Backend con node.js

Reactivos

Pregunta 1

¿Qué es Node.js?, ¿cómo ha revolucionado el desarrollo de *back-end*?

@ Revisa la explicación del tema 1.

Opciones:

a. Es una biblioteca de JavaScript que revolucionó el front-end.
b. Es un protocolo de red para comunicación que revolucionó el back-end.
c. Es una base de datos relacional.
d. Es un entorno de ejecución que permitió a los desarrolladores usar el lenguaje JavaScript para construir servidores eficientes y escalables.

Pregunta 2

¿Qué es un protocolo de red? Incluye un ejemplo común.

@ Revisa la explicación del tema 1.

Opciones:

a. Es un conjunto de reglas que rigen la comunicación entre dispositivos, por ejemplo: TCP/IP.
b. Es un software que controla la red, por ejemplo: Windows Server.
c. Es un hardware de red, por ejemplo: Router.
d. Es un lenguaje de programación, por ejemplo: Python.

Pregunta 3

En Node.js., ¿qué permite el objeto "request" (req)?

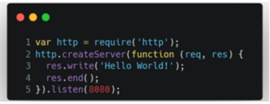
@ Revisa la explicación del tema 1.

Opciones:

a. Permite la edición de imágenes en tiempo real.
b. Permite la construcción de aplicaciones en tiempo real y basadas en datos.
c. Permite la integración con sistemas operativos específicos.
d. Permite la traducción automática de idiomas.

Pregunta 4

Si estás creando un servidor HTTP en Node.js que escucha en el puerto 8080, ¿cuál sería el código correcto?



```
var http = require('http');
http.createServer(function (req, res) {
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

@ Revisa la explicación del tema 1

Opciones:

a. http.createServer(8080);
b. http.listenServer(8080);
c. http.createServer().listen(8080);
d. http.createServer().port(8080);

Pregunta 5

Estás trabajando en una aplicación Node.js y quieres manejar mensajes extragrandes que se pueden codificar en fragmentos, ¿qué característica de Node.js utilizarías?

@ Revisa la explicación del tema 1.

Opciones:

a. El objeto "request (req)"
b. El módulo HTTP incorporado.
c. La arquitectura orientada a la entrada y salida sin bloqueo.
d. La interfaz HTTP en Node.js

Pregunta 6

¿Qué permite el Event Loop de Node.js en relación con las operaciones de E/S?

@ Revisa la explicación del tema 2.

Opciones:

a. No permite operaciones de E/S.
b. Permite operaciones de E/S sin bloqueo.
c. Permite operaciones de E/S con bloqueo.
d. Bloquea todas las operaciones de E/S.

Pregunta 7

En programación, ¿a qué se refiere el término "asincrónica"?

@ Revisa la explicación del tema 2.

Opciones:

a. A la ejecución de tareas fuera de secuencia.
b. A la ejecución de tareas en un orden estricto.
c. A la incapacidad de ejecutar múltiples tareas.
d. A la ejecución de tareas en un solo hilo.

Pregunta 8

En Node.js, ¿qué función realiza la biblioteca libuv?

@ Revisa la explicación del tema 2

Opciones:

a. Proporciona una interfaz para operaciones de E/S con bloqueo.
b. Detiene el bucle de eventos.
c. Maneja el bucle de eventos y proporciona una interfaz para operaciones de E/S sin bloqueo.
d. Crea múltiples hilos de ejecución.

Pregunta 9

En Node.js., ¿cómo puedes utilizar la asincronía para leer un archivo?

@ Revisa la explicación del tema 2.

Opciones:

a. Utilizando la función readFile del módulo fs de manera asincrónica, pasando una función de devolución de llamada.
b. Utilizando la función readFile del módulo fs de manera síncrona.
c. Bloqueando el Event Loop hasta que se lea el archivo.
d. Utilizando múltiples hilos de ejecución para leer el archivo.

Pregunta 10

Estás desarrollando una aplicación en Node.js y quieres manejar múltiples solicitudes de forma simultánea, ¿qué componente de Node.js te permite hacer esto?

@ Revisa la explicación del tema 2

Opciones:

a. El módulo fs.
b. La pila de llamadas de JavaScript.
c. La cola de tareas de JavaScript.
d. La biblioteca libuv

Pregunta 11

¿Qué es un módulo en Node.js?

@ Revisa la explicación del tema 3

Opciones:

a. Una colección de objetos y funciones de JavaScript que no pueden ser utilizados por aplicaciones externas.
b. Una forma de limitar la reutilización del código.
c. Una colección de objetos y funciones de JavaScript que pueden ser utilizados por aplicaciones externas.
d. Un método para hacer que el código sea más difícil de mantener.

Pregunta 12

Elige la opción que indique una buena práctica para manejar rutas en aplicaciones web.

@ Revisa la explicación del tema 3

Opciones:

a. Usar verbos en lugar de sustantivos.
b. Usar sustantivos en plural.
c. Alterar los datos con el método HTTP GET.
d. Evitar el uso de subrutas o subrecursos.

Pregunta 13

En Express.js., ¿qué método se utiliza para cargar una función de middleware?

@ Revisa la explicación del tema 3.

Opciones:

a. app.load()
b. app.middleware()
c. app.next()
d. app.use()

Pregunta 14

En Node.js estás creando un módulo llamado "saludo.js" con una función para saludar a un usuario, ¿cuál sería el código correcto para exportar la función?

@ Revisa la explicación del tema 3

Opciones:

a. import.saludarUsuario = function(nombre) { return '¡Hola, ' + nombre + '!'; };
b. module.saludarUsuario = function(nombre) { return '¡Hola, ' + nombre + '!'; };
c. exports.saludarUsuario = function(nombre) { return '¡Hola, ' + nombre + '!'; };
d. require.saludarUsuario = function(nombre) { return '¡Hola, ' + nombre + '!'; };

Pregunta 15

Si estás utilizando Express.js y quieres manejar todas las solicitudes HTTP en una ruta específica, ¿cuál sería el método correcto?

@ Revisa la explicación del tema 3.

Opciones:

a. app.get()
b. app.all()
c. app.post()
d. app.route()

Pregunta 16

Al trabajar con Node.js., ¿cuál es una de las mejores prácticas de seguridad recomendadas?

@ Revisa la explicación del tema 4.

Opciones:

a. Mantener el entorno actualizado, incluyendo Node.js y sus paquetes.
b. Utilizar siempre paquetes de terceros sin verificar su origen.
c. Ignorar las advertencias de vulnerabilidad en GitHub.
d. Evitar el uso de encabezados de seguridad HTTP.

Pregunta 17

¿Qué permite el uso de Tokens JSON Web (JWT) en la autorización de usuarios en Node.js?

@ Revisa la explicación del tema 4

Opciones:

a. Al convertir una contraseña en una cadena de caracteres de longitud fija.
b. Al restringir los ataques de inyección SQL.
c. Al acceder a recursos protegidos mediante la verificación del token por parte del servidor.
d. Al actualizar automáticamente todos los paquetes desactualizados.

Pregunta 18

En la autenticación y autorización en Node.js, ¿qué es el *salting*?

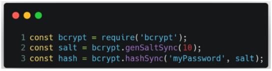
@ Revisa la explicación del tema 4

Opciones:

a. Un método para almacenar contraseñas en texto claro.
b. Un proceso de agregar una cadena de caracteres aleatoria a la contraseña antes de hacer el hashing.
c. Una técnica para permitir el acceso no autorizado a una aplicación.
d. Un token utilizado para acceder a recursos protegidos.

Pregunta 19

En términos de autenticación, ¿qué hace el siguiente código en una aplicación Node.js?



