Arquitectura DE Software- Practico

Arquitectura de software (Universidad ORT Uruguay)

ARQUITECTURA DE SOFTWARE-PRÁCTICO

¿Se puede usar typescript? Si

Antiguamente teníamos el triángulo HTML-CSS-JS, HTML tenía la estructura, CSS los etilos y JS las reacciones etc.

Hoy en día también se comienza a utilizar JS del lado del servidor.

If(x!=null) distinto de if(x!=undefined)

Undefined es distinto de null

En el if usar if(x) es mejor, si es null o undefined da false directamente. También sirve para NaN.

=== compara tipo y después valor

CLASE 2

JavaScript es un lenguaje, pero el motor puede variar, ósea donde se está ejecutando. Node js es uno de estos.

Ejemplo 1

Event loop es lo que usa para hacer asincronía en un solo hilo.

Maneja dos colas.

Call Stack es First In Last Out

EVENTLOOP = CALL STACK + CALLBACK QUEUE + APIs (en nuestro caso Node APIs)

Código Bloqueante:

Ojo con el procesamiento de imágenes, llamadas a BD que devuelven muchos datos y debo hacer operaciones sobre ellos. Nuestro código en general no lo es.

MÓDULO

En node puede ser una simple función. Básicamente es un archivo donde pongo código js y puede ejecutarse. Entones, la manera en que podemos usar eso módulos es con require.

package-lock.json genera el árbol de dependencias y lo maneja, qué versión maneja de cada una etc. Siempre quedarse con el entrante, nunca el nuestro.

La idea es usar express.js

CLASE 3 – 28/3

RESUMEN CLASE ANTERIOR

Event Loop:

Estamos acostumbrado a código que leemos de abajo hacia arriba o viceversa y listo.

Ej: var hola;

hola =1;

console.log(“hola”)

Puedo decir claramente en que orden se va a ejecutar esto.

Pero en node tengo un event loop siempre corriendo, en donde tengo un stack del “procesador”, que inicialmente esta vacío, que siempre espera cosas pendientes para ejecutar.

```
var hola;

hola=1;

var chau = 2;

console.log(hola);
```

```
function saludo (nombre){
    console.log(nombre)
}
saludo("Juan);
setTimeout(saludo("maria),1000);
console.log(chau);
```

En este caso el orden de ejecución sería linealmente hasta saludo("juan") y luego de eso el stack está libre

En el timeout agendo un pedazo de código para que se ejecute dentro de un tiempo x

Entonces, el timeout registra saludo para que se ejecute despues.

Se va a impirmir el chau y despues de los 1000 se imprime maria

En el stack vamos a tener que saludo quedó alli como pendiente.

EJEMPLO 1:

Quiero hacer un convertidor entre cases ej:

camelCase, slug-case, snake_case

Cómo lo implementaría?

- Interfaz convertidor y clases que lo implementan para cada conversión.
- Función con entrada y salida.

node converter.js para ejecutar

Si tengo un módulo que depende de otro lo mejor que puede pasar es que éste tenga una interfaz lo más sencilla y chica posible. Desde el punto de vista de modificabilidad del que llama el ejemplo es muy bueno, pero del que tiene que convertir es el peor caso.

VENTAJAS DE PIPE&FILTER (de modificabilidad)

-Puedo enchufar y desenchufar filters cuando quiera

-Si surgen nuevos filters los agrego fácilmente

Al filtro en sí no le interesa cómo llegó su input de esa manera, lo toma y lo usa.

Tampoco le importa cómo el output llegará al siguiente filtro, ni cuál será ese filtro.

En general las definiciones de cada filtro no están en el mismo archivo y se levanten de manera dinámica.

CLASE 4

EJERCICIO PIPES & FILTERS

Teniamos el filterMultiply y filterPrint.

