Objetivo: relacionar los conceptos previos a la generación de código en la construcción de API utilizando

express y dependencias necesarias.

Desarrolle cada punto en una presentación y demuestre con ejemplos.

¿ Que es Rounting ?

R// El **routing** (o enrutamiento) es un concepto utilizado en desarrollo web y de software que se refiere al mecanismo para gestionar y direccionar las peticiones de los usuarios a diferentes partes de una aplicación, según las URLs o rutas específicas. En aplicaciones web, el routing permite que los usuarios naveguen entre diferentes páginas o vistas de una aplicación sin necesidad de recargar toda la página.

**Enrutamiento en Aplicaciones Web**

1. **Cliente (Frontend)**: En aplicaciones de una sola página (SPA) como las hechas con React, Vue o Angular, el routing permite cambiar de vista o de página sin necesidad de recargar el navegador. Esto se logra con librerías de routing, como:
   * **React Router** para React
   * **Vue Router** para Vue
   * **Angular Router** Para Angular

En este tipo de enrutamiento, el navegador interpreta la URL y muestra la vista o componente correspondiente sin hacer una nueva solicitud al servidor.

1. **Servidor (Backend)**: En aplicaciones tradicionales o en el backend, el routing se encarga de dirigir las peticiones HTTP a diferentes controladores o manejadores, según la URL de la petición. Por ejemplo:
   * En **Express** (Node.js), puedes configurar rutas como app.get('/usuarios', handler), para manejar solicitudes a /usuarios.
   * En **Django** o **Flask** (Python), puedes definir rutas que se ejecutan cuando un usuario accede a una URL específica, como /productos.

**Tipos de Enrutamiento**

1. **Enrutamiento basado en servidor**: Cada vez que se navega a una nueva URL, se hace una solicitud al servidor, que responde con una página completa.
2. **Enrutamiento basado en cliente (SPA)**: La navegación entre páginas se gestiona en el cliente, y solo se cargan las partes necesarias, proporcionando una experiencia de usuario más rápida y fluida.

En resumen, el routing es clave para controlar cómo los usuarios navegan y acceden a diferentes partes de una aplicación, permitiendo que las aplicaciones web sean más interactivas y fáciles de manejar.

Que recursos se pueden encontrar en <https://www.npmjs.com/>?

R// En [npmjs.com](https://www.npmjs.com/), el sitio oficial de **Node Package Manager (npm)**, se pueden encontrar recursos como **paquetes, bibliotecas, y herramientas** que facilitan el desarrollo de aplicaciones en JavaScript y Node.js. A continuación, algunos de los principales recursos disponibles:

### 1. ****Paquetes (Packages)****

* **Módulos y bibliotecas**: npm es el repositorio más grande de paquetes JavaScript del mundo. Hay bibliotecas para casi cualquier funcionalidad que puedas necesitar: desde herramientas para manipulación de datos y DOM hasta gestión de peticiones HTTP y autenticación.
* **Frameworks y Librerías Populares**: Paquetes como express (para servidores web), react, vue, y angular (para front-end) están disponibles en npm.
* **Utilidades de Node.js**: Herramientas específicas para Node.js, como nodemon para reiniciar aplicaciones automáticamente, dotenv para cargar variables de entorno, y mongoose para trabajar con bases de datos MongoDB.

### 2. ****Herramientas de Desarrollo****

* **Herramientas de testing**: npm cuenta con herramientas como jest, mocha, y chai, que son populares para realizar pruebas unitarias, de integración, y de regresión.
* **Transpiladores y Compiladores**: Paquetes como babel ayudan a convertir el código JavaScript a versiones compatibles con más navegadores.
* **Empaquetadores y Compiladores Frontend**: webpack, rollup, y parcel son paquetes disponibles en npm para optimizar, empaquetar, y compilar aplicaciones web.

### 3. ****Scripts de Automatización****

* Los paquetes como gulp, grunt, y npm-run-all son ampliamente usados para automatizar tareas como minificación de archivos, pruebas automáticas, y compilación de código en aplicaciones.

