

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Compilado de Preguntas I2

IIC2173

Arquitectura de Sistemas de Software

2024-1

Recopilación por el equipo docente 2024-1

Índice

1. Preguntas tipo Quiz 3
2. Preguntas de Modelación 4
   a. I2 2021-2 4
      i. Pregunta 1 4
      ii. Pregunta 2 5
   b. I3 2021-2 6
      i. Pregunta 2 6
   c. I2 2022-2 7
      i. Pregunta 1 7
      ii. Pregunta 2 9
   d. I2 2023-2 10
      i. Pregunta 1 10

1. Preguntas tipo Quiz

1. ¿Cuál de las siguientes alternativas NO es un requisito no funcional (RNF)?
   - Diseño
   - Escalabilidad
   - Performance
   - Confiabilidad

R: Diseño

2. Imagine que implementará un sistema de votación universal donde las encuestas se pueden crear mediante clientes (CLI) para realizar encuestas y/o votaciones. Además, las votaciones se realizan con otro tipo de cliente (web o en app). La diferencia principal con los demás sistemas es que el mecanismo de registro de las votaciones se realiza con Blockchain. ¿Cuál(es) podrían ser los atributos de calidad (o RNF) que debería(n) ser fundamental(es) para el sistema?

R: Seguridad, Escalabilidad y Disponibilidad

Nota: ¿Por qué no performance? A priori no nos interesa si el sistema tiene un buen desempeño en el registro (guardar más rápido o más lento) ni procesamiento de los votos (de hecho, podría realizarse con otro servicio). Primero nos interesa que sea seguro (esencia de las votaciones), que sea escalable (universal) y que esté disponible la mayor parte del tiempo (cualquier horario).

3. Diga ventajas de la arquitectura Pipe and Filter:

R:

- Se puede reusar fácilmente
- Cada filtro que implementemos puede ser utilizado en paralelo a los otros.
- Podemos agregar filtros fácilmente al flujo

4. Si tuviera que diseñar un software que requiera la implementación de notificaciones para su funcionamiento. ¿Qué patrón arquitectónico utilizaría para su solución?

R: Cualquiera del estilo "Invocación Implícita". Es decir, Pub/Sub o Basado en eventos dependiendo del tipo de notificación. *

5. Se desea desarrollar un sistema para analizar el lecho marino a grandes profundidades usando submarinos que deben transferir información y localizaciones por ondas de baja frecuencia entre ellos, para mejorar la recolección de datos y sus interpretaciones. ¿Qué patrón arquitectónico utilizaría para su solución?

R: Peer to peer

6. ¿Qué es la escalabilidad? ¿Cuáles son sus tipos (cómo se puede implementar) y las características de cada uno de éstos?

R: La escalabilidad es un tipo de requerimiento no funcional, esta define la capacidad del software de adaptarse para alcanzar nuevos requisitos de tamaño y alcance.
Sus tipos más generales son:

- Horizontal: Añadir más máquinas.
- Vertical: Mejorar la máquina existente.

2. Preguntas de Modelación

a. I2 2021-2

i. Pregunta 1

En Brasil, el tema de la recolección de impuestos siempre ha sido un tema complejo. Dado que es una república federada, se ha hecho una división de impuestos a nivel federal, estatal y municipal. Un mismo bien puede ser gravado de distinta forma, dependiendo de donde se venda y por donde transite.

Especialmente notables son los impuestos sobre las empresas y la circulación de materiales y servicios. El gobierno chileno le envía como representante especial para diseñar un prototipo de sistema procesador en tiempo real de transacciones de las empresas locales, que sea capaz de responder en base a un precio bruto con los impuestos apropiados más el precio final calculado. La empresa antes de concretar una venta envía su precio bruto y usted debe responder con los precios finales desglosados

1. Un colega suyo le sugiere implementar los impuestos directamente, impuesto por impuesto en su código. ¿Qué implicancias tiene esto? ¿Es una decisión apropiada considerando el contexto de la solución?

