INSTITUTO SUPERIOR TECNOLÓGICO DEL SUR

CARRERA

DISEÑO Y PROGRAMACIÓN WEB

LENGUAJE WEB III

“Frameworks de Javascript”

PROFESOR(A) : Amado Cerpa Juan Andres

ALUMNO : Vilca Apaza Christian

SEMESTRE : V

14/04/2021

Índice

A. ¿Qué es un framework y para qué se utiliza?

Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Vamos, una manera de hacernos más fácil la programación.

B. 4 razones para utilizar un framework a la hora de programar

Evitar escribir código repetitivo
La mayoría de los proyectos tienen partes comunes necesarias para el funcionamiento como, por ejemplo, acceso a base de datos, validación de formularios o seguridad. Un framework nos evita tener que programar estas partes, de esta manera nos resulta más fácil centrarnos en programar la aplicación.

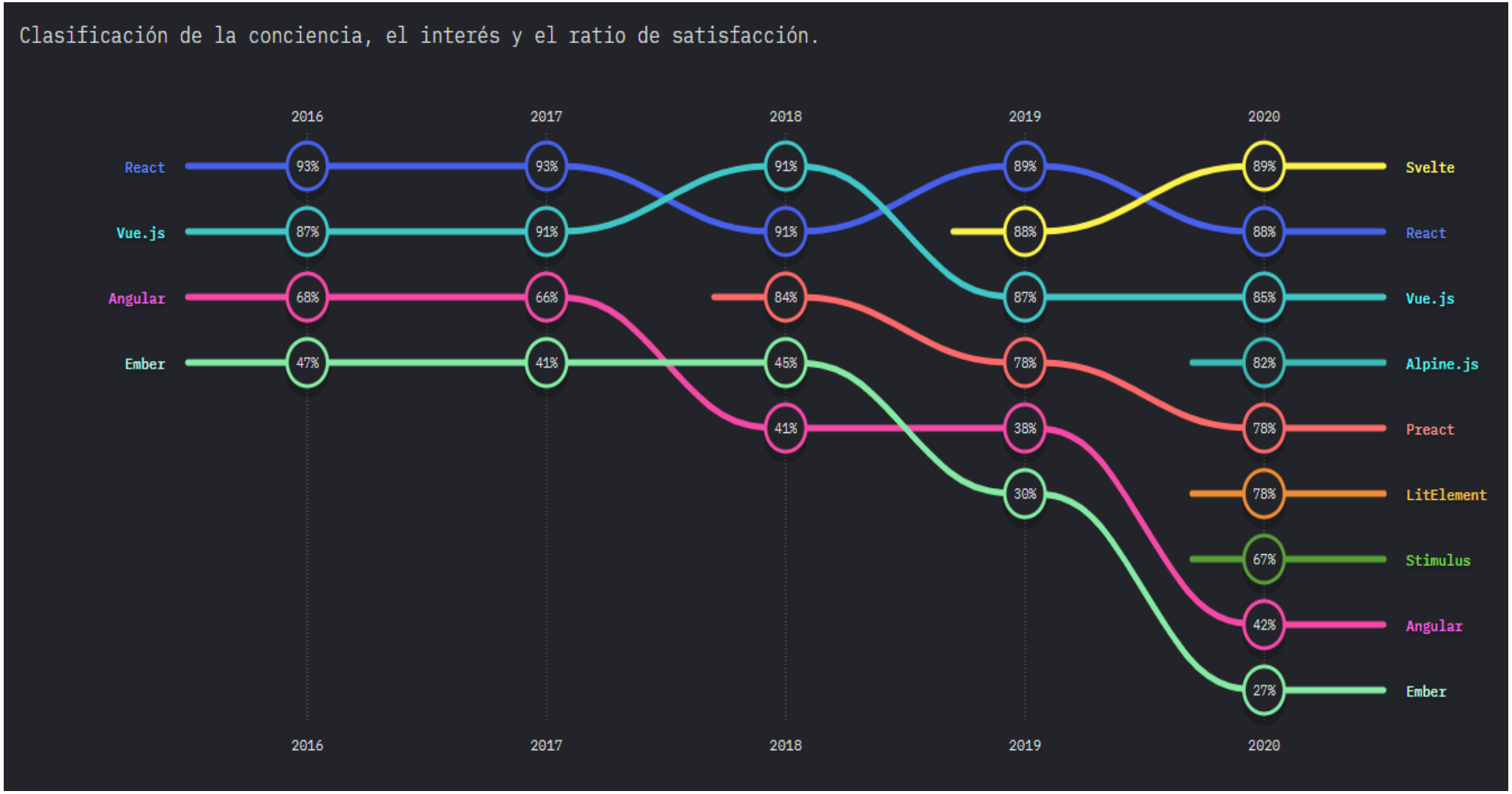
Utilizar buenas prácticas
Los frameworks están basados en patrones de desarrollo, normalmente MVC (Modelo-Vista-Controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario. Vamos, que, gracias a ellos, lo tenemos todo más ordenado.