La idea era implementar un pipe and filter en donde uno de sus filtros sea una multiplicación y el otro un print. Las características de los filtros es que ambos reciebn un input y un next. Osea, con que van a empezar y éste puede ser el resultado de un filter anterior o el principio de la cadena, al filtro no le interesa esto para nada. Quizás hasta este filtro puede haber sido aplicado antes.

next(null,result) en caso de haber error iria en lugar del null. El parámetro result es lo que quiero pasar al próximo filtro. En el caso del print como no procesé la data mando el mismo dato que recibí, pero en multiply si hay modificación.

Los pipes no deberían tener lógica de ver si el dato que esta pasando sea correcto, el filtro es quien debe encargarse de eso en caso de ser necesario.

En caso de que llegue algo que no le sirve el filtro puede decir ta lo dejo seguir o sino lanzar un error por ejemplo y cortar la ejecución.

Lo mejor que puede pasar es que los filtros puedan ser intercambiables en cuanto a lugar de la cadena, pero en la práctica en realidad quizás no es tan así. Por ejemplo, si tengo un filtro que limpia el string de caracteres inválidos no tendría sentido sacarlo del inicio porque puede explotar el resto.

En la función diferBinding de pipeline.js lo que hace es diferir el enlace a runtime de que pipeline queremos usar. Se toma de el archivo config. Esto es solo a modo de mostrarlo, pero no es necesario hacerlo siempre en P&F.

Abstract Pipeline:

Contiene una clase con su mismo nombre que es abstracta, ahora, en js no tenemos controles de que siga/pertenezca abstracta. Es un tema de diseño, nada más.

El event emmiter es el que me permite usar los .on .error etc….

La función constructor hace un this.filters y les asigna vacío. Esto crea una variable filter, es como si la definiese arriba.

Las clases que hereden de esta podrán acceder a filters. Osea sabré cuántos filter hay y en qué orden serán ejecutados. Lo puedo ver en el use que lo que hace es recibir un filter y lo agrega al filters.

La manera de indicar que no puedo hacer new de abstract pipeline es tirar un error en el método run. Osea, cuando lo vayan a correr les va a decir che mira que estas equivocado.

El use devuelve this para que puedas hacer pepiline.use(…).use(…) y asi sucesivamente.

Direct Pipeline:

Copio el archivo de filters pq luego quizas se estan ejecutando varias cosas a la vez y siempre quiero tener los filters actualizados, recordar que en runtime voy a estar poniendo y sacando.

Ver que en el run estoy poniendo una función dentro de otra función, tengo que justificar por qué. En este caso no tiene sentido loop fuera de aquí entonces en parte esta justificado. Si lo hiciese por separado requeriría más parámetros.

En el ejemplo:

```
var a =1;

var b = 2;

setTimeout(function(){console.log(a+b)},1000);

console.log("hola")
```

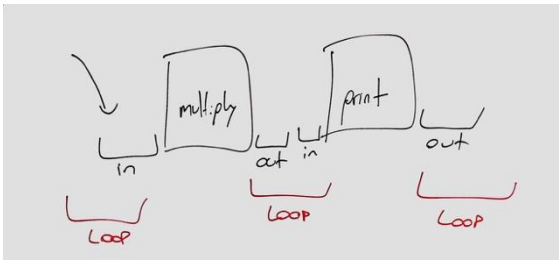
Por event loop intentaré ejecutar siempre el mismo nivel lo que no esté bloqueado entonces largo el timeout poniendo la función a pendiente de ejecución dentro de 1 segundo, e imprimo console.log("hola")

Qué pasaría si pongo 0 en vez de 1000? Pasaría lo mismo? Si, cuando se terminen las cosas para hacer en ese nivel va al siguiente.

Con el nextTick le digo al procesador que el vea cuando ejecutar el loop, acá no se ve mucho pero si se ve cuando tengo un proceso que recibe requests por todos lados y hay cosas que se están bloqueando y desbloqueando por todos lados.

La idea de loop es ejectuarse muchas veces entre los filtros.

En particular aquí:



.shift() es una función de los arrays y lo que hace es sacar el de la pos 0 y mover los ítems al principio.

filter.call(this, result, loop)

CLASE 5

Benchmark: Medir la performance desde distintos puntos de vista.

