**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Desarrollo Backend con Node.JS y MongoDB |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501123 - Construir sitios web según técnicas de interoperabilidad y protocolos técnicos. | RESULTADOS DE APRENDIZAJE | 220501123-02 - Construir la API de acuerdo con los requerimientos establecidos. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 2 |
| NOMBRE DEL COMPONENTE FORMATIVO | Construcción de API RESTful Con Node.JS |
| BREVE DESCRIPCIÓN | En este componente formativo se hará uso de Node.JS para crear una API RESTful que se comunique con una base de datos MongoDB para permitir la ejecución de algunas operaciones CRUD. |
| PALABRAS CLAVE | API, REST, RESTful, Servicios Web. |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

**Introducción**

**1. Definiciones**

**2. Generalidades de Node.Js**

2.1. Instalación de herramientas

2.2. Dependencias

2.3. Comandos y direccionamiento básico

**3. Construcción API**

3.1 Conexión Node.js y MongoDB

3.2 Construcción de consultas con Node

1. **DESARROLLO DE CONTENIDOS**

**Introducción**

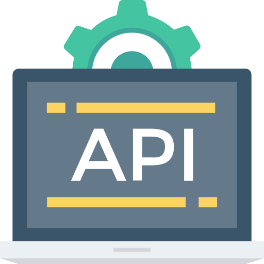
Le damos la bienvenida a este componente formativo en el que se estudiará la construcción de API RESTful con Node.JS; para conocer la importancia del tema y los diferentes componentes que se incluirán le invitamos a ver el siguiente video:



**1. Definiciones**

Como hemos mencionado, la mayoría de los sistemas de información requieren la gestión y transformación de datos, que, generalmente, se almacenan y protegen en sistemas gestores de bases de datos; sin embargo, se pueden presentar diferentes situaciones en las que se necesite refactorizar las funcionalidades ya establecidas, por lo que el uso de API RESTful es la respuesta para efectuar tales actualizaciones sin afectar el desempeño normal del sistema.

Entonces, ¿qué es una API?

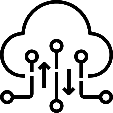


*Application Programming Interfaces* (API), que traducido literalmente al español significa Interfaz de programación de aplicaciones; es esencialmente un conjunto de especificaciones, reglas y/o mecanismos que se definen para establecer la comunicación entre dos componentes de software.

Así, una API de REST o conocida también como API RESTful es una interfaz que permite la comunicación entre dos componentes de *software*, usualmente, entre un sistema gestor de base de datos y una aplicación de *software* que la consume.

Hasta aquí hemos agregado otro término, REST, ¿a qué hace referencia?





Una de las características de REST consiste en el no almacenamiento del estado; es decir, cada vez que se realiza una solicitud de transmisión, el emisor se debe identificar por medio de credenciales o *tokens*, ya que esta información no se almacena de ninguna forma. Aunque esto parece una desventaja, se encuentra que, al no tratar con el manejo de memoria, se disminuyen los procesos requeridos para el procesamiento de los datos y lo hace altamente compatible para ser utilizado por cualquier tecnología y para entornos con necesidades de escalamiento horizontal.

Por medio de una API REST se pueden implementar los HTTPVerbs o los métodos HTTP, por medio de los cuales se especifica el tipo de solicitud que se realiza al servidor y así este puede dar el tratamiento adecuado y ofrecer respuestas correctas.

Veamos cada uno de los métodos HTTP y su función o uso:



Ahora bien, según Red Hat (2020) una API RESTful debe cumplir con los siguientes criterios:



**2. Generalidades de Node.Js**

JavaScript es, tal vez, uno de los lenguajes más comunes en el desarrollo de aplicaciones Web; generalmente, este lenguaje permite enriquecer el contenido que se muestra a los clientes por medio de un navegador web y junto a HTML y CSS completan el paquete completo de elementos que se debe combinar para ofrecer una buena experiencia de navegación por sitios WEB.

Así, es importante que conozcamos a qué hace referencia los términos HTML, CSS y JavaScript





Como observamos, JavaScript se ejecuta del lado del cliente, esto significa que utiliza los recursos del equipo de cómputo del usuario que usa el navegador web, para controlar muchos de los procesos de validaciones y de gestión de datos. En este sentido, se podría decir que **Node.js** es la versión de JavaScript que puede gestionar la ejecución de código del lado del servidor como lo hacen los lenguajes PHP, ASP.NET, JSP, entre otros.

Conozcamos algunas de las características de Node.js y JavaScript:





**2.1. Instalación de herramientas**

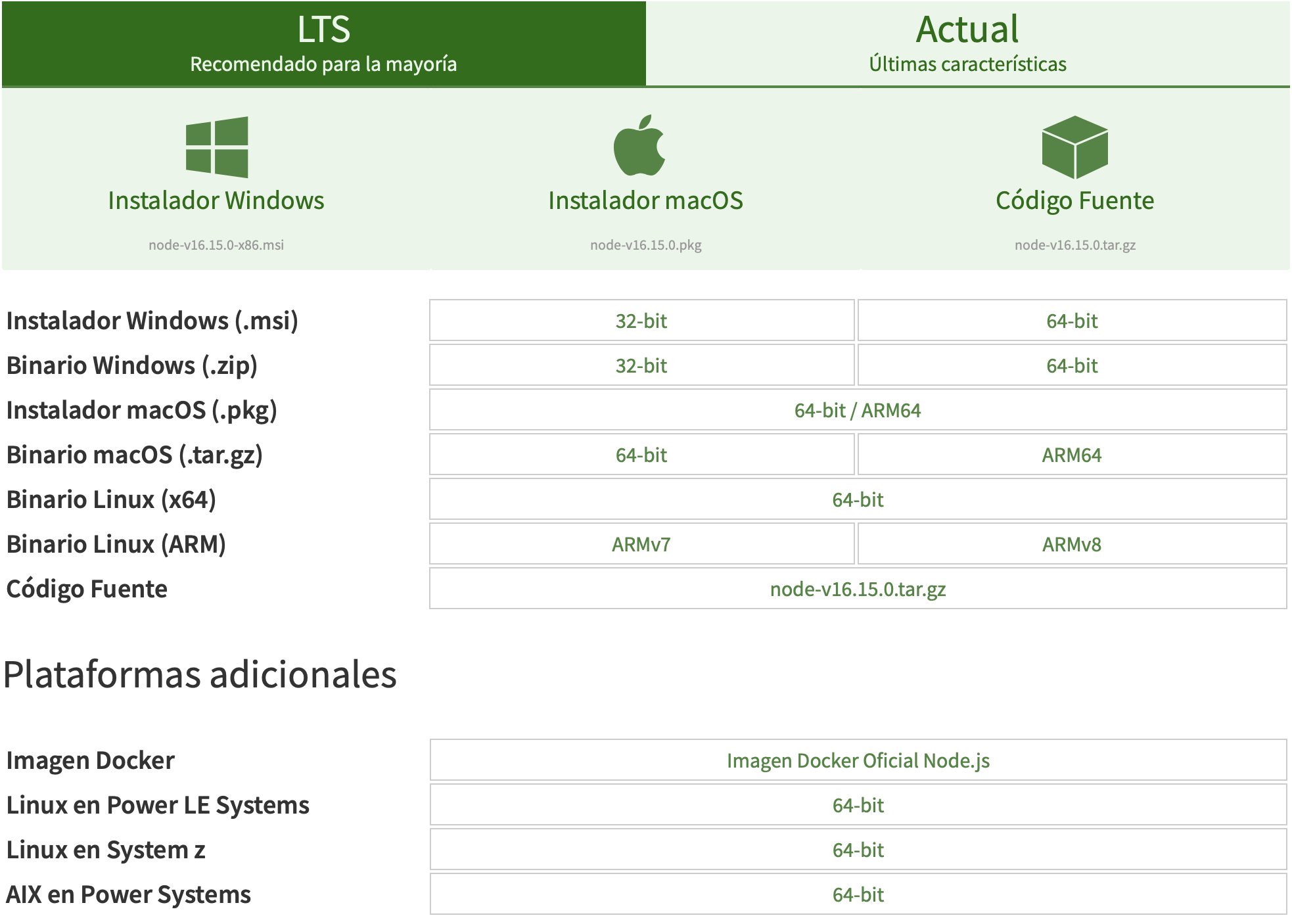
Para la instalación de Node.js se recomienda acceder a su página oficial de descargas ubicada en la dirección: <https://nodejs.org/es/download/>, en donde se presentan distribuciones para los sistemas operativos Windows y MacOS e incluso con el código fuente, recordemos que esta es una plataforma *open source*.

En la siguiente figura se observan las diferentes posibilidades de instalación habilitadas actualmente:

**Figura 1**

*Visiones instaladores Node.js*





*Nota.:* [*https://nodejs.org/es/download/*](https://nodejs.org/es/download/)

Como se observa en dicha figura, la página oficial recomienda la descarga de versiones LTS (*Long Term Support*), lo que significa que estas versiones tienen soporte de largo plazo y un funcionamiento estable, altamente validado y con gran cantidad de documentación asociada.



Es importante anotar que este campo del desarrollo *software* es muy dinámico, con permanentes actualizaciones que generan diferentes versiones; así, la última versión LTS de Node.js es la 16.15.0 y la de Node.js corresponde a la 18.1.0. Sin embargo, aunque estas herramientas son tan activas, presentan algunas generalidades que son esenciales conocer, con las cuales se garantiza su funcionamiento independiente de la versión que se descargue.

Adicionalmente, se requiere de un IDE de programación que permita el cargue y la ejecución de los paquetes de Node.js. En este punto, puede hacer uso de cualquier IDE que utilice normalmente para el desarrollo de aplicaciones afines a la WEB, particularmente se recomienda el uso de **Visual Studio Code.**

Este es un editor de código fuente que puede ser utilizado para el desarrollo de lenguajes como Python, Java, PHP, HTML/CSS, TypeScript, YAML y, por supuesto, JavaScript; tiene la posibilidad de ampliar sus funciones con plugins y buscadores de otros plugins, lo que permite una mayor personalización, según las necesidades de los usuarios, sin afectar su condición de código muy potente y liviano.