R: No es una decisión apropiada puesto que dificultará de sobre manera la extensión y la flexibilidad del código, causando que realizar un pequeño cambio en el código requiera de demasiado tiempo. Además, cada municipalidad podría requerir hacer cambios y dada la extensión del país, abarcar todo con este método sería carísimo.

2. Indique (al menos 3) atributos de calidad que Ud. considere son fundamentales en este problema.

   Para esta solución se usaron:
   - Escalabilidad
   - Disponibilidad
   - Performance

3. Indique los estilos arquitectónicos (al menos 2) que va a emplear para solucionar el problema. Indique qué aspecto del problema ataca cada uno (responsabilidad).

Algunos posibles estilos son:
- Cliente-servidor
- Basado en eventos
- Pipe & Filter

*Nota: Pizarra **no** aplica en ningún caso.*

4. Indique medidas adicionales (al menos 2) que podría tomar para garantizar la disponibilidad de su servicio

R: Escalar horizontalmente el sistema procesador y usar un *load balancer* para distribuir la carga entre cada réplica del sistema. Si una réplica falla, el resto responde.

5. Haga un diagrama de componentes UML indicando los componentes arquitectónicos y sus relaciones que reflejen una solución al problema planteado. Añada tanto detalle como haga falta para que quede claro como resuelve el problema planteado.

R: Diagrama de referencia

Dado que diversas instituciones le han solicitado que usted ejerza una lucha fuerte (muchas veces abrumadora) contra la corrupción en LATAM, adicionalmente se le pide que reporte el resultado del cálculo de impuestos directamente (y lo más rápido posible) a cada uno de los gobiernos locales para que puedan saber exactamente cuál será la recaudación esperada de las empresas que le toca supervisar. La respuesta no es crítica para que el proceso principal continúe. Además, se le pide que agregue un componente verificador con IA (de un desarrollador externo) para cada *request* al interior de su servicio, el cual es capaz de emitir alertas cuando una transacción es sospechosa y reportarla al SRF (un SII de allá).

6. ¿Qué sistemas adicionales aparecen en esta adición? ¿Es posible fragmentar este monolito en servicios y si así fuera, en que servicios lo fragmentaría? En caso de que no, explique porque

R: Sistemas adicionales: componentes de notificaciones y componente verficador (IA). Se puede usar Pipe & Filter, middlewares, adapters.

7. ¿Cómo puede abordar el problema de que cada municipio pueda demorar minutos (u horas si el presupuesto de TI es bajo) en procesar una recepción?
   R: Adapter con Pub/Sub, además de implementar pruebas de recepción.

8. Haga un diagrama de componentes UML de esta variante indicando los componentes arquitectónicos y sus relaciones que reflejen una solución al problema planteado en base a las respuestas que incluyó arriba. Añada tanto detalle como haga falta para que quede claro como resuelve el problema planteado.
   R: Diagrama de referencia

ii.Pregunta 2

LegitBusiness, en una maniobra financiera feroz, ejerce una adquisición hostil sobre Torches Inc., una empresa de la Tierra Media que es capaz de generar alertas de invasión a diversos niveles de terrenos, desde una comarca hasta un reino completo. La empresa tiene como objetivo ofrecer al monarca de turno una interfaz única donde pueda seleccionar los objetivos de sus alertas y confirmar la emisión.

No obstante, esta empresa siempre ha tenido un problema grave: los múltiples receptores de las alertas tienen diferentes interfaces, tiempos de respuesta y confirmaciones. Los empleados previos, unos goblins con bastante pereza implementaron un sistema síncrono de cliente servidor que espera cada petición. Le piden que modele un sistema que pueda cumplir con las expectativas del monarca y sus jefes.

1. ¿Qué atributos de calidad (al menos 2) NO deberían ser priorizados en un sistema así? (considere que están observando cuidadosamente su presupuesto)
   R: Algunos son Performance, escalabilidad, flexibilidad, reusabilidad.