Permitir hacer cosas avanzadas que tú no harías
Está claro que un framework siempre te va permitir hacer cosas de una manera fácil y segura, que para ti serían imposibles o al menos te costaría mucho tiempo hacerlas.

Desarrollar más rápido
Si tenemos en cuenta los puntos anteriores, sabremos que desarrollar una aplicación con un framework nos permite hacerlo más rápido, más limpio y más seguro.

C. Frameworks Javascript para Fronted

Empezaremos con los 3 Frameworks con más interés o crecimiento en el año 2020



I.Svelte

Svelte es un marco de componentes, como React o Vue, pero con una diferencia importante. Los marcos tradicionales le permiten escribir código declarativo controlado por estado, pero hay una penalización: el navegador debe hacer un trabajo adicional para convertir esas estructuras declarativas en operaciones DOM, utilizando técnicas como la diferenciación de DOM virtual que se comen el presupuesto de su marco y gravan al recolector de basura.
En cambio, Svelte se ejecuta en el momento de la compilación , convirtiendo sus componentes en un código imperativo altamente eficiente que actualiza quirúrgicamente el DOM. Como resultado, puede escribir aplicaciones ambiciosas con excelentes características de rendimiento.
La primera versión de Svelte se trataba de probar una hipótesis : que un compilador especialmente diseñado podría generar un código sólido que brindara una excelente experiencia de usuario. La segunda fue una pequeña actualización que arregló un poco las cosas.
La versión 3 es una revisión significativa. Nuestro enfoque durante los últimos cinco o seis meses ha sido brindar una experiencia de desarrollador sobresaliente. Ahora es posible escribir componentes con mucho menos texto estándar de lo que encontrará en otros lugares. Pruebe el nuevo tutorial y vea lo que queremos decir: si está familiarizado con otros marcos, creemos que se sorprenderá gratamente. Para hacerlo posible, primero necesitábamos repensar el concepto en el corazón de los marcos de interfaz de usuario modernos: reactividad.

- Los componentes svelte se construyen sobre HTML. Solo agregue datos.

- CSS tiene un alcance de componente de forma predeterminada: no más colisiones de estilo o guerras de especificidad. O puede usar su biblioteca CSS-in-JS favorita .

- Activas actualizaciones eficientes y granulares mediante la asignación de variables locales. El compilador hace el resto.

- Cree hermosas interfaces de usuario con un motor de transición potente y eficaz integrado en el marco.

II.React

Es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre. En el proyecto hay más de mil desarrolladores libres.

Declarativo
React te ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.
Las vistas declarativas hacen que tu código sea más predecible, por lo tanto, fácil de depurar.

Basado en componentes
Crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas.
Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM.

- Un componente simple
  Los componentes de React implementan un método llamado render() que recibe datos de entrada y retorna qué mostrar. Este ejemplo utiliza una sintaxis similar a XML llamada JSX. Puedes

acceder a los datos de entrada que se pasan al componente mediante render() a través de this.props. JSX es opcional y no es requerido para usar React.

III.Vue.js

Es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Fue creado por Evan You, y es mantenido por él y por el resto de los miembros activos del equipo central que provienen de diversas empresas como Netlify y Netguru.

Vue.js cuenta con una arquitectura de adaptación gradual que se centra en la representación declarativa y la composición de componentes. La biblioteca central se centra sólo en la capa de vista. Las características avanzadas necesarias para aplicaciones complejas como el enrutamiento, la gestión de estados y las herramientas de construcción se ofrecen a través de librerías y paquetes de apoyo mantenidos oficialmente, con Nuxt.js como una de las soluciones más populares.
Vue.js permite extender el HTML con atributos HTML llamados directivas. Las directivas ofrecen funcionalidad a las aplicaciones HTML, y vienen como directivas incorporadas o definidas por el usuario

- Componentes
  En el núcleo de Vue.js hay un sistema que nos permite renderizar datos de forma declarativa al DOM utilizando una sintaxis de plantilla sencilla.