Las siguientes son algunas de las características importantes de este editor de código:

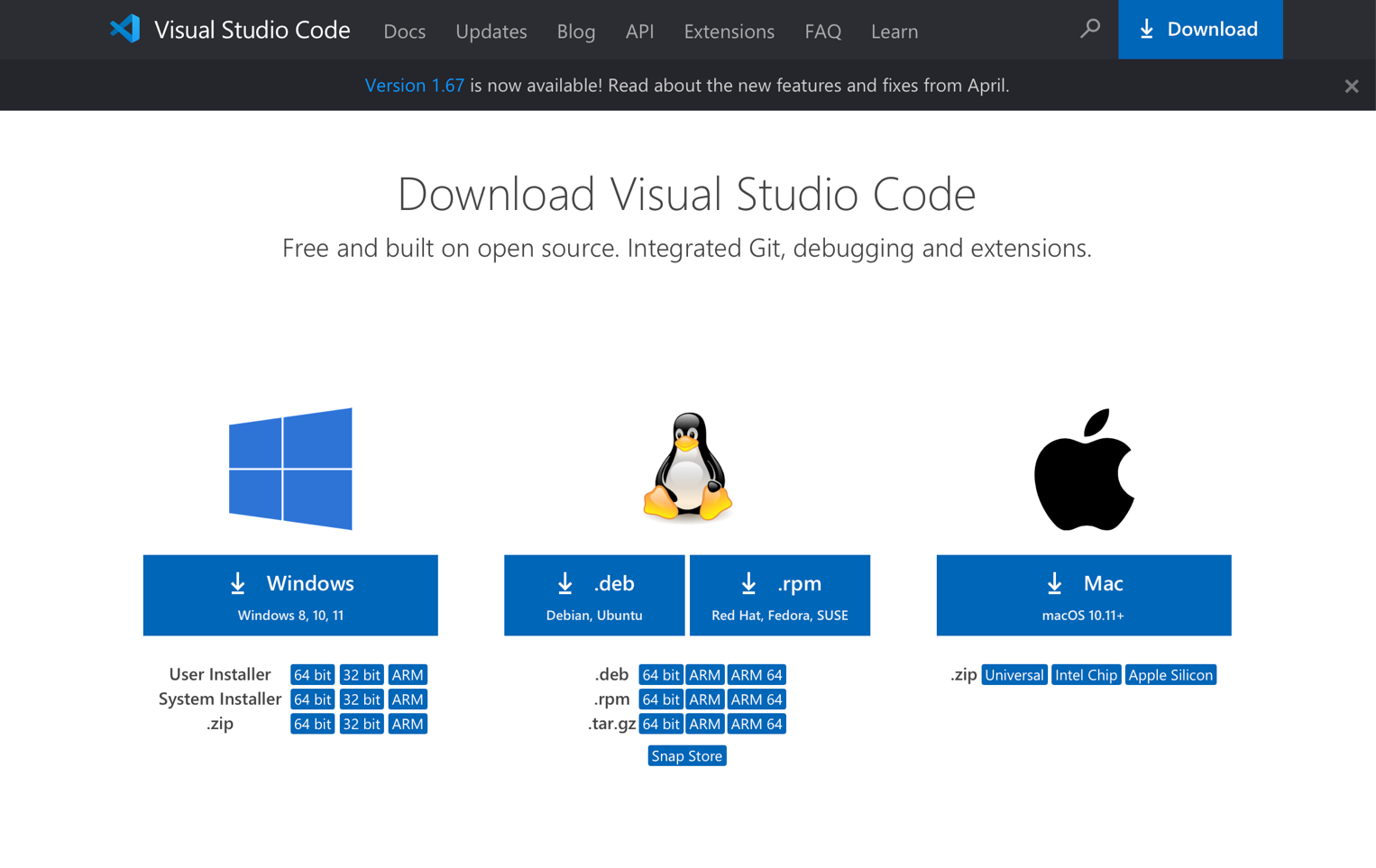
* Soporte para depuración.
* Integración con sistemas de versionamiento como GIT.
* verificación de sintaxis.
* Ayudas de finalización de código inteligente.
* Opciones de refactorización de código.

Visual Studio Code puede ser descargado de forma gratuita directamente desde su página oficial <https://code.visualstudio.com/Download>, está disponible para sistemas operativos Windows, versiones de Linux Debian, Ubuntu, Red Hat, Fedora y SUSE y sistema operativo MacOS.

Es así como a través de la siguiente figura se presenta la página oficial de Visual Studio Code con todas las opciones de descarga para cada sistema operativo soportado.

**Figura 2**

*Versiones instaladores Visual Studio Code*



*Nota.* [*https://code.visualstudio.com/Download*](https://code.visualstudio.com/Download)

Por último, se requiere de un sistema que nos permita hacer uso de la API REST para verificar su funcionamiento; recordemos que la API RESTful es un intermediario entre un sistema de base de datos y una aplicación que hace uso de estos servicios. Si bien las solicitudes que se realizan a la API se hacen por medio de HTTP, solo las solicitudes de tipo GET pueden ser verificadas por medio de un navegador web.







