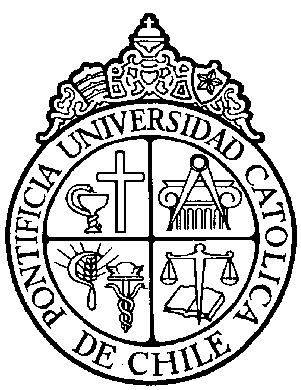
# Pontificia universidad católica de chile



# Escuela de ingeniería

# Departamento de ciencia de la computación

# 18 de Octubre de 2019

# IIC2173 Arquitectura de Sistemas de Software

##### Interrogación 2

##### Instrucciones: Sea preciso, no es necesario escribir mucho pero si ser preciso. En caso de ambigüedad use su criterio y agregue el supuesto en la respuesta correspondiente.

1. (2.0pts) Pheisbuk, una red social con millones de usuarios alrededor del mundo, en la cual Ud. y todos sus amigos/familiares están inscritos, está considerando integrar una nueva funcionalidad pensando en maximizar la personalizabilidad de los perfiles dentro de la plataforma. Para esto van a integrar una API estilo *webhook* para que cada vez que ocurra algo definido se notifique a uno o más endpoints que Ud. tenga configurados (por acción). Estas acciones pueden ser vistas a su perfil; like, reaction o un comentario a una publicación que haya realizado.
2. (1.4pts) ***¿***Cómo diseñaría Ud. la solución para cumplir con estas consideraciones?
   1. ¿Qué atributos de calidad quiere privilegiar? ¿Por qué? (al menos 2)
   2. ¿Qué patrón(es) arquitectónico(s) utilizaría? ¿Por qué?
   3. En el diagrama indique los conectores más importantes e indique (fuera del diagrama) el tipo, dimensión, sub-dimensión y valor.
   4. ¿Hay algún atributo de calidad (o más) que esté despriorizando en su solución? (explique)

R: Esperable respuestas basadas en eventos (invocación implícita) con un PubSub (PubSubHub)

1. (0.6pts)En una segunda etapa, espera liberar un *marketplace* de modo que se puedan compartir la forma de manejar la información de cada una de estas ocurrencias. En esta plataforma (*marketplace*) los usuarios pueden subir “add-ons” e instalar los de otras personas configurados con sus propios datos.
   1. ¿De qué complicaciones hay que preocuparse al momento de enfrentar la segunda etapa?
   2. ¿Existían antes de considerar los cambios?

R: Posibles temas de seguridad podrían salir. Es un tema a considerar desde la primera etapa.

1. (0.8 pts)Una gran empresa de productos electrónicos “inteligentes” se está enfocando en iluminar espacios de oficinas y de alumbrado público. Se han dado cuenta que sus productos tienen una conexión estable más del 99% del tiempo, pero también notaron que los relojes internos de éstos productos se veían distorsionados (atrasados o adelantados) de manera aleatoria.

La empresa ha tomado la decisión de diseño que las lámparas se sincronicen una hora antes de su horario de encendido y de apagado. Para ello han creado un complejo protocolo que realiza 3 *requests*, los cuales contienen el *timestamp* del cliente al momento de enviar el mensaje y en la respuesta del servidor va también el timestamp. Esto para tomar en consideración la posible latencia de cada dispositivo.

Mencione al menos dos problemas importantes (relacionado a los atributos de calidad) que se van a encontrar con esta solución y cómo las solucionaría Ud.

R: Disponibilidad: Dado que todos los productos van a hacer al menos 3 *request* más o menos a la misma hora, pero todos van a llegar levemente desfasados, por lo que tendrán gigantescos *peaks* de *requests* (auto DDOS). Esto hace que un par de veces al día se tuviese que escalar el servicio para manejar esta carga. Escalabilidad: Otro problema relacionado es que no puede utilizar caché, ya que dice que responde exactamente con la hora en la que se respondió y además registra información de cada interacción. En ambos casos una solución es distribuir los *requests* a lo largo de los períodos con los que se quiere sincronizar los productos y así nivelar la carga de los servidores.