2. Indique (al menos 3) atributos de calidad que Ud. considere son fundamentales en este problema. Enumérelos en orden de prioridad.
   R: Para esta solución se plantearon:
   - Interoperabilidad
   - Confiabilidad
   - Seguridad
   - Usabilidad

3. Indique los estilos arquitectónicos (al menos 2) que va a emplear para solucionar el problema. Indique qué aspecto del problema ataca cada uno (responsabilidad).
   R: Para esta solución se plantearon:
   - Publisher-Suscriber
   - Cliente-Servidor

4. Indique medidas adicionales (al menos 2) que podría tomar para garantizar la disponibilidad de su servicio
   R: Escalamiento horizontal: replicación de servidores.

5. Haga un diagrama de componentes UML indicando los componentes arquitectónicos y sus relaciones que reflejen una solución al problema planteado en base a las respuestas que incluyó arriba. Añada tanto detalle como haga falta para que quede claro como resuelve el problema planteado.

R: Diagrama de referencia

b. I3 2021-2

i. Pregunta 2

Debido a la corrupción de su gobierno local, sabe que en su país están los HQ de una organización criminal dedicada a manufacturar productos de biohacking ilegales en varias fábricas alrededor del mundo. Usted se une a un grupo de Hacktivistas muy poderoso y le encargan como primera misión que se infiltre y recupere evidencia que permita identificar esas fábricas y señalarlas para que las destruyan otros gobiernos más responsables.

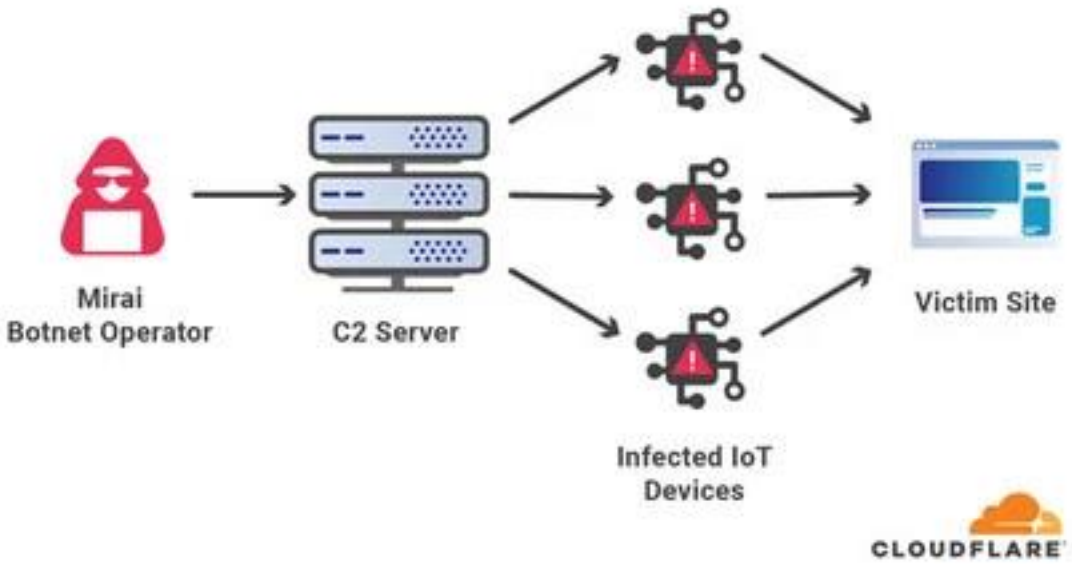
Esta organización es muy grande, de alcance mundial y posee muchos recursos. Según un informante, el área de tecnología cometió un error interesante. Utilizaron unas cámaras IP diseñadas por LegitBusiness las cuales tenían un error de seguridad gravísimo: La interfaz de servicio de cada cámara, especialmente diseñada para servicio técnico, quedó con las contraseñas por defecto (support/technician) y otorga acceso root al dispositivo. Además, por una extraña razón, los servidores on-premises de la empresa sufren de este mismo problema.