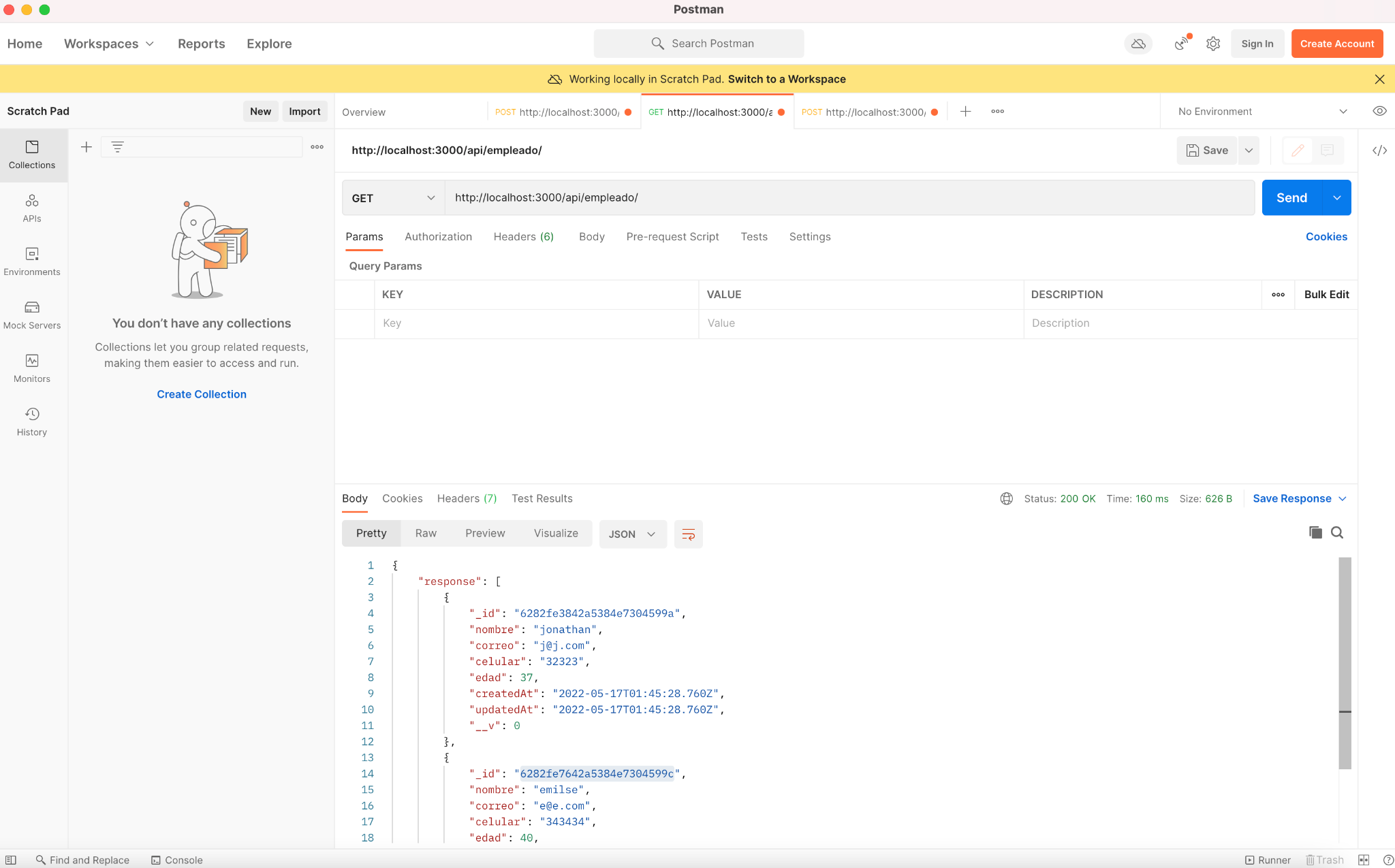


Para suplir esta necesidad, se recomienda instalar **Postman**, plataforma que facilita la construcción y prueba de APIS con una versión libre, con las funcionalidades necesarias para la verificación del funcionamiento de todos los métodos HTTP de nuestras API RESTFul. Puede utilizar esta plataforma de dos formas:

1. Descargue la versión *desktop* desde la página oficial disponible en: <https://www.postman.com/downloads/>

**Figura 3**

*Interfaz principal de Postman*



Nota.: <https://www.postman.com>

1. Realice el registro en línea, ingresando a la página oficial disponible en: <https://www.postman.com>, en este caso, le invitamos a observar el siguiente video, en el que se presenta el proceso para la instalación y configuración de Postman:



**2.2. Dependencias**

Una de las grandes potencialidades de Node.js es precisamente la facilidad de acoplar y reutilizar paquetes de JavaScript públicos que agilizan y disminuyen el tiempo requerido para la creación de servicios de alta calidad en muy poco tiempo. Node.js utiliza NPM para lograr estos resultados.

**¿Qué significa y cómo opera NPM?**

*Node Package Manager*(NPM) es el administrador de paquetes de Node y es el encargado de realizar las tres actividades fundamentales (npm, Inc)

* Ayuda a los desarrolladores JavaScript a compartir fácilmente módulos de código empaquetados.
* Sirve de colección pública de paquetes de código fuente abierto para Node.js.
* Sirve como interfaz de línea de comandos que permite instalar y publicar esos paquetes.