Escalabilidad vertical y horizontal, poner muchos servidores o ponerle mucho hardware a uno.

Vertical: aumento recursos a una máquina, osea, mejoro latencia y throughput, pero hasta cierto punto (punto de conveniencia). Imaginemos que tengo que cada usuario me ocupa 1mb y tengo 1 GB, tengo 1024 clientes que serán atendidos a la vez. Pero el 1025 va a sufrir la espera.

Hay veces que la tecnología me limita a mejorar, no aprovechan la mejora del hardware.

¿Cuándo un módulo se convierte en componente?

Hay un mapeo 1 a 1? No necesariamente, quizas un modulo logico levanta n componentes o que no levante componentes hasta determinado punto (on demand), o puede pasar que un componente sea levantado por varios módulos.

En el ejemplo de la clase anterior tenia que los filter que eran modulos se levantaban a demanda.

Es muy difícil replicar las bases de datos racionales de manera horizontal.

CQRS: La idea es que cuando vayamos a leer vamos por un lado y para escribir por otro, hacia la base de datos. Desde el punto de vista de modificabilidad, tengo que tratar de que la lógica de esto este lo más a bajo nivel posible para que no afecte. En cuanto a performance, la mejora ya que las escrituras no demoran las lecturas. La gran ventaja es que puedo optimizar por separado las lecturas de las escrituras. Copio las bases de datos en base a la necesidad, por ejemplo, si voy a tener muchas lecturas. Es más, puedo usar distintas tecnologías para las bases de datos en base a lo que voy a hacer con ellas.

Redis muy buena para leer y MongoDB para escribir.

Esto requiere mecanismos de sincronización entre las bases.

-Puede ser por ejemplo un trigger que cuando escribo le avise a la de lectura, pero acá me ato a la tecnología que usen.

-Al modelo de escritura le digo que escriba en la base de escritura y en las de lecturas, esto es malísimo, estoy matando todo lo que es CQRS. Solo logro que en la de escritura solo se escriba.

-Componente que al momento de escribir se le avisa y él escribe en las de lectura. Quizás este componente acumula y cada x tiempo sincroniza así mejoro performance. Tengo que ver el tema de que si acepto que a veces lean datos no actualizados.

CLASE 6

Diseño Arquitectónico

Diseño detallado vs Arquitectónico:

Toda arquitectura es diseño, pero no todo diseño es arquitectónico o arquitectónicamente relevante

Los diagramas que hacíamos en DA2 no tienen valor arquitectónico.

Nos interesa como interactúan los componentes etc

Características de un buen diseño arquitectónico

- Simple
- Difícil
- Intencional
- Clásico (atemporal)
- Copiable (no se copia y pega)
- Consistente
- Medible
- Existen heurísticas / principios

En obl vamos a tener layers, tiers, seguro!

CLASE 7

Para crear un nuevo módulo:

npm init

fs y os son módulos de node entonces no estan en el package.json

readFileStream es la mejor forma

Stream es como un puntero al archivo, no lo trae a memoria.

Luego de cada línea se emite un evento, entonces los vamos capturando.

Koa-router gestiona las rutas, es un middleware

Qué pasa si empiezo a agregar router.get muchas veces? Cada función tiene handlers para los callback.

Entonces, ¿Qué podemos hacer como primer paso para mejorarlo?

-Quitar rutas => Creamos carpeta controllers y archivo routes

Esto no soluciona aún.

Hagamos controllers que manejen las rutas de acuerdo

Ojo: ¡La función que uso el para leer un archivo carga todo en un array, si el archivo es muy grande puede reventar el array, podemos usarla si en cada línea vamos procesando y no guardamos en el array!

Ojo: ¡Cuando leemos del archivo todo viene como string, entonces ojo que hay que parsear para que queden como numeros!

EL size sirve para saber que no tengo que pedir más cuando sea más chico que el offset.

CLASE 9

En repo es la clase 7

Headers content-type y accept nos los mandan los clients para saber que tipo de respuesta esperan.

mongo d este comando levanta MongoDB

Las API tienen colecciones de postman para importar

Las requests estan en xml y en json, cual es la diferencia?

