizar el script de la base de datos y el script para poblar la base de datos. Con el script de la base de datos creamos esa base de datos y con el script de población repoblamos la base de datos, para proseguir con la realización de múltiples consultas con el fin extraer información permitiendo la manipulación de los datos como son la agregación, eliminación y la modificación.



**INTRODUCCIÓN**

¡Bienvenido al tutorial de API REST construida con Node.js, MySQL y Express!

En este tutorial, aprenderás a construir el API REST DORY utilizando Node.js y Express. Esta API REST se conectará a una base de datos MySQL utilizando mysql2 para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos.

La API REST que construirás permitirá a los usuarios enviar solicitudes HTTP a la API para crear nuevos usuarios, recuperar información sobre usuarios existentes, actualizar la información y eliminar usuarios.

En el transcurso del tutorial, aprenderás a:

* Configurar el entorno de desarrollo para construir una API REST con Node.js y Express.
* Configurar y establecer una conexión a una base de datos MySQL utilizando mysql2.
* Definir rutas y controladores para manejar solicitudes HTTP y realizar operaciones CRUD en la base de datos MySQL.
* Validar y procesar datos de entrada del usuario antes de enviarlos a la base de datos.
* Manejar errores y devolver respuestas adecuadas al usuario.

**PROCEDIMIENTOS**

A continuación, se describirán los pasos para crear la API REST DORY.

**Descripción General:**

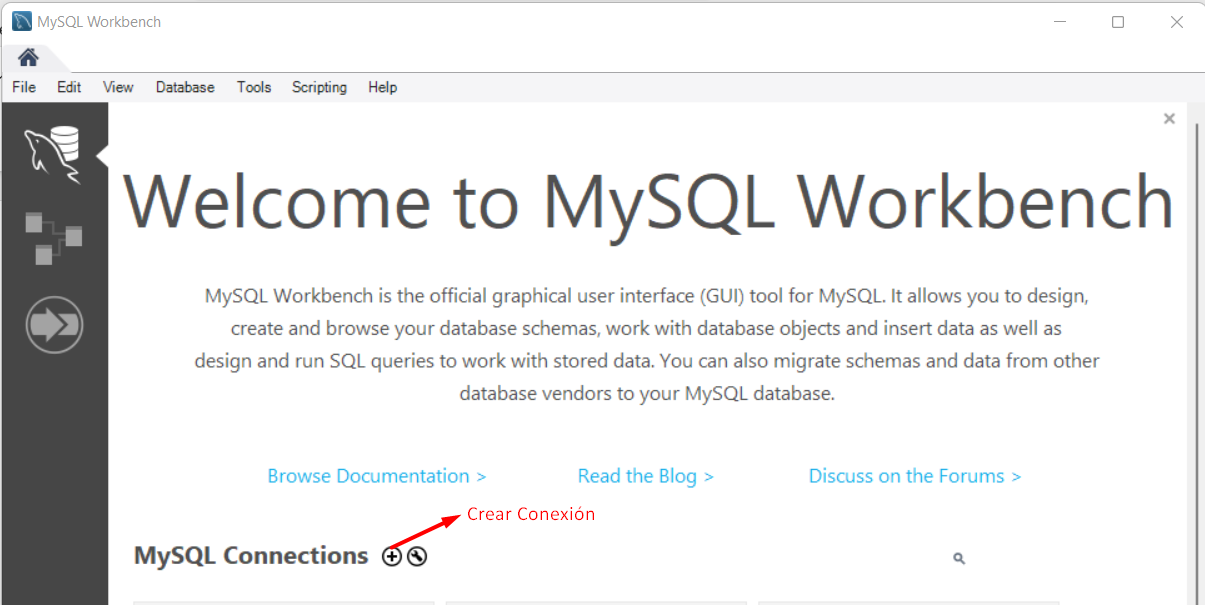
1. Configurar una base de datos MySQL y definir las tablas y relaciones necesarias.
2. Crear un servidor Node.js utilizando Express como framework.
3. Conectar la base de datos MySQL al servidor Node.js utilizando una biblioteca de conexión MySQL2.
4. Definir las rutas y métodos HTTP (GET, POST, PUT, DELETE) necesarios para la API REST en el servidor Node.js utilizando Express.
5. Escribir las funciones que manejan las solicitudes HTTP en el servidor Node.js, que interactúan con la base de datos MySQL y devuelven los resultados en formato JSON.
6. Probar la API REST utilizando una herramienta de prueba de API, como Postman.