HTML

```
<div id="app">
  {{ message }}
</div>
```

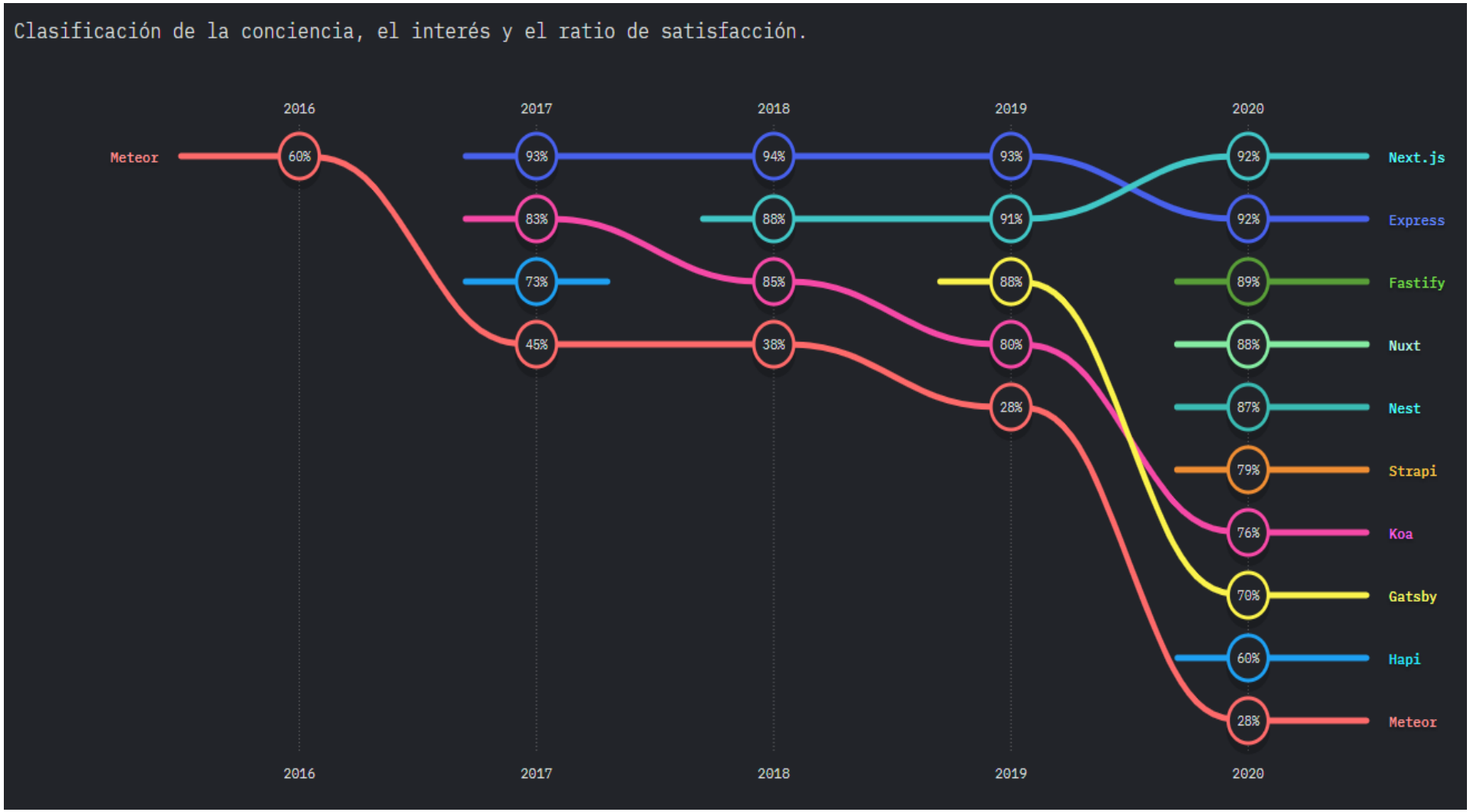
JS

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

¡Hola Vue!

D. Frameworks Javascript para Backend

Empezaremos con los 3 Frameworks con más interés o crecimiento en el año 2020



I.Next.js

Es un pequeño framework construido sobre React.js que viene a ayudar a reducir esta fatiga. ¿Cómo? Next nos permite, instalando una sola dependencia, tener configurado todo lo que necesitamos para crear una aplicación de React usando Babel, Webpack, server render y muchas otras técnicas como HMR o separación de código y… ¡hasta hace más fácil hacer deploy de nuestras aplicaciones!

Next se basa en los 7 principios de las aplicaciones web de Guillermo Rauch (creador de socket.io, Next.js y colaborador de mongoose y muchos otros proyectos).