A continuación, se detallan algunos de estos paquetes que serán usados en la construcción de la API RestFul:



**2.3. Comandos y direccionamiento básico**

A continuación se especifican algunos de los comandos que se ejecutan en una terminal tipo Shell y son utilizados durante la creación y gestión de una aplicación desarrollada con Node.js junto a su función:

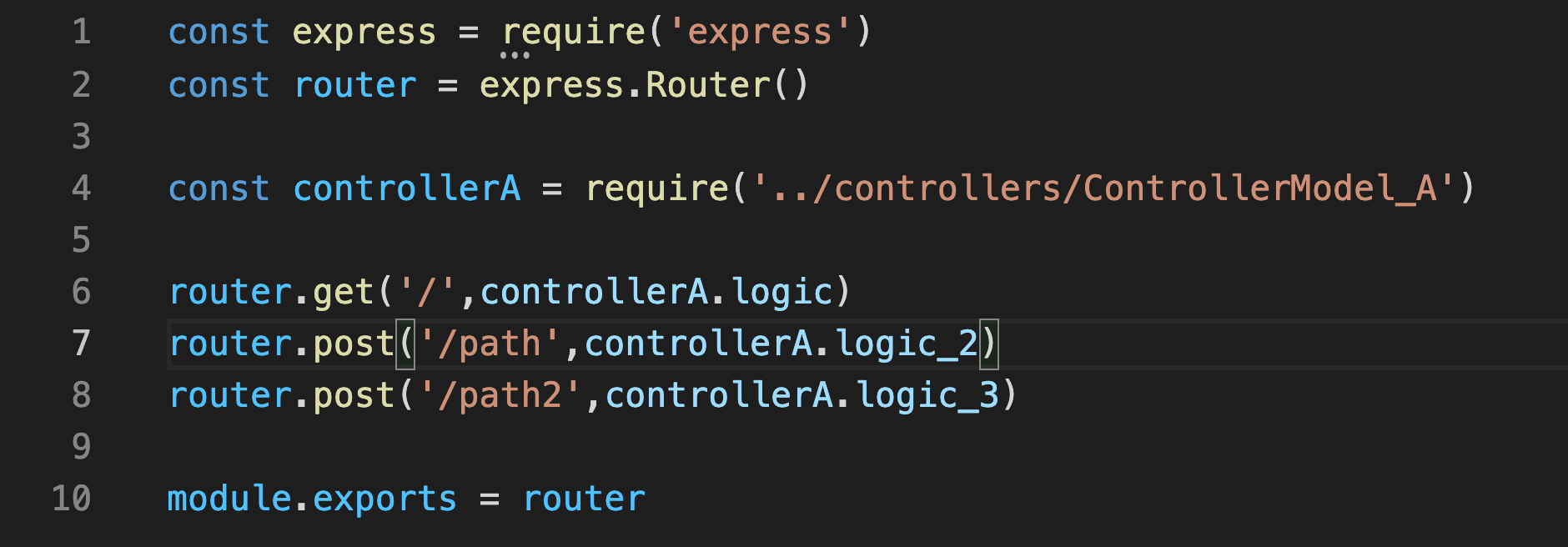


Recuerde que las Rutas o *Routes* son el concepto usado en Node.js para referirse a los componentes que permiten gestionar el direccionamiento dentro de una aplicación. Estas permiten asociar los métodos HTTP (GET, POST, PUT, DELETE, etc.) un patrón o camino URL (URL *path*) y con la función que será ejecutada para controlar dicho patrón.

En Node.js se utiliza el paquete de Express para la gestión de las rutas y tendrá una estructura similar a la que se visualiza en la siguiente figura:

**Figura 4**

*Esquema general de manejo de rutas en Node*



Como se puede observar en la imagen se necesita de una instancia de la clase Router() de express; luego, se debe instanciar el controlador que tiene definidas las acciones para un modelo específico y finalmente, con la instancia de la clase *router* definimos el método HTTP, la ruta relativa por donde será accedido desde una solicitud web a la API publicada por el servicio junto a la función específica en el controlador encargado de darle respuesta.

**3. Construcción API**

En este momento ya disponemos de las herramientas de *software* necesarias y los conceptos involucrados en la construcción de la APi RestFull, ahora veamos los siguientes aspectos a tener en cuenta:

**3.1 Conexión Node.js y MongoDB**

Lo primero que se debe realizar es la construcción de una aplicación Node.js que se pueda ejecutar satisfactoriamente y luego tratar de establecer una conexión con el servidor MongoDB que previamente ha sido configurado como sistema gestor de base de datos.

Se recomienda realizar los siguientes pasos:



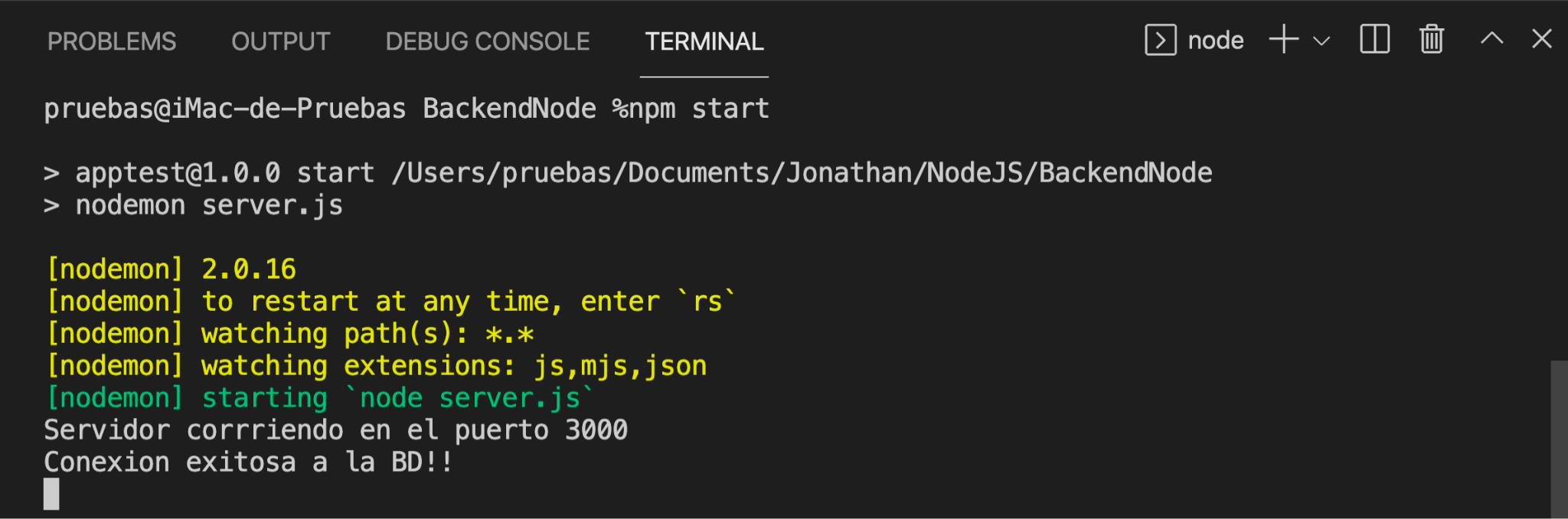
Ahora bien, en el siguiente videopor medio de un ejemplo se explica el paso a paso para crear el servidor Node.js con su configuración inicial conectado a una instancia de MongoDB provista por el servicio cloud Atlas.



Basado en la explicación anterior, en la siguiente figura se visualiza el resultado final esperado luego de realizar el proceso de creación y configuración inicial del servidor:

**Figura 5**

*Ejecución del servidor Node.js por medio de la terminal de Visual Studio Code*



**3.2 Construcción de consultas con Node**

Para la construcción del servicio RESTful se recomienda utilizar algún patrón arquitectónico que facilite la disminución de dependencias, la actualización rápida de componentes y facilidades para el escalamiento y pruebas. Tenga en cuenta que esta API es la responsable de definir el esquema de base de datos y la forma en que será manipulado por los clientes.

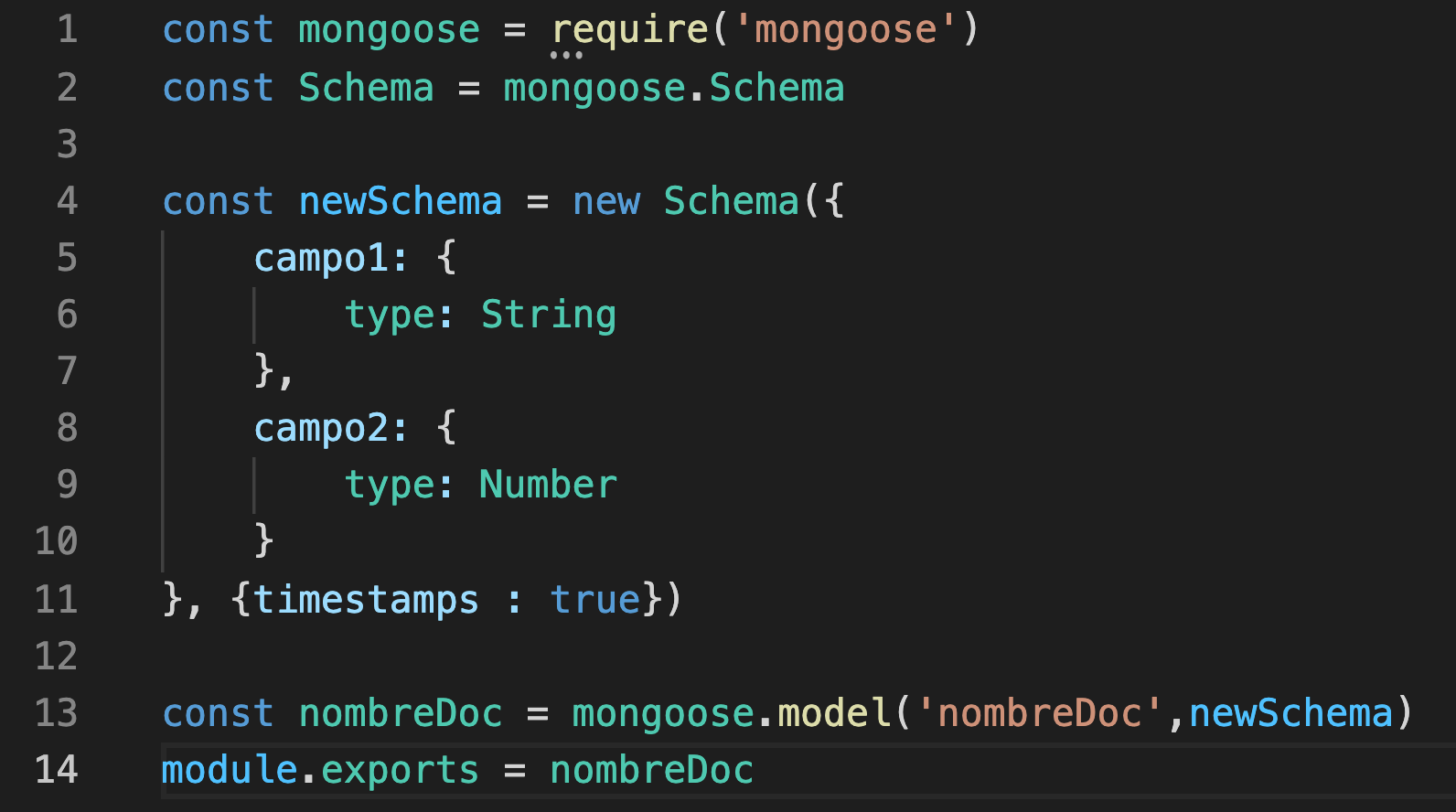
Para controlar el esquema de la base de datos deberá hacer uso de una instancia de la clase Schema de la librería mongoose.js, con el cual definirá un modelo o estructura general de los documentos que serán almacenados en MongoDB, especificando como mínimo cada uno de los campos con su correspondiente tipo de dato.