### 4. ****Dependencias de Aplicación y Desarrollo****

* npm permite gestionar dependencias específicas de desarrollo (devDependencies) y dependencias necesarias para el entorno de producción (dependencies). Muchos paquetes están diseñados para facilitar tareas exclusivas de desarrollo, como eslint para linting y prettier para formateo de código.

### 5. ****Documentación y Ejemplos****

* Cada paquete incluye una página detallada con la documentación del paquete, que explica cómo instalarlo, sus funcionalidades principales, e instrucciones de uso.

### 6. ****Componentes de UI****

* npm también tiene bibliotecas de componentes de interfaz de usuario, como bootstrap, material-ui, y antd (Ant Design), que ayudan a crear interfaces rápidas y responsivas para aplicaciones web.

### 7. ****Plugins para Frameworks y CMS****

* Plugins que extienden la funcionalidad de otros frameworks y plataformas, como gatsby (para crear sitios web estáticos), plugins para next.js y plugins específicos para CMS como wordpress y strapi.

### 8. ****APIs de Integración****

* Bibliotecas y SDKs para interactuar con APIs de terceros, como axios (para peticiones HTTP), stripe (procesamiento de pagos), y firebase.

### 9. ****Ejemplos y Starter Kits****

* npm ofrece paquetes que son plantillas de proyectos o "starter kits" listos para usar, como create-react-app para React, express-generator para Express, o vue-cli para Vue.js.

### 10. ****Actualizaciones y Versiones****

* npm permite ver las versiones anteriores y actualizaciones recientes de cada paquete, lo que facilita mantener el código actualizado con las últimas mejoras y correcciones.

Que son Dependencias de Desarrollo y de aplicación?

R// En desarrollo de software, especialmente cuando usamos herramientas como **Node.js** o **Python**, es común encontrar dos tipos de dependencias: **dependencias de desarrollo** y **dependencias de aplicación** (también llamadas **dependencias de producción**).

**1. Dependencias de Aplicación (Producción)**

Las dependencias de aplicación son las librerías y módulos necesarios para que una aplicación funcione correctamente en un entorno de producción. Son aquellas que el código necesita en todo momento, incluso cuando la aplicación está en uso por los usuarios finales.

* **Ejemplos**:
  + En una aplicación de Node.js, una dependencia de aplicación podría ser una librería de bases de datos como mongoose (para MongoDB) o express (para manejar rutas en un servidor).
  + En una aplicación de Python, podría ser flask o django, que se necesitan para que la aplicación funcione en un entorno de producción.

**2. Dependencias de Desarrollo**

Las dependencias de desarrollo son aquellas necesarias solo durante el proceso de desarrollo. Estas incluyen herramientas y librerías que ayudan en tareas como compilación, pruebas, o linters, pero no son necesarias para ejecutar la aplicación en producción.

* **Ejemplos**:
  + En una aplicación de Node.js, dependencias como eslint (para verificar el código), jest (para pruebas), o webpack (para empaquetar la aplicación) suelen ser dependencias de desarrollo.
  + En una aplicación de Python, herramientas de pruebas como pytest o tox, o de formateo de código como black, son ejemplos de dependencias de desarrollo.

¿ Que puertos lógicos existen, cuales son los puertos que utilizan las aplicaciones de desarrollo y

frameworks node, React Angular, Vue, Docker.

R// Los **puertos lógicos** en redes y desarrollo de aplicaciones son números asignados a procesos o servicios específicos para permitir la comunicación a través de una red, como Internet o una red local. Los puertos están divididos en tres rangos principales:

**Rango de Puertos Lógicos**

1. **Puertos Bien Conocidos (0-1023)**:
   * Reservados para servicios y protocolos conocidos.
   * Ejemplos:
     + **80**: HTTP
     + **443**: HTTPS
     + **21**: FTP
     + **22**: SSH
     + **25**: SMTP
2. **Puertos Registrados (1024-49151)**:
   * Usados por aplicaciones de software que requieren un puerto específico.
   * Ejemplos:
     + **3306**: MySQL
     + **5432**: PostgreSQL
     + **27017**: MongoDB
     + **8080**: HTTP alternativo (común en desarrollo)
3. **Puertos Dinámicos o Privados (49152-65535)**:
   * Utilizados de manera dinámica para conexiones temporales o asignaciones aleatorias por el sistema operativo.
   * Ejemplo: conexiones de cliente para navegadores web y aplicaciones.