**DESARROLLO DE API-REST DORY**

En el desarrollo del Api Rest Dory lo primero que se debe realizar es la configuración del entorno de trabajo; es decir, instalar en su computadora Node.js, MySQL, Postman, Visual Studio Code, MySQL Workbench. Ver anexos (instalaciones)

**Configuración de base de datos MySQL**

1. Abrir MySQL Workbench y crear una conexión



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Seleccionar la conexión creada y abrirla.

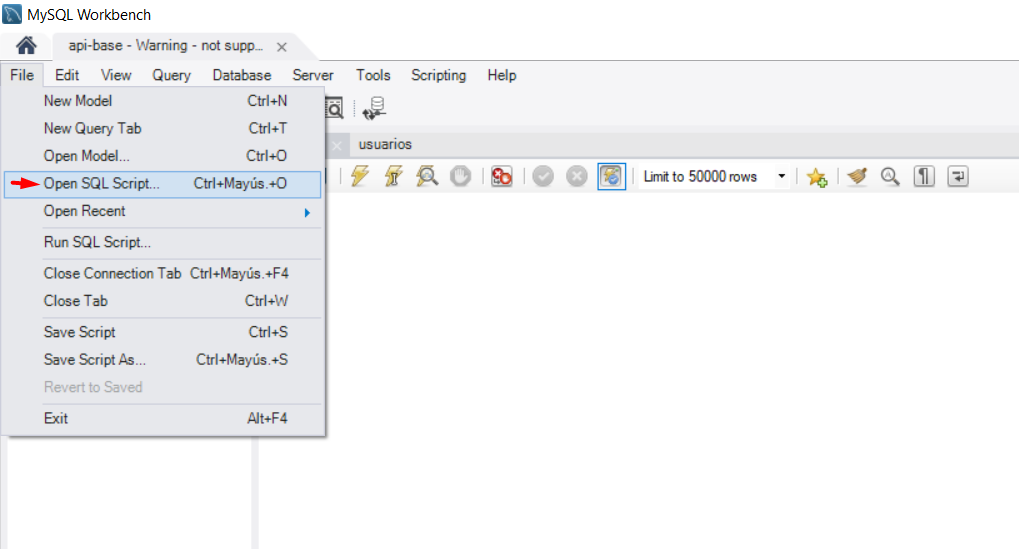
Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