Conversando con sus colegas, deciden usar un vector de ataque basado en Mirai para tomar el control de las cámaras, con extensiones adicionales para aprovechar esta situación

Mirai es un malware que tiene como objetivo tomar el control de dispositivos IoT corriendo Linux, aprovechando el patente descuido de los fabricantes de diversos productos IoT, especialmente cámaras IP, DVRs, Routers y la cámara que acaban de diseñar.

Ustedes cuentan con el código base de Mirai, disponible para propósitos académicos en https://github.com/jgamblin/Mirai-Source-Code, además de que pueden buscar en Google el funcionamiento de este malware. Con las cámaras intervenidas, podrán recoger la evidencia que necesitan.

Diagrama de Mirai:



1. En base al diagrama, el contexto y conocimiento, responda:
    a. ¿Que RNFs se priorizan en el diseño de este virus?
    R: Escalabilidad, Flexibilidad y Portabilidad.
    b. ¿Cómo Mirai carga sus versiones de código? ¿Hay algún patrón asociado?
    R: Código móvil mediante un servidor de distribución (casi un CDN)
    c. ¿Cómo encuentra nuevos aparatos susceptibles de ser infectados?
    R: Mirai usa un rastreo sistemático para detectar nuevos dispositivos.

2. Dadas las condiciones de su caso, ¿con qué modificaciones podría mejorar la adquisición de nuevos dispositivos para su botnet?
    R: incluir las passwords mencionadas, usar los servidores de directorio, esparcirse horizontalmente con multicast.

3. Diseñe un diagrama de componentes que muestre como su variante de Mirai intervendrá los componentes de la cámara web, asegurará su persistencia y que aproveche 2 features adicionales de la cámara intervenida para recolectar evidencia.

R: Sin diagrama de referencia. Lo importante es seguir un flujo de: usar el motor de CV, interviene API, cambia contraseña de soporte y las sube al C&C.

c. I2 2022-2

i. Pregunta 1

La empresa FinTra, una nueva Fintech del mercado, los contacta con una idea revolucionaria para el desarrollo de una nueva aplicación. Se quieren integrar al mundo de las transacciones y los bancos para hacer de estas acciones algo más simples, y dadas sus buenas referencias desde la empresa LegitBussiness los contratan para esto.

Les piden diseñar una aplicación que permita ingresar la información necesaria para acceder a sus bancos (asegurando un 100% la seguridad de estos datos) y que, con esta información, pueda hacer un scrapping de sus páginas web según sea necesario para llevar a cabo una transacción a una cuenta que se le indique, esto porque se dieron cuenta que estas empresas usualmente no exponen APIs para estos servicios.

Esta aplicación además deberá guardar algunos contratos específicos como documentos formales estáticos según cada banco, esto pensando en futuras implementaciones que tienen planificadas, además de guardar las propias transacciones. Estos documentos se usarán para mostrarle en una vista de análisis ciertas estadísticas al usuario según palabras clave dentro de estos.

1. Indique si serviría usar caché dentro de la aplicación, y si es así, en qué parte (o partes) se aprovecharía mejor su uso. Justifique.

   R: Si. En CSS de bancos en el scrapper, u otro bien justificado. Tener ojo en que sea algo que no cambie constantemente, ya que en otro caso aplicar caché haría que el sistema sea más lento.

1. Indique cuál o cuáles otros tipos de bases de datos serían los necesarios para esta aplicación, justificando para qué usaría cada una (o cuál es la responsabilidad que tendría cada una).

   R: Mínimo de llave valor (caché) para guardar el CSS del banco del scrapper y NoSQL de documentos para el manejo de estos documentos de contratos que se guardarán de esta forma.