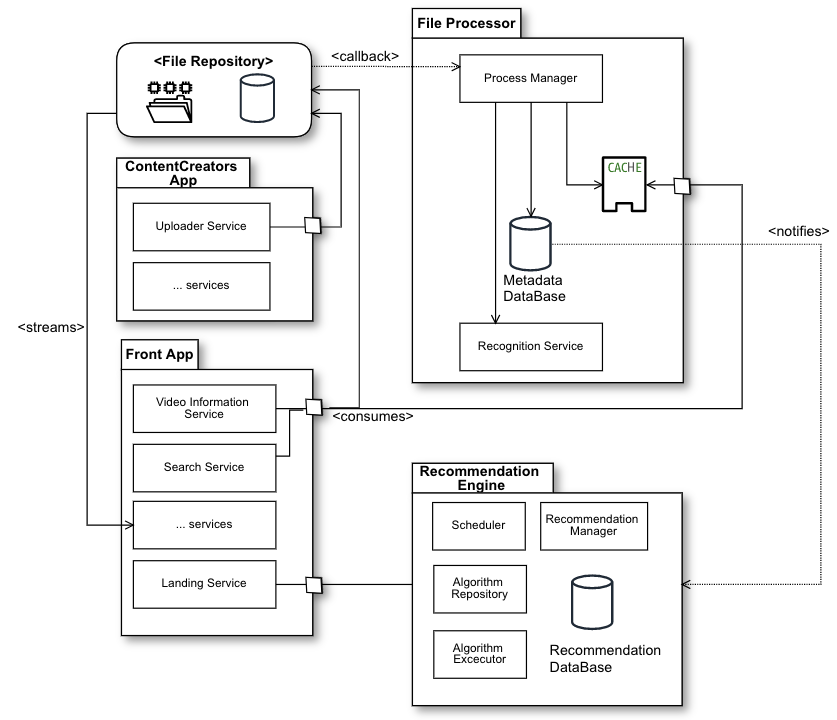
* *Bonus* si mencionan sincronización de servidores mediante vector clocks.

Performance: Para realizar una sincronización exitosa tanto el producto cliente como el servidor deben manejar correctamente los conceptos de latencia de la red, para evitar des-sincronizar los productos de manera casi aleatoria. Soluciones son principalmente basadas en software o en el algoritmo para realizar la sincronización. También ayudaría tener los servidores repartidos por el mundo para tratar de minimizar la latencia y distribuir (escalabilidad) los *requests* por zonas.

Autenticación: se podría tratar de falsear información de los productos creando clientes que repliquen los *requests* realizados por los productos. Este tipo de ataque de seguridad, también podría afectar la disponibilidad de los servicios.

\* En el enunciado se menciona que hay una alta conectividad, por lo que problemas de ese tipo deben ser bien fundamentados para considerar puntos.

1. (1.0pts) Considere el siguiente diagrama para su evaluación



Para comprender el diagrama considere una aplicación como *Netflix*, donde un grupo de usuarios “administradores de contenido” tienen su propio portal para subir y manejar contenido. Los videos (películas, documentales, series, etc) quedan almacenados en un sistema para guardar documentos/archivos, y su información básica queda accesibles en una base de datos.

Una vez que se sube o modifica un archivo/registro, se notifica a un sistema separado para realizar una serie de procesos de reconocimientos automático para identificar género, actores, trama y otra metadata, tanto del video como del contenido agregado por los “administradores”. La información generada en este paso se almacena en una base de datos para su futuro procesamiento y luego se deja en un caché para el consumo de otros servicios.

A su vez, esta información de metadata recolectada se notifica a un sistema recomendador que se ejecuta con una cierta periodicidad.

Por su parte, el cliente final accede al sistema mediante “FontApp” vía web o aplicaciones para sus dispositivos móviles. Esta aplicación se comunica con los servicios para tener el listado de películas, búsqueda, ver más información acerca de una película, así como tener recomendaciones personalizadas para él. Una vez seleccionado un video, se conecta al “File Repository” mediante un *stream* de datos.

1. ¿Qué atributos de calidad se están privilegiando en el diagrama?¿Por qué?
2. ¿Qué patrones arquitectura identifica en el diagrama?¿Por qué?
3. ¿Qué atributos de calidad se están dejando de lado? ¿Qué problemas pueden aparecer en el sistema?
4. Si tuviese que agregar la funcionalidad de evaluar un video, ¿cómo modificaría el diagrama?
5. (bonus) Si se quiere aumentar la conversión y se prueban distintos experimentos, como por ejemplo, buscar “la mejor carátula” para cada película, dependiendo del cliente. ¿Qué consideraciones se debiesen considerar para lograr este objetivo? ¿Qué información se debe rescatar?