- Renderizar en el servidor no es opcional
- Actúa inmediatamente a las acciones de los usuarios
- Reacciona inmediatamente a los cambios de datos
- Controla el intercambio de datos con el servidor
- No rompas el historial, mejóralo
- Envía actualizaciones de código
- Predice el comportamiento del usuario

Inspirado por la facilidad de uso de PHP, y beneficiándose del sistema de módulos de JavaScript, haciendo que cada archivo exporte un componente de nuestra aplicación que puede probarse individualmente y pudiendo descargar miles más desde npm.

Cero configuraciones
Next.js logra proveer de un entorno de cero configuraciones asumiendo unas pocas cosas sobre la estructura de tu aplicación en el sistema de archivos, una de estas es que necesitamos crear una carpeta pages en la raíz de nuestro proyecto y cada archivo dentro de esa carpeta va a ser una ruta, así index.js es /, mientras que platzi.js sería /platzi.
Gracias a esto Next.js nos provee de un sistema de rutas, separación de código por ruta, actualizaciones de código sin necesidad de recargar y no tener que configurar nuestros entry points de WebPack ya que Next.js ya sabe donde buscar nuestros archivos.

Solo JavaScript, todo es una función
Cada ruta de nuestra aplicación no es más que un archivo de JavaScript que exporta un componente de React, el cual puede ser tan simple como una función y puede a su vez importar más componentes hechos sencillamente con funciones para que nuestra aplicación no sea otra cosa más que muchas funciones compuestas.
Incluso cuando se trata de estilizar nuestra aplicación Next.js nos provee de un sistema de CSS en JS llamado styled-jsx (creado especialmente para Next.js) que nos permite escribir nuestros estilos directo en JS haciendo uso de todo el poder de CSS.

¿Lo mejor de styled-jsx? Sólo genera el CSS necesario para los componentes renderizados (tanto en el servidor como en el cliente) y una vez un componente se deja de usar se quita su CSS, nunca vamos a tener más CSS del necesario y todo este va a estar hecho específicamente para nuestros componentes, es imposible que se pisen estilos.

```
import React from'react';
import Link from'next/link';

functionHomePage() {
  return (
    <main>
      <h1>Página principalh1>
      <Linkhref="/about">
        <a>Ir a <em>/aboutem>a>
      Link>
      <stylejsx>{`
        h1 { color: red; }
      `}style>
    main>
  );
}

exportdefault HomePage;
```

II. Express y Node.js

¿Qué son Express y Node?

Node (o más correctamente: Node.js) es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. La ejecución en tiempo real está pensada para usarse fuera del contexto de un explorador web (es decir, ejecutarse directamente en una computadora o sistema operativo de servidor). Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros.
Desde una perspectiva de desarrollo de servidor web, Node tiene un gran número de ventajas:
¡Gran rendimiento! Node ha sido diseñado para optimizar el rendimiento y la escalabilidad en aplicaciones web y es un muy buen complemento para muchos problemas comunes de desarrollo web (ej, aplicaciones web en tiempo real).

El código está escrito en "simple JavaScript", lo que significa que se pierde menos tiempo ocupándose de las "conmutaciones de contexto" entre lenguajes cuando estás escribiendo tanto el código del explorador web como del servidor.

JavaScript es un lenguaje de programación relativamente nuevo y se beneficia de los avances en diseño de lenguajes cuando se compara con otros lenguajes de servidor web tradicionales (ej, Python, PHP, etc.) Muchos otros lenguajes nuevos y populares se compilan/convierten a JavaScript de manera que puedes también usar CoffeeScript, ClosureScript, Scala, LiveScript, etc.

El gestor de paquetes de Node (NPM del inglés: Node Packet Manager) proporciona acceso a cientos o miles de paquetes reutilizables. Tiene además la mejor en su clase resolución de dependencias y puede usarse para automatizar la mayor parte de la cadena de herramientas de compilación.

Es portable, con versiones que funcionan en Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, y NonStop OS. Además, está bien soportado por muchos de los proveedores de hospedaje web, que proporcionan infraestructura específica y documentación para hospedaje de sitios Node.

Tiene un ecosistema y comunidad de desarrolladores de terceros muy activa, con cantidad de gente deseosa de ayudar.