Cuando hacemos con JSON, le ponemos el accept

Las de xml le decimos que acepta xml.

Content-type es que va en ese formato y el accept indica que queremos que nos devuelva.

Como se implementa esto del lado del servidor? KOA recibe y devulve, le ponemos patrones que interceptan esas requests, ej le pusimos el de login.

ORDERS-API

Ahora agregamos un middleware nuestro que es respond

Los controladores también son middlewares, reciben ctx y next

La diferencia de este es que no tiene uri, intercepta siempre

Todos los middlewares los pongo con el app.use de la clase server.js

Node index.js para correr en el visual

Qué es MongoDB?

BD relacionales son buenas con las consistencias y disponibilidad.

Mongo es no relacional siendo buena en las particiones y consistencia que es el tercer factor del Teorema del CAP (No podemos tener los 3 factores, siempre sacrifico uno)

En Mongo no hay tablas, el análogo es collections. Es como una bolsa de objetos, no hay restricciones de esquemas, puedo grabar un usuario que solo tenga un nombre u otro que tenga varios campos, y asi sucesivamente. En mongo tampoco tenemos relaciones entre entidades.

Todo esto nos permite programar rapidamente, es más eficiente pero recordar que no tenemos consistencia. El lenguaje de consulta no es SQL, está basado en java script.

Por ejemplo:

Db.<coleccion>.<operación>

db.order.find()

En clase c4 hay dos ejemplitos para ver.

Mongus para mongo

Y sqlize o algo asi para sql

En clase 7 hay ejemplos mas profundos

- MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
- The document model maps to the objects in your application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
- MongoDB is free to use.

En mongo ya tengo un nivel de validación, eso es muy positivo.

¿Por qué mongoDB usa guid para los id?

Porque si tenemos por ej una tactica de replicar datos (replico la base de datos entera), si guardamos en una y en otra imaginemos que agrego dos elementos a la base 1 y dos a la base 2 y caen en distinto orden quedan con distinto id. Mongo DB asume que ya vas a hacerlo distribuido.

IMPLEMENTACIÓN DE TÁCTICAS

- PING/ECHO
    - Lo hariamos poniendo un componente que esté corriendo y pegándole a una API cada x tiempo.
    - Una librería que lo hace es la que vemos ping/echo-module
    - setTimeout es ejecuta una función al llegar al timeout
    - setInterval es ejecuta la función en el intervalo
    - El código está muy a bajo nivel pero lo bueno es que trae cosas que la táctica nos pide como el timeout.
    - pingService.on(failure,()) la funcionq ue pongo es para que se llame en ese caso, me suscribo a ese modulo.
- MONITOR
    - Vamos a usar PM2, es un monitor de eventos para nodejs.
    - Esto nos permite monitorear aplicaciones en nodejs y servicios que utilizamos, nos da métricas de nuestra aplicación tipo cuantas veces se reinició, cual es el delay del event loop (si es muy grande hay que revisar pq probablemente estemos bloqueando), cuanta memoria esta consumiendo la aplicación, entre otras.
    - Npm install pm2 -g esto quiere decir que cualquiera lo puede usar en nuestra computadora.

    - Pm2 start index.js
    - pm2 list
    - pm2 monit lo monitorea continuamente, se actualiza en real time
    - pm2 stop index – esto detiene el proceso
    - Si dejo pm2 corriendo y el watch esta en true, si cambio algo del código y guardo el se reinicia todo.
- MANTENER MULTIPLES COPIAS DE COMPUTOS
    - Load Balancing ve a quien manda, la decisión depende pero en general usamos Round Rubin, va calculando la carga en tiempo real mas o menos y la manda al que tenga menos carga.
    - En ecosystem.config.js defino que quiero dos instancias

MONGO

Performance es mucho major todo en una colección de mongoDB

mongoose es la librería para usar mongo

en index.js:

- Tengo un attributes, es para que la librería haga validaciones para guardar porque la base no lo hará, no es relacional.
- Dados los atributos creo los modelos

CLASE 10

Message Queue

En pub-sus todos tienen que estar activos sino no reciben. Acá si reciben.

redis-server para iniciarlo se puede instalar con homebrew

instalar bull en el proyecto luego del npm init

instalo el Proyecto de arena y lo ejecuto

http://localhost:4567/

corro todo y en el arena veo que falla porque por default tengo que mandar objetos

job.data en el consumer hasta ahí me da el objeto que mande los consiguientes puntos son el objeto que mande

Para procesar correctamente tengo que invocar el callback que es el done

Puedo poner políticas de retry también, ver la documentación de bull

Casos de discusión:

- Simular envío masivo de datos para ver el efecto en el consumidor
  - Podemos hacerlo con un for o usar el runner de clase 8 (c8)
  - Se lo banca bien
- Observar comportamiento si el consumidor demora en procesar los mensajes, para esto puede apoyarse en la función *setTimeout*
  - SetTimeout para hacer el caso en que demoro, primer parámetro es una función y dsp cuando quiero que demore en milisegundos
  - Nos damos cuenta de que no interesa la velocidad de quien produce, el consumidor lo procesará a su ritmo sin problema
- De manera similiar al punto anterior pero si Redis es detenido
  - Redis es en memoria pero persiste en disco, no se pierden
- Se puede tener dos consumidores?
  - Competing consumers == > Dos compiten para agarrar los mensajes, tengo mayor capacidad de procesamiento.

PARTE 5:

Ejecuto el orders api y el orders runner

No estamos preparados para recibir esa cantidad de requests en un tiempo tan corto.

Como mejorar esto?

Trade off entre disponibilidad o performance, en este caso necesitamos más performance.

Pongo las colas, cuando recibo la request lo único que hago es ver si lo puedo guardar, si puedo respondo y encolo el insertar. También podría agregar múltiples consumidores para guardar. Ojo porque esos consumidores están pegando a una misma base.

Hay que considerar el caso, no hacer sobrea arquitectura, ver con qué resuelvo de la manera más simple.

Ahora hacemos el ejercico de:

Publicamos en una cola las requests y retornamos enseguida, dejamos el insertar para otra cola. Estamos evitando la capa de validación para poder hacerlo más rápido.

CLASE 11

Exception Handling

Cuando detectamos una *exception* el sistema tiene que gestionarlo de alguna manera, lo mas sencillo que podemos realizar es simplemente un *crash* del sistema aunque desde el punto de vista de la disponibilidad, usabilidad y sentido común es una muy mala decisión. Existen otras posibilidades para gestionar errores, Javascript como muchos otros lenguajes ya contempla esta problemática de manera nativa. Para gestionar nuestros errores de manera efectiva antes tenemos que entender los conceptos de:

- Objeto Error
- Try...catch...finally
    - El catch tiene algunas particularidades:
        - Si tengo catch(e) atrapa todo, no puedo hacer un catch de una subclase. Podemos preguntar el typeof para segmentar.
        - Qué tiene que ver el nombre de las funciones con el manejo de errores?
            - Si no las nombro luego en el error no me dice en que funcion falla, etc. Me dice lineas que no sirven de nada.
- Throw
- Call stack
- Nombres de funciones
- Mundo asíncrono como promises y async/await
    - Cuando tengo funcion asincrona el error lo pongo en el callback, en sincronas lo pongo con try catch como siempre.

¿Cómo reutilizar nuestra librería?

Instalarlo localmente o enviarlo a npm.

Localmente tenemos que agregarlo en el development.json

PASOS:

1- Creamos el proyecto con el npm init como siempre
2- Creamos el index.js
3- Instalar dependencia local en el proyecto en el cual quiero usarlo
    a. Cómo?
    b. Voy a package.json en “dependencies”
        i. “logger” : “file:../../../logger” ->Tengo que apuntar a donde este el package.json de nuestra librería, porque alli esta el nombre, el archivo main y las dependencias.