R: Hay un intento por la gestión de recursos. La mantenibilidad se debería ver beneficiada dada la separación clara de responsabilidades de los servicios back.

En el diagrama se ven cliente-servidor, pipe&filter, pizarra

Falta replicar las películas para hacer el streaming mediante DNS preferentemente (mejor si mencionan algo de seguridad de acceso a los recursos en los DNS). Debido a lo anterior no escala tan bien en base a los usuarios finales. El sistema recomendador solo se está nutriendo de la metadata de los videos y no la info subida por los admins.

Para el rate service, agregar el servicio al front del cliente, agregar un back de ratings (podría ser del mismo back dentro de la FrontApp, que ya debería manjar la base de datos de usuarios) y agregar ese vínculo desde ese back al sistema recomendador.

Para optimizar la conversión en base a las carátulas se puede agregar dentro de FileProcessor como un nuevo algoritmo que tome imágenes descriptivas o llamativas (existen algoritmos para ésto) y extraerlas con su propia versión de la descripción del video y entregar las distintas versiones al sistema recomendador. Por su parte, el sistema recomendador entrega distintas versiones en la landing para medir la reacción del cliente final en base a los clicks (“ver más” y capturar esta interacción). Luego considerar esa información en la categorización del tipo de cliente, película y otros facotres.

1. (2.2 pts) Con un grupo de compañeros empiezan a idear un emprendimiento dedicado a brindar servicios de limpieza y conteo de inventario a grandes recintos, tales como malls incluyendo tiendas), bodegas de centros de distribución, cines, grandes tiendas, etc. (OJO: Debe considerar que los edificios pueden tener desniveles y/o escalones)

La idea es desarrollar, de manera interna y/o con asociación a otra empresa, robots enfocados en la limpieza (ej: iRobot Roomba), de conteo de inventario (ej: Zippedi), entre otros (pueden considerar más clases de robots inteligentes si así lo desean).

La interfaz de cada organización puede tener distintas instalaciones, en cada instalación pueden haber más de un “equipo” de robots y se pueden manejar de manera independiente. Desde esta interfaz pueden acceder al estado de cada robot y de los reportes generados. También permite manejar y calendarizar las ejecuciones de los equipos.

Para su respuesta considere al menos como hacen los equipos de robots para no realizar dos veces el mismo trabajo en el mismo lugar; la interacción entre los robots, la distribución de información y responsabilidades; cómo se redistribuye la carga en caso de que falle un robot.

\* los supuestos que tome para el diseño de su solución deben estar escritos al principio de su solución

1. (0.6pts) Indique (al menos 2) atributos de calidad que Ud. considere son fundamentales en este sistema. Enumérelos en orden de prioridad y justique.
2. (0.6pts) Indique los patrones arquitectónicos (al menos 2) que va a emplear para solucionar el problema. Indique qué aspecto del problema ataca cada patrón (responsabilidad).
3. (1.0pts) Haga un diagrama de bloques indicando los componentes arquitectónicos y sus relaciones que reflejen una solución al problema planteado. Añada tanto detalle como haga falta para que quede claro cómo el patrón soluciona el problema planteado.
4. (0.3pts) Para cada conector, indique el tipo, dimensión, sub-dimensión y su valor.

(Hint: apóyese en el apéndice entregado)

1. (0.3pts) ¿Qué atributos de calidad (al menos 1) se ven perjudicados en su solución? Justifique.