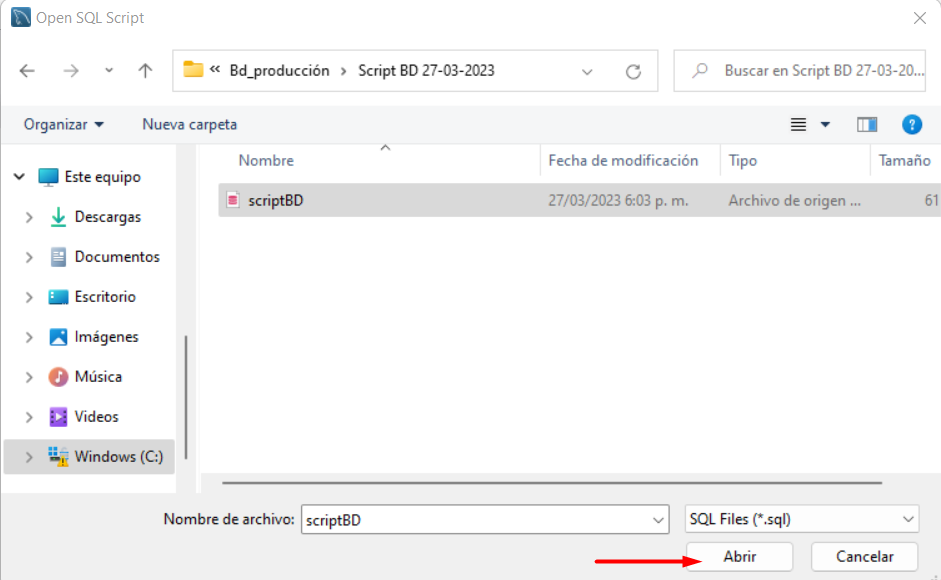
Descripción generada automáticamente

1. Abrir el Script para crear la base de datos.

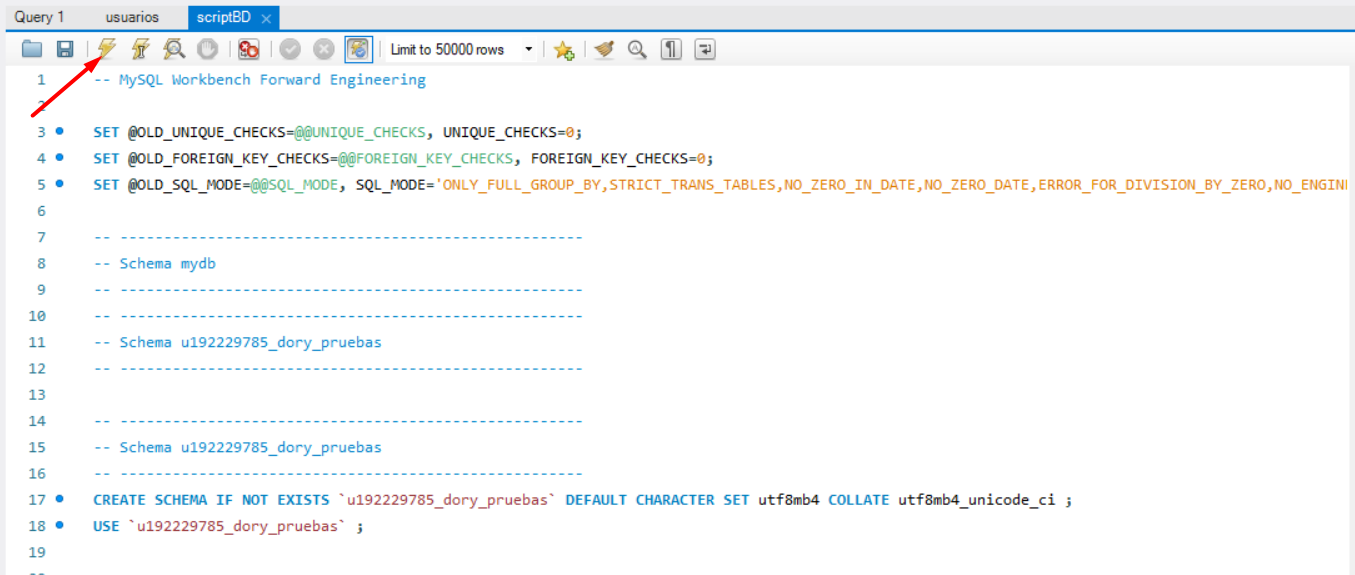
Pasos:

* Seleccionar “File” del menú principal
* Escoger el Ítem “Open SQL Script”
* Ubicar el archivo el script de la base de datos en su ordenador
* Abrir el script de la base de datos.