```
const bcrypt = require('bcrypt');
const salt = bcrypt.genSaltSync(10);
const hash = bcrypt.hashSync('myPassword', salt);
```

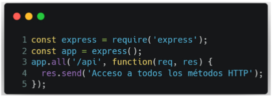
@ Revisa la explicación del tema 4.

Opciones:

a. Convierte una contraseña en una cadena de caracteres de longitud fija mediante hashing y salting.
b. Genera un token JWT para autorizar a los usuarios.
c. Verifica las dependencias vulnerables en la aplicación.
d. Configura la monitorización y el registro de la aplicación.

Pregunta 20

Dado el siguiente código en Node.js, ¿qué hace este fragmento en términos de seguridad?



```
const express = require('express');
const app = express();
app.all('/api', function(req, res) {
  res.send('Acceso a todos los métodos HTTP');
});
```

@ Revisa la explicación del tema 4

Opciones:

a. Restringe los ataques de inyección SQL en la aplicación.
b. Autentica a los usuarios mediante el uso de contraseñas y tokens.
c. Agrega encabezados de seguridad HTTP para proteger la aplicación.
d. Permite manejar todas las solicitudes HTTP en una ruta específica.

Pregunta 21

¿Qué tipo de bases de datos organiza los datos en tablas divididas en filas y columnas?
@ Revisa la explicación del tema 5

Opciones:

a. MySQL
b. MongoDB
c. Redis
d. LevelDB

Pregunta 22

¿Cuál es el propósito principal de las bases de datos en el mundo de la información?
@ Revisa la explicación del tema 5.

Opciones:

a. Facilitar la navegación en la web.
b. Mejorar la velocidad de las transacciones en línea.
c. Almacenar la huella digital que una persona deja en cada sitio, transacción o paso por la web.
d. Mejorar la interfaz gráfica de los sitios web.

Pregunta 23

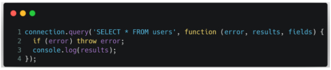
En un lenguaje de programación, ¿qué técnica se utiliza para interactuar con bases de datos basadas en documentos?
@ Revisa la explicación del tema 5

Opciones:

a. ORM (Object-Relational Mapping)
b. ODM (Object-Document Mapping)
c. SQL (Structured Query Language)
d. JIT (Just-in-time)

Pregunta 24

En Node.js y en términos de manejo de datos con una base de datos MySQL, ¿qué hace el siguiente código?