**Listado de Patrones**

1. Influenciados por el lenguaje
   1. Main y sub-rutinas
   2. Orientado al objeto
2. En capas
   1. Máquinas virtuales
   2. Cliente-servidor
3. Flujo de datos
   1. Por Lotes, secuencial
   2. Tubería y filtro (pipe & filter)
4. Memoria compartida
   1. Pizarra
   2. Basado en reglas
5. Intérprete
   1. Intérprete
   2. Código móvil
6. Invocación implícita
   1. Basado en eventos
   2. Publisher-Subscriber
7. Peer-To-Peer
8. Derivados
   1. C2, Corba

**Listado de atributos de calidad**

(\*): no se puede utilizar

1. Mantenibilidad
2. Flexibilidad
3. Testeabilidad
4. Escalabildiad
5. Disponibilidad
6. Seguridad
7. Usabilidad (\*)
8. Portabilidad
9. Reusabilidad
10. Confiabilidad
11. Integrabilidad / Interoperabilidad
12. Performance / Desempeño
13. Eficiencia
14. Gestión de recursos
15. Time to Market
16. Projected Life time
17. – entre otros -

**Conectores**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Type** | **Dimension** | **Sub Dimension** | **Value** |
| Communication  Coordination | Procedure call | Parameters | Data transfer | Reference, value, name |
| Semantics | Default values, keyword parameters, inline parameters |
| Return values |  |
| Invocation record | Push from L to R, push from R to L, hash table |
| Entry point | Multiple |  |
| Single |  |
| Invocation | Explicit | Method call, macro call, inline, system call |
| Implicit | Exceptions, callbacks, delegation |
| Synchronicity |  | Asynchronous, synchronous |
| Cardinality | Fan out |  |
| Fan in |  |
| Accessibility |  | Private, protected, public |
| Communication  Coordination | Event | Cardinality | Producers |  |
| Observers |  |
| Event patterns |  |
| Delivery |  | Best effort, exactly one, at most once, at least once |
| Priority | Outgoing |  |
| Incoming |  |
| Synchronicity |  | Synchronous, asynchronous, time out synchronous |
| Notification |  | Polled, publish/subscribe, central update, queued dispatch |
| Causality |  | Absolute, relative |
| Mode | Hardware | Page faults, interrupts, traps |
| Software | Signals, GUI input/output, triggers |
| Communication  Conversion | Data access | Locality |  | Thread specific, process specific, global |
| Access |  | Accessor, mutator |
| Availability | Transient | Register, cache, DMA, heap, stack |
| Persistent | Repository access, file I/O, dynamic data exchange, database access |
| Accessibility |  | Private, protected, public |
| Life cycle |  | Initialization, termination |
| Cardinality | Defines |  |
| Uses |  |
| Facilitation | Linkage | Reference |  | Implicit, explicit |
| Granularity | Unit |  |
| Syntactic | Variable, procedure, function, constant, type |
| Semantic |  |
| Cardinality | Defines |  |
| Uses |  |
| Provides |  |
| Requires |  |
| Binding |  | Compile-time, run-time, pre-compile-time |
| Communication | Stream | Delivery |  | Best effort, exactly once, at most once, at least once |
| Bounds |  | Bounded, unbounded |
| Buffering |  | Buffered, unbuffered |
| Throughput |  | Atomic units, higher-order units |
| State |  | Stateless, stateful |
| Identity |  | Named, unnamed |
| Locality |  | Local, remote |
| Synchronicity |  | Synchronous, asynchronous, time out synchronous |
| Format |  | Raw, structured |
| Cardinality | Binary |  |
| N-ary | Multi sender, multi receiver, multi sender/receiver |
| Coordination  Facilitation | Arbitrator | Fault handling |  | Authoritative, voting |
| Concurrency | Mechanism | Semaphore, rendezvous, monitor, lock |
| Weight | Light, heavy |
| Transactions | Nesting | Single, multiple |
| Awareness | None, supported, required, new |
| Isolation | Read, write, read/write |
| Security | Authentication |  |
| Authorization | Capabilities, access control lists |
| Privacy | Encryption, information padding, screening |
| Integrity | Redundancy check, certificates |
| Durability | Single session, multi session |
| Scheduling | Time |  |
| Weight |  |
| Conversion | Adaptor | Invocation conversion | Address mapping |  |
| Marshaling |  |
| Translation |  |
| Packaging conversion |  | Wrappers, packages |
| Protocol conversion |  |  |
| Presentation conversion |  |  |
| Facilitation | Distributor | Naming | Structure based | Hierarchical, flat |
| Attribute base |  |
| Delivery | Semantics | Best effort, exactly once, at most once, at least once |
| Mechanism | Unicast, multicast, broadcast |
| Routing | Membership | Bounded, ad hoc |
| Path | Static, cached, dynamic |