Actividad 3: Node.js
Comparación de frameworks a nivel de aplicación

1)

a. Realizar una breve descripción de las características principales (Seleccionar los 3 o 4 más utilizados en la actualidad) resaltando ventajas y desventajas.

- **Express.js:**
  Express.js es el framework de Node.js más utilizado. Es un framework minimalista que permite crear aplicaciones Web y APIs. Posee una arquitectura rápida, robusta y asíncrona.
  Su API permite a los usuarios configurar rutas para enviar/recibir peticiones entre el front-end y la base de datos (actuando como un marco de trabajo del servidor HTTP). Una buena ventaja de express es que soporta muchos otros paquetes y otros motores de plantillas como Pug, Mustache, EJS y muchos más.
  Express se alinea con operaciones de E/S de alta velocidad y con la naturaleza “single-threaded” (un sólo hilo) de Node, haciéndolo casi un requerimiento por default para apps desarrolladas con Node.js

  Características:

  - Paquetes veloces del lado del servidor. Express incorpora muchas características de node como funciones que pueden acelerar el proceso con pocas líneas de código
  - Alta performance, múltiples operaciones pueden ser ejecutadas independientemente de otras utilizando programación asíncrona
  - Alta cobertura de testing
  - Gran abanico de herramientas de HTTP, lo que resulta en programas más reusables y fáciles de entender
  - Better content negotiation- this helps in better communication between the client and server by providing HTTP headers to URLs, which fetch the exact information for the users/client-side.
  - Patrón MVC

  Cuándo usar Express.js
  Se pueden desarrollar aplicaciones más rápido ya que cuenta con bases disponibles para la generación de APIs. Se puede usar en casi cualquier nivel empresarial porque tiene un enrutamiento robusto, plantillas, infraestructura de seguridad y soporte de errores.
  Resulta adecuado para cualquier tipo de aplicaciones web o móvil de cualquier tamaño. Para los equipos de desarrolladores principiantes esta es el mejor framework porque cuentan con una comunidad disponible para apoyarlos

- **Hapi.js**

Hapi.js es un framework open-source para aplicaciones web. Es utilizado para crear servidores proxy, REST APIs y otras aplicaciones de escritorio ya que el framework es conocido por su confianza y alto contenido en todo lo que se refiera a seguridad. Tiene una basta cantidad de plugins incorporados para evitar utilizar middleware no oficial.

Características:

- Aplicaciones escalables
- Mínimos gastos generales
- Medidas de seguridad por default
- Rico ecosistema
- Rapido y facil arreglo de errores
- Compatibilidad con MySQL, MongoDB y otras bases de datos
- Compatible con REST APIs y aplicaciones HTTPS proxy
- Cacheo, autenticación y validación de entrada por default

Cuándo utilizar Hapi.js
Hapi.js se puede utilizar cuando se quiere desarrollar con mayores medidas de seguridad, se desea un sistema escalable, en tiempo real y/o con aplicaciones orientadas al social-media. Los desarrolladores usualmente lo utilizan para crear servidores proxies y APIs.

- **Socket.io**

Socket.io es una librería de JavaScript utilizada para desarrollar aplicaciones en tiempo real y establecer una comunicación bidireccional entre clientes y servidores. También es utilizado para construir aplicaciones con requerimientos de desarrollo de websocket. Por ejemplo, aplicaciones de chat como WhatsApp, que necesitan estar funcionando continuamente y actualizando su contenido en tiempo real, recargando los procesos de fondo para capturar las actualizaciones o mensajes. También ofrece análisis de la aplicación en tiempo real con unas pocas líneas de código.

Más de mil compañías, incluidas Bepro, Barogo y Patreon utilizan esta librería.