2. Se le indica además que la aplicación tendrá una gran cantidad de clientes en ciertos momentos críticos del mes (por ejemplo, a principio de este). Indique qué medidas implementaría a nivel de arquitectura para manejar esto.

   R: Sharding de la DB de contratos, LoadBalancer o AutoScalling Group , Caché a nivel de la aplicación (más que solo el scrapper), Otros dependiendo de la justificación.

3. Indique los atributos de calidad (RNF) que Ud. considere son fundamentales en este problema. Enumérelos en orden de prioridad y justifique brevemente.

   R: Algunas opciones son:

   - Seguridad – El sistema bancario requiere seguridad
   - Confiabilidad - No se puede permitir que la aplicación no haga las transacciones de acuerdo a los parámetros indicados
   - Escalabilidad - Se menciona que tendrá una gran cantidad de usuarios en momentos críticos del mes
   - Integración - Debe integrarse a distintos sistemas de bancos
   - Disponibilidad: Requerido para una buena experiencia de usuario
   - Otro bien justificado

   No aplica:

   - Mantenibilidad - No hay motivos que lo indiquen
   - Flexibilidad - No necesitamos modificarlo en el futuro como sistema, o no es esperable, los sistemas bancarios se demoran en cambiar
   - Usabilidad, salvo que se exprese como una consecuencia de los anteriores, dado que por sí mismo no tiene valor

4. Indique los estilos arquitectónicos que va a emplear para solucionar el problema. Indique qué aspecto del problema ataca cada estilo (responsabilidad).

   R: Algunos son:

   - Basado en eventos - podemos usar un bus de eventos para vincular distintos servicios como uno de autenticación y otro del accionamiento del scrapper
   - Pipe & Filter - Para el análisis de documentos
   - Client Server - Entre la aplicación Frontend y el servicio de manejo de las transacciones con el scrapper se crea esta relación.
   - Otro bien justificado

   No aplica:

- Pizarra (queremos usar muchas DB)
- Main y subrutinas: no es necesario

5. Confeccione un diagrama de componentes UML indicando los componentes arquitectónicos y sus relaciones, además de sus estilos especificados previamente, tal que reflejen una solución al problema planteado. Añada tanto detalle como haga falta para que quede claro cómo resuelve el problema planteado.

R: Diagrama de referencia.

ii. Pregunta 2

En el mundo del desarrollo se está potenciando cada vez más el concepto de Cloud IDE, o entornos de desarrollo integrados en la nube. Estos entornos son hosteados 100% en la nube, donde el desarrollador no requiere instalar software en su equipo o aprovisionar recursos propios, lo cual da diversas ventajas versus la instalación en una máquina propia.

LegitCloud, subsidiaria de LegitBusiness le ha pedido que implemente un pequeño prototipo de IDE online. Este IDE debe poder proporcionar un entorno seguro y mínimo de desarrollo virtual, fácilmente desechable y tener un tiempo mínimo de carga, así como recuperar su estado rápidamente. Se utiliza a través de un frontend web servido vía CDN.

Debe tener la capacidad de subir código a un repositorio, recuperarlo y hacer lo mismo con archivos blob (archivos binarios grandes como bases de datos, imágenes, videos, etc), los cuales se suben. Además, debe poder conectarse a un daemon que pueda manejar containers para que el programador pueda crear sus propios contenedores y utilizarlos en su IDE. Finalmente, debe poder ofrecer una terminal en tiempo real para que el usuario pueda ingresar comandos en una consola y manipular el entorno que se le ofreció al programador.

1. ¿Cómo podemos manejar la escalabilidad en la creación de estos entornos virtuales desechables?

R: Se pueden usar máquinas virtuales en hosts escalados horizontalmente, y separando responsabilidades de servicios podemos escalar unos u otros de forma independiente (e.g. solo escalar el servicio de snapshots si tuviera más demanda).
También se puede escalar en una misma máquina al estilo VPS.
Otra opción es que debe haber un componente centralizado que gobierne esta escalabilidad, asegurando consistencia. No puede ser un elemento Peer to Peer.