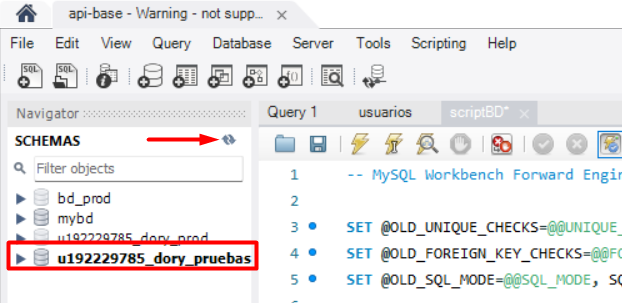




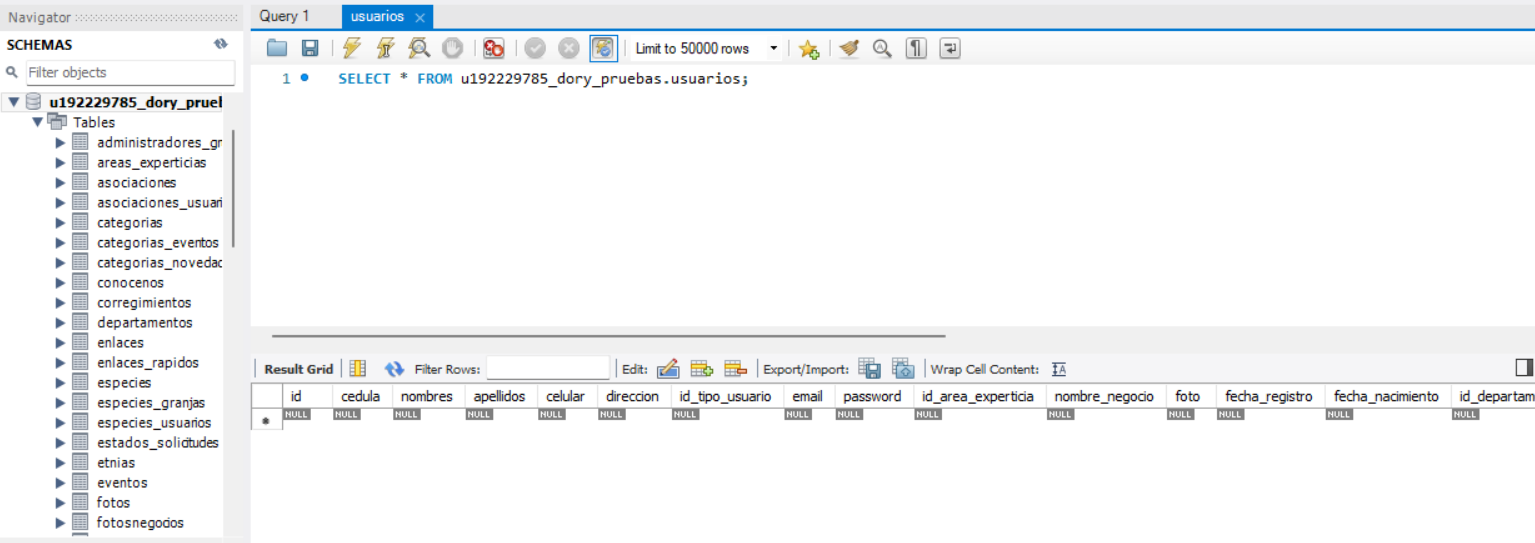
1. Ejecutar el script de la base de datos.



1. Seleccionar la opción actualizar SCHEMAS, para visualizar la base de datos creada.



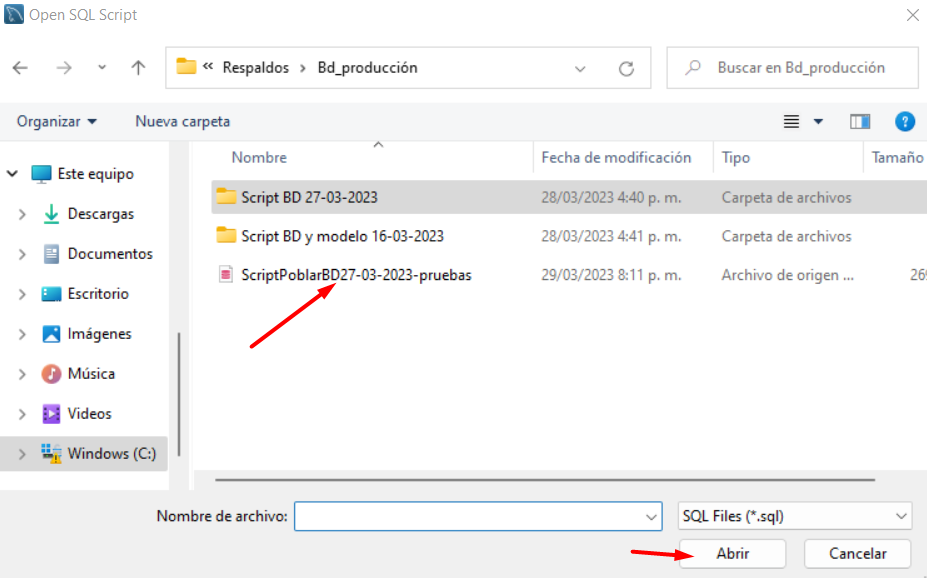
1. Abrir la base de datos, para visualizar las tablas creadas.