Características:

- Soporte binario (con una librería del lado del cliente y otra del lado del servidor)
- Soporte de multiplexación
- Confiabilidad
- Soporte de auto-reconexión
- Detección de errores y autocorrección
- Similar APIs for a client and server-side development

Cuándo utilizar Socket.io
Socket.io permite desarrollar aplicaciones en tiempo real en las que los servidores necesitan enviar datos sin ser éstos requeridos por el lado del cliente (aplicaciones de chat, de videoconferencias, juegos multijugador)

- **Sails.js**
  Sails.js se asemeja a la arquitectura MVC con patrones vistos en otros frameworks como Ruby on Rails y provee soporte para un desarrollo moderno y orientado a los datos. Es compatible con todas las bases de datos y flexible integrando frameworks de Javascript. Las APIs están basadas en datos, con una arquitectura escalable orientada a servicios.

  Es muy práctico para crear aplicaciones customizadas de alto nivel. Sus políticas para la escritura de código ayudan a reducir la cantidad necesitada de código, permitiendo la integración con módulos NPM al ser más flexible y abierto.

  Si bien es una plataforma con un frontend-agnostic-backend, este framework utiliza Express para lo que son las request HTTP y Socket.io para los WebSockets

  También permite compartir la misma API utilizada por otro servicio web u otro equipo de desarrollo, lo cual ayuda disminuyendo los tiempos y el esfuerzo.

  Sails agrupa un ORM, que hace posible la compatibilidad con casi todas las bases de datos, llegando incluso a proporcionar un gran número de proyectos comunitarios. Algunos de sus adaptadores oficialmente soportados incluyen MYSQL, MongoDB, PostgreSQL, Redis, e incluso Local Disk.

  Características:

  - REST APIs auto-generadas
  - Políticas de seguridad reutilizables
  - Backend agnóstico (el framework es independiente de cualquier frontend)
  - ORM (Object Relational Mapping) compatible con integración de base de datos de Express para las request HTTP y de Socket.io para WebSockets

  Cuando utilizar Sails.js
  Debido a su precisión, simplificación de datos y funciones middleware reutilizables, se puede construir aplicaciones de chat customizadas con este framework.

Posee una excelente compatibilidad con Socket.io, haciéndolo muy útil para aplicaciones como juegos y de redes sociales. Sails.js también es muy común para aplicaciones Node.js customizadas a escala empresarial.

Se pueden producir apps listas para producción en cuestión de semanas, además se asemeja al patrón de arquitectura MVC utilizado en Ruby on Rails. Sin embargo, no es muy utilizado para la creación de pequeñas apps. Esto se debe a que Sails tiene algunas limitaciones en su flexibilidad cuando se trata de personalizaciones de alto nivel como con Express.

b. Realizar un cuadro comparativo con sus características principales.

	Express.js	Hapi.js	Socket.io	Sails.js
Performance	Rápido	Media - alta	Media. Se vuelve lenta cuando se escala	Media. Se enfoca en mejorar la eficiencia del desarrollador por sobre la performance
Soporte de la comunidad	Masivo. Posee una comunidad donde se manejan un montón de preguntas y respuestas	Amplio soporte, posee una gran comunidad en github	Soporte amplio	Gran soporte, ya que es un framework bien establecido desde hace bastante tiempo
Facilidad de uso	Depende la dificultad del proyecto, generalmente sencillo de aprender	Fácil. Posee una amplia cantidad de plugins que facilitan su uso	Fácil porque requiere que el programador sepa solo del framework	Fácil porque cuenta con blueprints que hacen más fácil conectar las APIs con un código minimal
Mejor para	Pequeños y grandes proyectos. También es una buena forma de aprender cómo trabajar con Node	Ideal para crear aplicaciones seguras, en tiempo real y escalables	Generalmente usada en aplicaciones en tiempo real o apps que requieran un chat en vivo o sala de conferencias	Proyectos medianos que necesitan implementarse rápido

2) Seleccionar 2 framework y realizar 1 ejemplo con cada uno donde se expongan sus principales diferencias. Crear carpetas para cada uno de ellos (indicar en un readme.md como instalar y correr el proyecto)

Express, Sails