**Puertos Comunes en Aplicaciones y Frameworks de Desarrollo**

1. **Node.js**:
   * **3000**: El puerto por defecto en aplicaciones Node.js (aunque se puede cambiar a otros puertos).
   * Otros puertos utilizados para APIs, dependiendo de la configuración del servidor.
2. **React**:
   * **3000**: El puerto por defecto al ejecutar un proyecto de desarrollo de React (utilizando npm start o yarn start).
   * En caso de conflicto (por ejemplo, si Node.js ya está en uso en 3000), React suele proponer un puerto alternativo (por ejemplo, 3001, 3002, etc.).
3. **Angular**:
   * **4200**: El puerto por defecto para el servidor de desarrollo de Angular (ng serve).
   * Este puerto también se puede cambiar según las necesidades del proyecto.
4. **Vue**:
   * **8080**: El puerto por defecto cuando se utiliza vue-cli-service serve en desarrollo.
   * Puede cambiarse configurando el proyecto en el archivo de configuración o en la línea de comandos.
5. **Docker**:
   * **2375**: Puerto para la API de Docker en TCP (sin seguridad).
   * **2376**: Puerto para la API de Docker en TCP (con TLS).
   * Los contenedores y servicios de Docker suelen configurarse en puertos específicos según las aplicaciones en ejecución. Por ejemplo, una aplicación web en un contenedor puede mapear el puerto 80 o 3000 del contenedor al mismo puerto en la máquina host.
6. **Bases de datos y otros servicios en desarrollo**:
   * **MongoDB**: 27017
   * **PostgreSQL**: 5432
   * **MySQL**: 3306
   * **Redis**: 6379

Para que sirve nodemon?

R// **Nodemon** es una herramienta que facilita el desarrollo de aplicaciones en **Node.js** al monitorear automáticamente los cambios en los archivos del proyecto y reiniciar el servidor cada vez que detecta alguna modificación. Esto elimina la necesidad de reiniciar manualmente el servidor después de cada cambio, ahorrando tiempo y mejorando la productividad.

**Principales Ventajas de Usar Nodemon**

1. **Auto-reinicio del servidor**:
   * Nodemon observa los archivos en tu proyecto (por defecto, los archivos .js y otros relevantes) y, cuando detecta un cambio, automáticamente detiene y reinicia el servidor de Node.
   * Esto es útil en aplicaciones en las que constantemente haces ajustes en el código y necesitas ver los resultados rápidamente.
2. **Ahorro de tiempo en desarrollo**:
   * En lugar de detener el servidor manualmente y reiniciarlo después de cada cambio, Nodemon lo hace automáticamente, permitiéndote centrarte solo en escribir y probar el código.
3. **Configuración flexible**:
   * Puedes especificar qué archivos o carpetas debe observar y cuáles ignorar.
   * Es posible personalizar el comando a ejecutar y configurar scripts específicos si necesitas ejecutar algo más que node app.js.

Para que sirve Postman y Thunder Client?

R// **Postman** y **Thunder Client** son herramientas de desarrollo que se utilizan principalmente para probar **APIs** (interfaces de programación de aplicaciones). Ambas herramientas permiten hacer solicitudes HTTP (como GET, POST, PUT, DELETE, etc.) a servicios web y examinar las respuestas, lo que ayuda a los desarrolladores a **debuguear, verificar y desarrollar** sus APIs de manera eficiente.

### ****Postman****

**Postman** es una de las herramientas más populares y completas para probar APIs. Se trata de una aplicación independiente que ofrece una interfaz gráfica de usuario para interactuar con servicios web.