Puedes crear de forma sencilla un servidor web básico para responder cualquier petición simplemente usando el paquete HTTP de Node, como se muestra abajo. Este, creará un servidor y escuchará cualquier clase de peticiones en la URL http://127.0.0.1:8000/; cuando se reciba una petición, se responderá enviando en texto la respuesta: "Hola Mundo!".

```
// Se carga el módulo de HTTP
var http = require("http");

// Creación del servidor HTTP, y se define la escucha
// de peticiones en el puerto 8000
http.createServer(function(request, response) {

   // Se define la cabecera HTTP, con el estado HTTP (OK: 200) y el tipo de contenido
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Se responde, en el cuerpo de la respuesta con el mensaje "Hello World"
   response.end('Hola Mundo!\n');
}).listen(8000);

// Se escribe la URL para el acceso al servidor
console.log('Servidor en la url http://127.0.0.1:8000/');
```

Express

Es el framework web más popular de Node, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.
A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más. Puedes encontrar una lista de paquetes middleware mantenida por el equipo de Express en Express Middleware (junto con una lista de algunos de los paquetes más populares de terceros).

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hola Mundo!');
});

app.listen(3000, function() {
  console.log('Aplicación ejemplo, escuchando el puerto 3000!');
});
```

III. Fastify

Es un Framwork Node.js de código abierto que se mantiene enfocado en brindar una excelente experiencia de desarrollador, una sobrecarga de rendimiento mínima y una arquitectura de complementos flexible.
Los esfuerzos de desarrollo de Fastify se centran en seis características y principios principales. En primer lugar, como sugiere su nombre, Fastify se centra en la velocidad y actualmente puede atender hasta 30.000 solicitudes por segundo.
Fastify proporciona extensibilidad a través de ganchos, complementos y decoradores.
El proyecto Fastify recomienda encarecidamente la opción de usar JSON Schema para validar rutas y serializar salidas para mejorar el rendimiento y la precisión.
Fastify aprovecha Pino , un registrador de Node.js con una sobrecarga muy baja, para el registro.
Fastify se esfuerza por ser amigable para los desarrolladores, con un enfoque en el código expresivo sin sacrificar el rendimiento y la seguridad.

Si bien no es un marco de TypeScript, Fastify está preparado para TypeScript y es totalmente compatible con las definiciones de tipos de TypeScript.

Funciones principales

Estas son las principales características y principios sobre los que se ha construido fastify:

Alto rendimiento: hasta donde sabemos, Fastify es uno de los frameworks web más rápidos de la ciudad, dependiendo de la complejidad del código, podemos atender hasta 30 mil solicitudes por segundo.

Extensible: Fastify es completamente extensible a través de sus ganchos, complementos y decoradores.
Basado en esquema: incluso si no es obligatorio, recomendamos usar JSON Schema para validar sus rutas y serializar sus salidas, Fastify compila internamente el esquema en una función de alto rendimiento.
Registro: los registros son extremadamente importantes pero costosos; ¡Elegimos el mejor registrador para casi eliminar este costo,Pino!
Apto para desarrolladores: el marco está diseñado para ser muy expresivo y ayudar a los desarrolladores en su uso diario, sin sacrificar el rendimiento y la seguridad.
Listo para TypeScript: trabajamos duro para mantener un archivo de declaración de tipo TypeScript para poder apoyar a la creciente comunidad de TypeScript.

```
// Require the framework and instantiate it
const fastify = require('fastify')({ logger: true })

// Declare a route
fastify.get('/', async (request, reply) => {
  return { hello: 'world' }
})

// Run the server!
const start = async () => {
  try {
    await fastify.listen(3000)
  } catch (err) {
    fastify.log.error(err)
    process.exit(1)
  }
}
start()
```

Conclusión

En conclusión, podemos encontrar un sinfín de frameworks de javascript sea para backend o fronted hay para elegir y siguen creciendo, ahora en la actualidad también hay algo llamado deno que seria el remplazo para node.js en un futuro ya que fue creado por el mismo creador de node.js.

Bibliografía

https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza#:~:text=Un%20framework%2C%20seg%C3%BAn%20wikipedia%2C%20es,hacernos%20m%C3%A1s%20f%C3%A1cil%20la%20programaci%C3%B3n.

https://openwebinars.net/blog/los-5-frameworks-de-javascript-para-frontend-mas-usados-en-2018/

https://svelte.dev/blog/svelte-3-rethinking-reactivity

https://svelte.dev/

https://es.reactjs.org/docs/getting-started.html

https://es.reactjs.org/

https://es.wikipedia.org/wiki/Vue.js

https://vuejs.org/v2/guide/

https://platzi.com/blog/nextjs-el-futuro-de-las-aplicaciones-con-react/

https://nextjs.org/docs/basic-features/pages

https://en.wikipedia.org/wiki/Next.js

https://www.infoq.com/news/2020/03/fasify-nodejs-framework/

https://www.fastify.io/

https://2020.stateofjs.com/es-ES/