2. ¿Tiene sentido guardar todo el entorno (Sistema operativo, SSOO, herramientas incluidas con el sistema, librerías base) en este repositorio? ¿Cómo conecta esto con el concepto de layers en containers?

R: No. Solo necesitamos guardar en un volumen o una capa las modificaciones del usuario. Podemos prescindir de lo demás y si se requiere un entorno personalizado, este puede ser creado y replicado con containers.

3. ¿Qué tecnologías web podría utilizar para lograr la conexión en tiempo real de la consola con el navegador?

R: Websockets idealmente. Pueden mencionar requerimientos Cloud para mantener esto, pero como se pregunta web queda como opcional.

4. ¿Cómo conectaría los containers a su máquina principal?

R: Algunas opciones son:

- En el mismo runtime se puede especificar una conexión mediante un network driver.
- Adicionalmente debe disponibilizarse una API para el motor de containerizacion.

5. ¿Cuáles son los NFR más fundamentales, nombre 3 mínimo?
   R: En esta respuesta aplican:
   - Performance: Se requiere mucho performance para usuarios exigentes.
   - Escalabilidad: Se requiere escalabilidad horizontal/vertical y una capacidad de distribucion de recursos potente. Los IDEs pueden volverse muy pesados.
   - Flexibilidad: Debe tener la capacidad de hostear diversos entornos.
   - Portabilidad: Proporcionada por los containers.
   - Seguridad: Los usuarios son programadores y por lo tanto puede que manejen codigo crítico para la organización para la cual trabajen. Además, debe haber un sistema de usuarios, el cual tambien requiere seguridad.
   - Confiabilidad. Muy parecido a la seguridad. El código es un elemento delicado y es clave que se transfiera correctamente, asi como su capacidad de conservarse bien. Lo mismo para los snapshots de las capas de containers.

   No validos:
   - Integrabilidad.
   - Disponibilidad: De no estar bien justificado.

6. ¿Cuáles son los estilos de arquitectura más fundamentales, nombre 2 mínimo?
   R: Para este problema aplican:
   - Código móvil para el front y en el servidor.
   - Máquina virtual para los containers y comandos.
   - Eventos e interprete para la consola.
   - Cliente/servidor, mencionando cada uno de los puntos donde se ocupa.
   - N-Capas, En varios de los servicios (ver diagrama).
   - *Sobre eventos: Es posible proponer una arquitectura con ESB para los servicios, pero es una solución alternativa que debe estar muyjustificada.

   No aplica:
   - Pizarra
   - Pubsub: no hay broadcasting 1-N o N-N, o necesidad de invocación implícita
   - P2p: se pide una arquitectura centralizada
   - P&F: no es relevante para la solución: puede agregarlo en algún nicho pero no tiene puntaje
   - OO, main y subrutinas

7. Confeccione un diagrama de componentes UML indicando los componentes arquitectónicos y sus relaciones, además de sus estilos especificados previamente, tal que reflejen una solución al problema planteado. Añada tanto detalle como haga falta para que quede claro cómo soluciona el problema planteado.

R: Diagrama de referencia.

d. I2 2023-2

i. Pregunta 1

La comida vía delivery es un sistema muy común de adquisición de comida. El comfort es muy tentador y es posible conseguir comida a varias horas de forma sencilla.

No obstante, tiene varias falencias. Los usuarios pueden verse abrumados con la enorme cantidad de ofertas de comida y dado el limitado espacio de UI que tienen los usuarios para escoger en un celular, las opciones más masivas suelen dominar el espacio de venta, en detrimento de restaurantes más pequeños e innovadores, lo que limita la competencia.