Veamos entonces a través de la siguiente figura la estructura general requerida para crear un nuevo modelo:

**Figura 6**

*Esquema general de un modelo*



Una vez definidos los modelos se deben crear los controladores que determinan las operaciones a realizar en dicho modelo, entre las más comunes se encuentran:

1. Buscar un documento específico.
2. Listar todos los documentos.
3. Registrar un nuevo documento, etc.

Los controladores harán uso de los métodos que serán emparejados con los comandos que se ejecutan del lado de MongoDB.



**Llamado a la acción**

Si desea profundizar en la funciones disponibles para los modelos de mongoose, consulte la documentación oficial disponible en: <https://mongoosejs.com/docs/models.html>

Para ejemplificar un poco este tema, en el siguiente video se presenta un ejemplo, que muestra el proceso para crear un modelo y su correspondiente controlador, con lo cual se responde a tres diferentes funciones asociadas a los métodos HTTP GET y POST:



El último paso para completar la API RESTful es la construcción de las rutas. Tenga en cuenta que estas deben ser importadas en la clase de servidor. Se recomienda tener direcciones de rutas por cada modelo del esquema de la base de datos e implementar como mínimo en el controlador todos los métodos tipo CRUD que serán utilizados por los clientes externos.

Es por ello, que en el siguiente video se muestra un ejemplo, en el que se realiza el proceso para crear las rutas de cada uno de los métodos definidos en el controlador y la forma de adecuar el servicio para hacer uso de estas rutas; también se indica cómo utilizar la herramienta Postman para hacer las pruebas manuales de cada una de las rutas definidas:



En el ejemplo anterior solo se implementaron las solicitudes HTTP de tipo GET y POST, la implementación de los otros métodos HTTP que complementan el CRUD siguen la misma dinámica; por ende, con los conocimientos adquiridos es posible completar el ejercicio sin ningún problema.

Recordemos que las operaciones que se pueden realizar sobre los documentos están establecidas en la clase controlador que importa el modelo particular sobre el que se realizarán las opciones. Así, se presentan algunas de estas funciones de Mongoose que pueden ejecutar acciones del lado de la base de datos MongoDB, conózcanoslas:

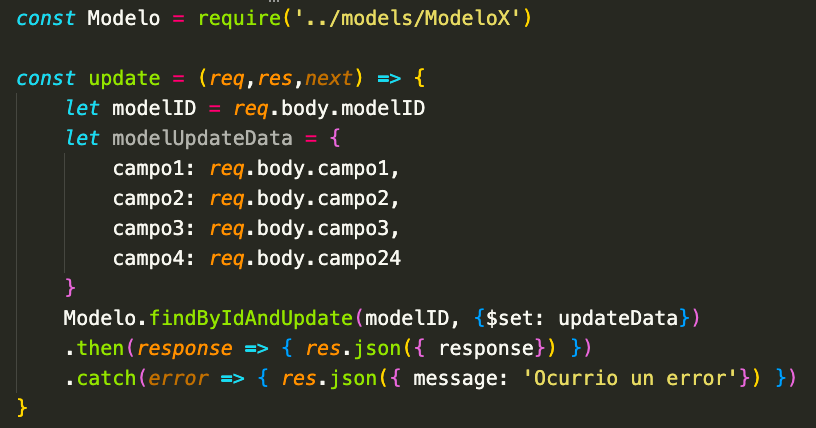


En la siguiente figura se muestra un borrador de la estructura que tendría una función en el controlador para actualizar una colección de tipo Modelo X, se puede observar que en el controlador se debe importar primero el modelo para poder invocar la función findByIDAndUpdate(), la cual recibirá como parámetro el id del documento a actualizar y el documento con toda su estructura de cómo debería quedar luego de la actualización. El método findByIDAndUpdate() asocia a la palabra reservada $set los valores a actualizar. Tenga en cuenta que cualquier campo de la colección puede ser actualizado excepto el identificador.