1. Si desea ejecutar pruebas a la Base de datos, inserte registros a las tablas creadas en la base de datos por cualquiera de los siguientes medios:

* Insertando manualmente datos a las tablas por Workbench. Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

  Descripción generada automáticamente
* Ejecutando sentencias de inserción MySQL. 
* Abrir el script de población de la base de datos de prueba y ejecutarlo del mismo modo como creo la base de datos.



**Creación del proyecto e instalaciones de las dependencias**

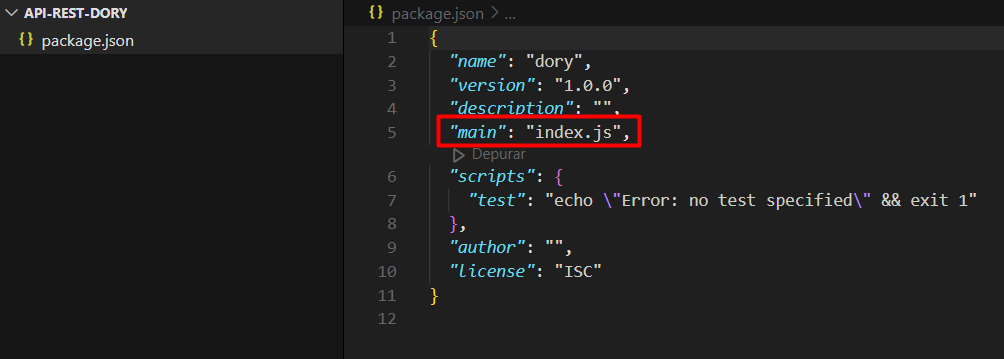
1. Crear una carpeta para tu proyecto, abrir una terminal y ubicarnos en dicho directorio.



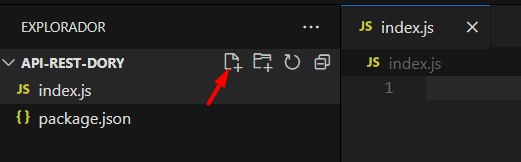
1. Inicializa el proyecto node.js, para eso ejecuta el comando: npm init



Este comando te permitirá crear el proyecto. Automáticamente se creará el archivo **package.json** y en él se especifica por defecto el archivo principal index.js.



1. Crear el archivo **index.js** en la raíz del proyecto.



1. Instalar dependencias necesarias como: express, mysql2, body-parser, entre otras. Al instalar express se crea automáticamente el archivo el package-lock.json.

Texto

Descripción generada automáticamente





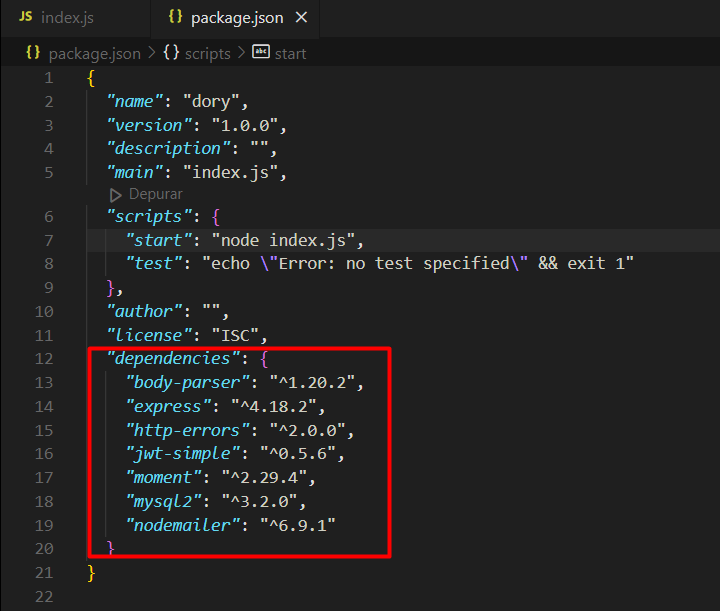
1. Instalar **nodemon** de manera global



En la package.json en la sección de Scripts se agrega el siguiente código:



**Nota:** En el archivo el package-lock.json se registran todas las dependencias que se instalan en el proyecto.



1. En la terminal ejecuta: **npm start** ó **nodemon** para que se reinicie la aplicación.

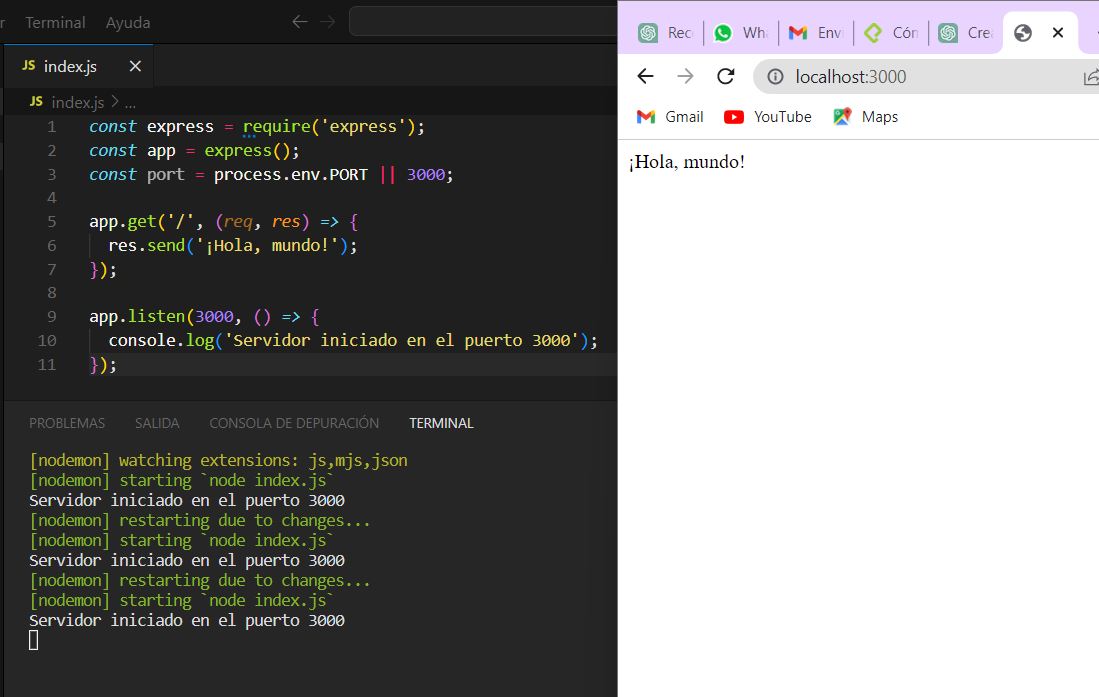




7. Para salir del servidor basta con presionar la tecla Ctrl+C.

**Creación de servidor Node.js utilizando Express**

En el archivo principal index.js creo el servidor y verificó su funcionamiento con el siguiente código:



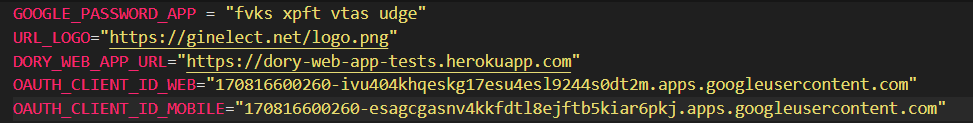
**Conexión de la base de datos MySQL al servidor Node.js utilizando la biblioteca MySQL2**

1. Realizar la declaración de variables de entorno, estas permiten la configuración flexible, portabilidad y seguridad de una aplicación. Al utilizar variables de entorno, se evita tener que codificar información confidencial en el código fuente y se permite que diferentes componentes de una aplicación se comuniquen entre sí sin tener que conocer los detalles de la configuración de la aplicación.

* Se debe instalar el paquete Dotenv con el siguiente comando:



* Crear un archivo “.env” en la raíz del proyecto, y agregar las variables de entorno y sus valores



1. Crear la conexión con la base de datos. Para esto se crea un archivo js en la raíz del proyecto llamada config donde tendremos un objeto de configuración que tendrá la base de datos y la clave secreta del token. Exportamos el objeto config.