Dada esta problemática, se le ocurre modelar un sistema llamado FeedMe, que ofrece un proceso literalmente invertido: un usuario publica sus necesidades de comida y los diversos restaurantes a modo de oferentes envían sus propuestas de servicio dentro de un plazo dado. En ese plazo, el usuario puede observar las propuestas escogidas en tiempo real a medida que van llegando. Debido a que son hechas en base a una petición abierta del usuario, pueden ser personalizadas y tienen su precio propio. En caso de que los usuarios no tengan claro como formular su petición, pueden pedirle a un modelo de AI (componente pre-armado) que en base a sus pedidos anteriores y/o datos personales (ubicación, nombre, datos de internet) les ofrezca vía una conversación una alternativa apropiada.

Este fomento de la competencia hace que los restaurantes pequeños puedan competir. Se espera que a cada solicitud que hagan los usuarios, se agolpen cientos de restaurantes a revisar en un portal propio específico lo que hay disponible y comienzan a emitir propuestas que puedan agradar al usuario.

Al seleccionar una propuesta, el sistema busca un repartidor disponible, el cual se inscribió previamente vía una app y sistema de registro propio. Se hace un matching en un componente de software dedicado y el usuario recibe un estimado del tiempo de llegada. Al llegar el pedido a la puerta del usuario, este queda registrado en sistema como completado.

Naturalmente, la privacidad de los datos de uso es clave, especialmente habiendo tanta información personal ligada a preferencias. Se espera que el sistema tenga métodos de manejo de esta información

Las siguientes preguntas están relacionadas con la meta de lograr un sistema capaz de manejar todos los puntos anteriores. Se espera una adopción inicial de 150 restaurantes y un registro aproximado de 30000 DAU en la primera semana.

1. Investigue lo que es una base de datos de documentos y como podría apoyar con la flexibilidad requerida en los datos a almacenar

   R: El alumno puede no describir por completo lo que es una base de datos de documentos. Se tomará como válido si aplica los principios BASE y su diferenciación vs las otras Bases de Datos No-SQL.

2. Comente la importancia del manejo de autenticación y autorización en un sistema de este estilo, así como las implicancias de su implementación.

R: Posibles respuestas:

- Debe mencionar que es importante
- por el manejo de usuarios tipo "persona" que conllevan leyes y regulaciones respecto a sus datos personales
- La gran cantidad de información que requiere diversos niveles de acceso por la presencia de diversos roles
- El acceso público a varias de las funciones requiere control extra
- Existe manejo de pagos por lo que los datos de pago deben estar muy asegurados

3. Indique (mínimo 3, pero pueden ser más) atributos de calidad vistos en clases que Ud. considere son fundamentales en este problema. Enumérelos en orden de prioridad y justifique brevemente.

   R: Válidos:

   - Performance
   - Escalabilidad

- Elasticidad (solo si se justifica correctamente)
- Seguridad
- Usabilidad
- Otro bien justificado

No válidos:
- Integrabilidad (es un único sistema, los restaurantes se conectan a tu app)
- Portabilidad (todo código funciona en un solo ambiente)
- Soporte

4. Indique (mínimo 2, pero pueden ser más) los estilos arquitectónicos vistos en clases que va a emplear para solucionar el problema. Indique qué aspecto del problema ataca cada estilo (responsabilidad).
   R: Válidos:
   - Pub/Sub
   - Basado en eventos
   - En lotes secuenciales
   - En capas
   - Otro bien justificado

   No válidos:
   - Peer to Peer (Todos los sistemas son distintos)
   - Código móvil (No hay inyecciones o traspaso de código)
   - Main y subrutinas (es bloqueante)
   - Intérprete

5. Confeccione un diagrama de componentes UML indicando los componentes arquitectónicos y sus relaciones, además de sus estilos especificados previamente, tal que reflejen una solución al problema planteado. Añada tanto detalle como haga falta para que quede claro cómo soluciona el problema planteado. Incluya anotaciones explicando funcionamiento y supuestos, así como nombres de interfaces
   R: Diagrama de referencia