Los parámetros utilizados en la consulta son obtenidos desde el *body* del objeto de tipo *request,* es decir, que en el momento de hacer la solicitud del servicio se deberá hacer uso de un método de tipo POST o similar que acepte el envío de parámetros. Observe dicha figura:

**Figura 7**

*Esquema general para la actualización de un documento de un modelo específico*

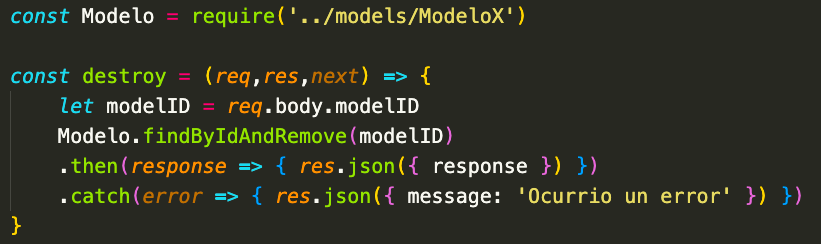


Por último, en la siguiente figura, se muestra un borrador de la estructura de una función que permite la eliminación de un documento, de acuerdo con un número de identificador dado. Al igual que las demás operaciones vistas, se requiere la importación del modelo sobre el que se va a realizar el proceso de eliminación y la invocación de la función findByIDAndRemove()

El parámetro del identificador debe ser capturado desde el *body* del objeto *request*, lo que implica que la invocación de este servicio se realizará por medio de un método POST o similar que acepte el envío de parámetros en el cuerpo de la solicitud enviada al servidor. Dicha figura es:

**Figura 8**

*Esquema general para la eliminación de un documento de un modelo específico*



*.*

Finalmente, recuerde que cada función del controlador debe ser explotada al módulo por medio del comando:

module.exports = {fun1, fun2, … fun n}

En el que fun1, fun2 y fun n corresponden a las funciones implementadas para la gestión de operaciones sobre los modelos construidos y que, adicionalmente, cada una de estas funciones deberán tener su respectiva representación en el archivo de rutas.

Se recomienda usar las indicaciones del video **“Pruebas y rutas con Postman”** para completar el ejercicio de habilitar las operaciones de actualización y eliminación de colecciones y así completar todas las operaciones CRUD del Backend de Ejemplo.

**C. SÍNTESIS**

El Node.js se considera un entorno de código abierto para JavaScript; asíncrono, diseñado para crear aplicaciones network escalables, por lo que se hace necesario conocer a profundidad sus generalidades, dependencias, comandos, herramientas y direccionamiento básico, entre otras cosas. De igual manera se debe conocer de construcción API con el fin de poder crear una API RESTful que logre una comunicación con MongoDB. Por esta razón este componente formativo profundizó estos temas, los cuales se resumen a través del siguiente esquema:



**D. ACTIVIDADES DIDÁCTICAS (OPCIONALES SI SON SUGERIDAS)**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la actividad | Generalidad de Node.js |
| Objetivo de la actividad | Fortalecer algunos conceptos relacionados con Node.js requeridos para la construcción de una API RESTful, de acuerdo con los requisitos del cliente. |
| Tipo de actividad sugerida | Completar espacios |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexos / Actividad didáctica CF02 |

**E. MATERIAL COMPLEMENTARIO**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del recurso o  archivo del documento o material |
| 2.3. Comandos y direccionamiento básico | © OpenJS Foundation. (s. f.). *Index | Node.js v18.1.0 Documentation*. | Página web | <https://nodejs.org/api/> |

**F**. **GLOSARIO**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| Middleware: | en Node.js, hace referencia a un bloque de código que es ejecutado en el medio desde que se hace una solicitud hasta que llega al servidor. |
| ORM: | modelo de programación que consiste en mapear tablas de un modelo relacional a objetos en el paradigma de programación orientada a objetos. |
| Shell: | ventana de comandos |

**G.** **REFERENCIAS BIBLIOGRÁFICAS**

npm, Inc. (s.f.). *npm*. [https://Www.Npmjs.Com](https://www.npmjs.com)

Red Hat, Inc. (2020,). *¿Qué es una API de REST?* <https://www.redhat.com/es/topics/api/what-is-a-rest-api>

Stack Overflow. (2021a). *Stack Overflow Developer Survey 2021*. <https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language-prof>

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha |
| Autor (es) | Jonathan Guerrero Astaiza | Experto Temático | Regional Cauca – Centro de Teleinformática y Producción Industrial | Mayo de 2022 |
| María Fernanda Chacón Castro | Diseñador Instruccional | Regional Distrito Capital - Centro de Gestión Industrial | Mayo de 2022 |
| Andrés Felipe Velandia Espitia | Asesor Pedagógico | Regional Distrito Capital - Centro de Diseño y Metrología | Mayo de 2022 |
| Rafael Neftalí Lizcano Reyes. | Responsable Equipo Desarrollo Curricular | Regional Santander - Centro Industrial del Diseño y la Manufactura | Mayo de 2022 |
| Sandra Patricia Hoyos Sepúlveda | Corrección de estilo | Regional Distrito Capital - Centro de Diseño y Metrología | Agosto de 2022 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del cambio |
| Autor (es) |  |  |  |  |  |