HINTS

PROCESAMIENTO DE LA RESPUESTA

Tipos de Errores:

Es útil distinguir entre los diferentes tipos de errores que pueden ocurrir en una aplicación Node.js. En general, los errores se pueden dividir en dos categorías principales: errores del programador y problemas operativos. Los argumentos malos o incorrectos para una función son un ejemplo del primer tipo de problema, mientras que las fallas transitorias cuando se trata de API externas están firmemente en la segunda categoría.

1. Errores operativos

Los errores operativos son en su mayoría errores esperados que pueden ocurrir en el curso de la ejecución de la aplicación. No son necesariamente errores, sino circunstancias externas que pueden interrumpir el flujo de ejecución del programa. En tales casos, el impacto completo del error puede entenderse y manejarse adecuadamente. Algunos ejemplos de errores operativos en Node.js incluyen los siguientes:

- Una solicitud de API falla por alguna razón (por ejemplo, el servidor está inactivo o se excedió el límite de velocidad).
- Se pierde una conexión a la base de datos, quizás debido a una conexión de red defectuosa.
- El sistema operativo no puede cumplir con su solicitud para abrir un archivo o escribir en él.
- El usuario envía datos no válidos al servidor, como un número de teléfono o una dirección de correo electrónico no válidos.

Estas situaciones no surgen por errores en el código de la aplicación, sino que deben manejarse correctamente. De lo contrario, podrían causar problemas más serios.

2. Errores del programador

Los errores del programador son errores en la lógica o sintaxis del programa que solo se pueden corregir cambiando el código fuente. Este tipo de errores no se pueden manejar porque, por

definición, son errores en el programa. Algunos ejemplos de errores del programador incluyen:

- Errores de sintaxis, como no cerrar una llave.
- Escriba errores cuando intente hacer algo ilegal, como realizar operaciones en operandos de tipos no coincidentes.
- Parámetros incorrectos al llamar a una función.
- Errores de referencia cuando escribe mal el nombre de una variable, función o propiedad.
- Intentar acceder a una ubicación más allá del final de una matriz.
- No poder manejar un error operativo.

MANEJO DE ERRORES OPERATIVOS

Los errores operativos son en su mayoría predecibles, por lo que deben anticiparse y tenerse en cuenta durante el proceso de desarrollo. Esencialmente, el manejo de este tipo de errores implica considerar si una operación pudiese fallar, por qué podría fallar y qué debería suceder si falla. Consideremos algunas estrategias para manejar errores operativos en Node.js.

1. Informe el error en la pila

En muchos casos, la acción apropiada es detener el flujo de ejecución del programa, limpiar cualquier proceso inacabado e informar el error a la pila para que pueda manejarse adecuadamente. A menudo, esta es la forma correcta de abordar el error cuando la función donde ocurrió está más abajo en la pila, de modo que no tiene suficiente información para manejar el error directamente.

2. Vuelva a intentar la operación

Las solicitudes de red a servicios externos a veces pueden fallar, incluso si la solicitud es completamente válida. Esto puede deberse a una falla transitoria, que puede ocurrir si hay una falla en la red o una sobrecarga del servidor. Dichos problemas suelen ser efímeros, por lo que en lugar de informar el error de inmediato, puede volver a intentar la solicitud varias veces hasta que tenga éxito o hasta que se alcance la cantidad máxima de reintentos. La primera consideración es determinar si es apropiado volver a intentar la solicitud. Por ejemplo, si el código de estado HTTP de la respuesta inicial es 500, 503 o 429, puede ser ventajoso volver a intentar la solicitud después de un breve retraso.

Puede verificar si el encabezado HTTP Retry-After está presente en la respuesta. Este encabezado indica la cantidad exacta de tiempo que debe esperar antes de realizar una solicitud de seguimiento. Si el encabezado Reintentar después no existe, debe retrasar la solicitud de

seguimiento y aumentar progresivamente la demora para cada reintento consecutivo. Esto se conoce como la estrategia de retroceso exponencial. También debe decidir el intervalo de demora máximo y cuántas veces reintentar la solicitud antes de darse por vencido. En ese momento, debe informar a la persona que llama que el servicio de destino no está disponible.

3. Enviar el error al cliente

Cuando se trata de entradas externas de los usuarios, se debe suponer que la entrada es mala por defecto. Por lo tanto, lo primero que debe hacer antes de iniciar cualquier proceso es validar la entrada y reportar cualquier error al usuario de inmediato para que pueda ser corregido y reenviado. Al entregar errores del cliente, asegúrese de incluir toda la información que el cliente necesita para construir un mensaje de error que tenga sentido para el usuario.

4. Cancelar el programa

En el caso de errores irrecuperables del sistema, el único curso de acción razonable es registrar el error y terminar el programa inmediatamente. Es posible que ni siquiera pueda cerrar el servidor correctamente si la excepción no se puede recuperar en la capa de JavaScript. En ese momento, es posible que se requiera que un administrador de sistemas investigue el problema y lo solucione antes de que el programa pueda volver a iniciarse.

PREVENCIÓN DE ERRORES DEL PROGRAMADOR

Debido a su naturaleza, los errores del programador no se pueden manejar; son errores en el programa que surgen debido a un código o lógica rotos, que posteriormente deben corregirse. Sin embargo, hay algunas cosas que puede hacer para reducir en gran medida la frecuencia con la que ocurren en su aplicación.

La clase Error (o una subclase) siempre debe usarse para comunicar errores en su código. Técnicamente, puede lanzar cualquier cosa en JavaScript, no solo objetos de error, pero esto no se recomienda ya que reduce en gran medida la utilidad del error y hace que el manejo de errores sea propenso a errores. Mediante el uso constante de objetos de error, puede esperar acceder de manera confiable a error.message o error.stack en lugares donde se manejan o registran los errores. Incluso puede aumentar la clase de error con otras propiedades útiles relevantes para el contexto en el que ocurrió el error.

Los errores operativos son inevitables y deben tenerse en cuenta en cualquier programa correcto. La mayoría de las veces, se debe emplear una estrategia de error recuperable para que el programa pueda continuar funcionando sin problemas. Sin embargo, si el error es lo suficientemente grave, podría ser apropiado finalizar el programa y reiniciarlo. Intente cerrar

correctamente si surgen tales situaciones para que el programa pueda iniciarse nuevamente en un estado limpio.

Los errores del programador no se pueden manejar ni recuperar, pero se pueden mitigar con un conjunto de pruebas automatizado y herramientas de tipeo estático. Al escribir una función, defina el comportamiento de los parámetros incorrectos y actúe apropiadamente una vez detectados. Permita que el programa se bloquee si se detecta una excepción no detectada o un rechazo no controlado.

Use un servicio de monitoreo de errores, como Honeybadger, para capturar y analizar sus errores. Esto puede ayudarlo a mejorar drásticamente la velocidad de depuración y resolución.