```
connection.query('SELECT * FROM users', function (error, results, fields) {
  if (error) throw error;
  console.log(results);
});
```

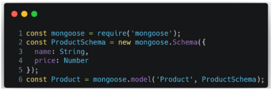
@ Revisa la explicación del tema 5.

Opciones:

a. Inserta nuevos registros en la tabla de usuarios.
b. Actualiza los registros existentes en la tabla de usuarios.
c. Elimina todos los registros de la tabla de usuarios.
d. Realiza una consulta SQL para recuperar todos los registros de la tabla de usuarios.

Pregunta 25

Dado el siguiente código en Node.js y en términos de definición de un modelo con Mongoose para MongoDB, ¿qué hace este fragmento?



```
const mongoose = require('mongoose');
const ProductSchema = new mongoose.Schema({
  name: String,
  price: Number
});
const Product = mongoose.model('Product', ProductSchema);
```

@ Revisa la explicación del tema 5.

Opciones:

a. Conecta a una base de datos MongoDB utilizando las credenciales proporcionadas.
b. Define un modelo Product que se mapeará a una colección en MongoDB.
c. Realiza una consulta SQL para recuperar todos los registros de una tabla.
d. Inserta nuevos registros en la tabla de productos en una base de datos MySQL.

Pregunta 26

En el contexto del despliegue de aplicaciones Node.js., ¿qué permite hacer la Dockerización?
@ Revisa la explicación del tema 6

Opciones:

a. Separar y gestionar componentes front-end y back-end de manera independiente.
b. Administrar y escalar contenedores utilizando herramientas como AWS Lambda.
c. Proporcionar una plataforma para desarrollar y ejecutar aplicaciones sin preocuparse por la infraestructura subyacente.
d. Empaquetar aplicaciones en contenedores autocontenidos que se pueden desplegar fácilmente en cualquier plataforma.

Pregunta 27

A nivel de usuario, ¿qué tipo de pruebas busca simular la experiencia en los diferentes productos de un software, incluyendo acciones como inicio de sesión, depósitos o configuración de cuenta?
@ Revisa la explicación del tema 6

Opciones:

a. Pruebas de rutas con Express.
b. Pruebas unitarias.
c. Pruebas integrales.
d. Pruebas de integración.

Pregunta 28

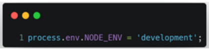
¿Cuál de las siguientes opciones es una ventaja al utilizar Jest como marco de trabajo para pruebas en Node.js?
@ Revisa la explicación del tema 6

Opciones:

a. Es compatible con TypeScript.
b. Es compatible solo con proyectos basados en Vue.js.
c. No permite pruebas instantáneas.
d. No es adecuado para aplicaciones React.

Pregunta 29

En Node.js y en términos de configuración de entornos, ¿qué hace el siguiente código?



```
process.env.NODE_ENV = 'development';
```

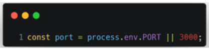
@ Revisa la explicación del tema 6

Opciones:

a. Define una variable de entorno para especificar el puerto del servidor.
b. Define una variable de entorno para especificar las credenciales de la base de datos.
c. Define una variable de entorno para especificar la ubicación de los archivos y carpetas estáticas.
d. Define una variable de entorno para especificar el entorno de desarrollo.

Pregunta 30

Dado el siguiente código en Node.js y en términos de configuración de entornos, ¿qué hace este fragmento?