#### Características principales de **Postman**:

1. **Pruebas de APIs**: Permite hacer solicitudes HTTP a cualquier endpoint de una API y ver la respuesta en tiempo real.
2. **Soporta múltiples métodos HTTP**: Puedes hacer solicitudes GET, POST, PUT, DELETE, PATCH, y más.
3. **Autenticación y Autorización**: Ofrece soporte para autentificación con tokens, API keys, OAuth, Basic Auth, entre otros.
4. **Interfaz gráfica avanzada**: Incluye características como la creación de colecciones de pruebas, variables, y entornos (por ejemplo, para distintos entornos de desarrollo, prueba y producción).
5. **Pruebas automatizadas**: Puedes escribir scripts para automatizar las pruebas o validar las respuestas de la API (por ejemplo, comprobar el código de estado HTTP, verificar los datos devueltos, etc.).
6. **Monitoreo de APIs**: Puedes configurar la ejecución automática de colecciones de pruebas de APIs a intervalos regulares.
7. **Documentación automática**: Postman puede generar documentación interactiva para tus APIs de manera automática.
8. **Importación y exportación**: Permite importar y exportar colecciones de pruebas, lo que facilita compartir configuraciones entre equipos.

#### Ejemplo de uso de Postman:

1. Seleccionas el método HTTP (GET, POST, PUT, etc.).
2. Ingresas la URL del endpoint de la API.
3. Configuras parámetros, cabeceras, o datos del cuerpo de la solicitud si es necesario.
4. Enviando la solicitud, Postman te muestra la respuesta del servidor con el código de estado, los encabezados y el cuerpo de la respuesta.

### ****Thunder Client****

**Thunder Client** es una extensión ligera para el editor de código **Visual Studio Code (VSCode)** que ofrece funcionalidades similares a las de Postman, pero de una forma más simple e integrada dentro del flujo de trabajo de los desarrolladores que usan VSCode.

#### Características principales de Thunder Client:

1. **Integración en VSCode**: Está directamente integrado en VSCode, lo que significa que no necesitas salir de tu editor de código para probar tus APIs.
2. **Interfaz sencilla**: Ofrece una interfaz más minimalista y simple en comparación con Postman, lo que puede resultar atractivo para desarrolladores que buscan una herramienta liviana.
3. **Soporta múltiples métodos HTTP**: Al igual que Postman, Thunder Client permite enviar solicitudes GET, POST, PUT, DELETE, PATCH, etc.
4. **Autenticación**: Soporta autenticación mediante tokens de tipo Bearer, API keys, y Basic Auth.
5. **Variables y entornos**: Puedes configurar variables globales y de entorno, lo que facilita el cambio de configuración para diferentes entornos (desarrollo, producción).
6. **Historial de solicitudes**: Guarda un historial de todas las solicitudes realizadas para que puedas revisar rápidamente cualquier solicitud anterior.
7. **Soporte para colecciones**: Permite organizar tus solicitudes en colecciones, lo que es útil para estructurar pruebas o diferentes partes de una API.

Que es un template engine y cuales existen?

R// Un **template engine** (motor de plantillas) es una herramienta que permite crear HTML dinámico en aplicaciones web al mezclar lógica de programación con código HTML. Básicamente, facilita la generación de páginas web con contenido dinámico al permitir que los desarrolladores inserten variables y estructuras de control (como bucles y condiciones) en plantillas HTML.

**¿Para Qué Sirve un Template Engine?**

Los motores de plantillas son útiles para:

* **Generar HTML dinámico**: Permiten mostrar datos de manera dinámica en la web, como listas de productos, información de usuarios, o resultados de búsqueda.
* **Separar la lógica de la presentación**: Facilitan la separación entre el código de la aplicación y la estructura de presentación HTML, haciendo que el desarrollo sea más limpio y fácil de mantener.
* **Facilitar el diseño de interfaces de usuario**: Ayudan a diseñar vistas personalizadas con datos específicos de cada usuario o contexto.

Que es una Función Callback?

R//Una **función callback** es una función que se pasa como argumento a otra función para que sea ejecutada en un momento posterior. Las callbacks son especialmente útiles en programación asincrónica, permitiendo que el código continúe ejecutándose mientras se esperan operaciones que pueden tardar, como llamadas a APIs, consultas a bases de datos o temporizadores.