```
const port = process.env.PORT || 3000;
```

@ Revisa la explicación del tema 6.

Opciones:

a. Establece el puerto en el que la aplicación debe ejecutarse, utilizando una variable de entorno o el puerto 3000 como valor predeterminado.
b. Define una variable de entorno para la dirección de la base de datos.
c. Configura una variable de entorno para la ubicación de archivos y carpetas estáticos.
d. Define una variable de entorno para las credenciales de API externas.

Pregunta 31

¿Qué ventaja ofrecen los microservicios en comparación con las aplicaciones monolíticas?

@ Revisa la explicación del tema 7.

Opciones:

a. Requieren un esfuerzo adicional para responder a los cambios en el sistema.
b. Un fallo en un componente que afecta a todo el sistema.
c. Permiten despliegues más rápidos y frecuentes.
d. Están estrechamente integrados en una única unidad.

Pregunta 32

Según el documento, ¿cuál de los siguientes pasos es parte del proceso para usar Docker con Node.js?

@ Revisa la explicación del tema 7

Opciones:

a. Publicar en Docker Hub como paso obligatorio.
b. Construir la imagen Docker.
c. Utilizar Mocha para escribir pruebas unitarias.
d. Desarrollar la aplicación utilizando Feathers.

Pregunta 33

¿Qué permite hacer la contenerización en el despliegue de microservicios?

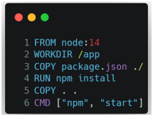
@ Revisa la explicación del tema 7.

Opciones:

a. Escalar toda la aplicación en lugar de servicios individuales.
b. Encapsular los microservicios en contenedores, incluyendo el código, las bibliotecas y las dependencias.
c. Utilizar aplicaciones monolíticas en lugar de microservicios.
d. Publicar imágenes en Docker Hub sin una cuenta.

Pregunta 34

Dado el siguiente código para un Dockerfile en Node.js, ¿qué hace este fragmento?



```
FROM node:14
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
CMD ["npm", "start"]
```

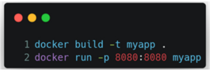
@ Revisa la explicación del tema 7.

Opciones:

a. Instala las dependencias de una aplicación Node.js y define cómo iniciarla en un contenedor Docker.
b. Construye una imagen Docker para una aplicación PHP.
c. Publica la imagen en Docker Hub.
d. Escala la aplicación en diferentes sistemas.

Pregunta 35

Dado el siguiente código en Node.js y en términos de despliegue de aplicaciones en un contenedor Docker, ¿qué hace este fragmento?



```
docker build -t myapp .
docker run -p 8080:8080 myapp
```

@ Revisa la explicación del tema 7.

Opciones:

a. Publica la imagen en Docker Hub.
b. Escala la aplicación en diferentes sistemas.
c. Instala las dependencias de una aplicación PHP.
d. Construye una imagen Docker y ejecuta la aplicación en un contenedor en el puerto 8080.

Pregunta 36

¿Cuál de las siguientes afirmaciones describe correctamente la importancia de la optimización de desempeño en el desarrollo de aplicaciones?

@ Revisa la explicación del tema 8

Opciones:

a. La optimización de desempeño solo mejora la velocidad de una aplicación.
b. La optimización de desempeño reduce la capacidad de respuesta de una aplicación.
c. La optimización de desempeño solo es relevante en aplicaciones pequeñas.
d. La optimización de desempeño mejora la velocidad y la eficiencia, además, contribuye a una mejor experiencia del usuario.

Pregunta 37

En el contexto de Node.js, ¿cuál de las siguientes estrategias **no** es una técnica común de optimización?

@ Revisa la explicación del tema 8.

Opciones:

a. Uso de funciones sincrónicas.
b. Gestión de la memoria.
c. Publicación de la aplicación en múltiples plataformas.
d. Optimización de consultas a la base de datos.

Pregunta 38

¿Qué herramienta de *debugging* permite depurar aplicaciones Node.js en el navegador Chrome?

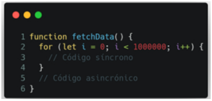
@ Revisa la explicación del tema 8

Opciones:

a. Node.js Debugger.
b. Chrome DevTools.
c. WebStorm Debugger.
d. IronNode.

Pregunta 39

Dado el siguiente código en Node.js, ¿cuál es la mejor práctica para optimizar el rendimiento?



```
function fetchData() {
  for (let i = 0; i < 1000000; i++) {
    // Código síncrono
  }
  // Código asincrónico
}
```

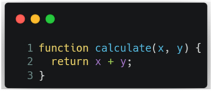
@ Revisa la explicación del tema 8.

Opciones:

a. Convertir el bucle en una función asincrónica.
b. Aumentar el número de iteraciones en el bucle.
c. Agregar el código síncrono.
d. Ninguna de las anteriores.

Pregunta 40

Dado el siguiente código en Node.js, ¿cuál sería la mejor manera de depurar y encontrar errores?



```
function calculate(x, y) {
  return x + y;
}
```

@ Revisa la explicación del tema 8.

Opciones:

a. Utilizar herramientas de depuración como Node.js Debugger o Chrome DevTools.
b. Utilizar pruebas unitarias y de integración.
c. Implementar monitoreo y registro.
d. Revisar el código con el equipo de marketing.