### ¿Por Qué Usar Funciones Callback?

Las callbacks son útiles en situaciones en las que una operación toma tiempo en completarse, y no queremos bloquear la ejecución del código hasta que termine. En lugar de esperar, podemos definir una función que se ejecutará cuando la operación termine, permitiendo que el resto del código siga funcionando.

¿Qué son los endpoints?

R// Los **endpoints** son las direcciones o puntos de acceso de una API (Interfaz de Programación de Aplicaciones) a los que se pueden hacer peticiones desde una aplicación cliente para interactuar con un servidor. Cada endpoint permite acceder a diferentes recursos y realizar acciones específicas, como obtener, crear, actualizar o eliminar datos en el servidor.

Que Métodos HTTP hay y como funcionan en JavaScript, cuales son los más comunes.?

R// Los **métodos HTTP** son verbos que especifican la acción que se desea realizar sobre un recurso en el servidor cuando se hace una solicitud desde el cliente. Los métodos más comunes son **GET**, **POST**, **PUT**, y **DELETE**, aunque existen otros menos utilizados, como **PATCH**, **HEAD**, y **OPTIONS**. Cada método tiene un propósito específico y se usa en diferentes escenarios según el tipo de operación que se quiera realizar sobre el recurso.

¿Qué ocurre cuando se ejecuta res.render()?

R//Cuando se ejecuta **res.render()** en **Node.js** con el framework **Express**, se está generando y enviando una respuesta **HTML** al cliente, basándose en una plantilla o vista (template) que ha sido previamente configurada.

En términos más específicos:

**¿Qué hace res.render()?**

1. **Renderiza una vista**: **res.render()** busca una vista o plantilla (template) dentro de un directorio que contiene las vistas de la aplicación y la renderiza para generar HTML.
   * La vista generalmente está escrita en un motor de plantillas como **EJS**, **Pug** (anteriormente conocido como Jade), **Handlebars**, entre otros.
   * El motor de plantillas genera HTML dinámicamente, sustituyendo las variables o bloques de código con los datos proporcionados en el objeto que pasas como segundo argumento a **res.render()**.
2. **Envia la respuesta al cliente**: Una vez que la plantilla ha sido renderizada con los datos que se le pasan, se envía esa respuesta **HTML** al navegador del cliente, lo que permite que la página web se renderice visualmente.

Cuales son las Ventajas de usar render() y un motor de plantillas?

### R// 1. Generación dinámica de contenido

* **Ventaja**: Permite generar contenido dinámico en función de los datos que se le pasen a la vista. Los motores de plantillas como **EJS**, **Pug** o **Handlebars** permiten que el contenido de las páginas web cambie en función de las variables, datos de base de datos, o interacciones del usuario.
* **Ejemplo**: Si tienes una lista de productos en tu base de datos, puedes renderizar automáticamente cada uno en una página HTML sin tener que escribir manualmente el HTML para cada uno.

**2. Separation of Concerns (Separación de responsabilidades)**

* **Ventaja**: Facilita la separación entre la lógica del servidor (backend) y la presentación (frontend). Esto significa que puedes mantener el código HTML (la presentación) separado de la lógica de negocio que procesa los datos y maneja las solicitudes.
* **Ejemplo**: La lógica de la aplicación (obtener datos de la base de datos) se maneja en el servidor, mientras que las vistas y el HTML se mantienen en archivos de plantillas.

**3. Reutilización de vistas**

* **Ventaja**: Las plantillas permiten reutilizar partes de la página (como cabeceras, pies de página, menús, etc.) sin tener que escribir HTML repetidamente. Puedes crear un archivo de plantilla común (por ejemplo, un archivo de header.ejs o footer.ejs) y luego incluirlo en varias páginas.
* **Ejemplo**: Si tienes un pie de página común para todas tus páginas, solo lo escribes una vez y lo incluyes en las demás plantillas.

**4. Mejora de la legibilidad del código**

* **Ventaja**: Las plantillas proporcionan una sintaxis limpia y sencilla para insertar variables y estructuras lógicas en el HTML, lo que hace que el código sea más legible y fácil de mantener en comparación con insertar HTML manualmente dentro de archivos JavaScript.