Pauta I1 2025-2

1. Preguntas de desarrollo (4 ptos)

1.a) (4 ptos) Dado el siguiente mensaje HTTP, indique si cumple todos los niveles de interfaz uniforme de REST, y si no es así indique qué cosas le faltan. Para esto, indique todos los errores que pueda encontrar para cada nivel.

Request:	*Response:*

GET /flight/select_seat HTTP/1.1
 	
Headers: {
Content-type: application:json
}
 	
Body: {
“fligth_id”: 102,
“seats”: [“2A”, “2B”]
}
	HTTP/1.1 201 CREATED
	
	<flight>
	<id>102</id>
	<seats>
	<item>2A</item>
	<item>2B</item>
	</seats>
	</ftem>

Respuesta: Se debe dar un punto por cada Restricciónde Interfaz Uniforme correctamente indicado (0.5 ptos) e indicado su error o si no hay error (0.5 ptos).

1. *Recursos únicos y opacos: Se accede al id desde el body, no desde el recurso.*
2. *Los recursos son conceptuales y se manipulan mediante representaciones: Se pide mal configurado y se envía Application/json y se responde en formato XML sin header indicandolo.*
3. *Los mensajes son autodescriptivos: La request para crear o asignar asientos es GET en vez de POST/PUT. La response se responde correctamente para POST.*
4. *HATEOAS: No está implementado el seguimiento mediante links de HATEOAS.*

2. Pregunta UML 1 (29 ptos)

El mercado de vivienda familiar es sumamente demandado, donde se mueven grandes sumas de dinero y hay demanda de sobra por lugares donde habitar. En esta linea, la búsqueda por una plataforma completa para posicionar ofertas en este mercado se convierte en un lugar atractivo. LegitBusiness le encarga el diseño de un sistema novedoso llamado LegitHomie que no solamente ofrezca la lista de hogares disponibles para arrendar/vender, sino que incorpore una serie de features de interés para los posibles arrendatarios/compradores.

La primera feature clave es el posicionador multiportal. Su sistema debe ser capaz de colocar las propiedades posicionadas en su portal en otros portales similares, utilizando sus APIs cuando estén disponibles o mediante un bot por sitio capaz de hacer todas las operaciones requeridas para mantener esa publicación. El posicionador además manejará las cuentas de acceso para estos portales guardándolas directamente en su base de datos para uso por otros módulos de LegitHomie.

LegitHomie también tiene la capacidad de verificar la autenticidad de la oferta posicionada. Al momento de publicar la oferta deseada, se le solicitarán diversos documentos al oferente que validen que la propiedad sea de quien publica, como su identificación (que se verifica con el registro civil), certificado de dominio vigente (que se verifica con el conservador de bienes raíces de la región) y otros antecedentes (que se verifican con el SII. Estos serán automáticamente validados por LegitHomie mediante un módulo validador con AI. Al quedar una oferta validada, se le mostrará una insignia al posible comprador.

Finalmente, LegitHomie usa un interesante sistema de validación de propiedad. Cuando una persona o empresa pasa el proceso de verificación, se le envía un representante al publicador. Este lleva un pequeño drone autónomo, capaz de recorrer todos los rincones de la casa con un programa de navegacion automatico al ser activado dentro de la casa. Al finalizar el recorrido, genera un video (AVC en mkv) y un mapa 3D (stl) el cual es recibido por LegitHomie para ser analizado vía internet. Dentro de LegitHomie, el propietario podrá subir fotos, las cuales serán comparadas mediante un motor de correlación para que no se suban fotos que no correspondan al lugar. Fotos, video y mapa serán disponibilizados para que el posible arrendador los vea.

Diseñe un sistema que cumpla con las características solicitadas

Puntos de interés

- El registro civil y los conservadores regionales requieren un adapter
- El drone se activa a mano y se le introduce el id de la casa escaneada
- La navegacion del drone no está en el scope, es 100% autónoma. SI que es parte de su scope lo que pasa con el registro
- El motor de correlación es un módulo blackbox listo para usar, con 100% de precision y exactitud.
- Se espera una demanda sostenida de aproximadamente 100000 weekly active users (WAU) con un 70% principalmente correspondiente a arrendadores/compradores

2.a) (3 ptos) El registro civil y los conservadores regionales solamente funcionan mediante APIs poco integrables, que utilizan un protocolo muy específico y una sintaxis basada en XML que solo un par de desarrolladores de LegitHome entienden. Como podría mejorar la integrabilidad de su sistema con estos proveedores? Que tácticas de integrabilidad usaría? Justifique

El conocimiento de como integrarse con estos proveedores y la abstraccion de complejidad puede lograrse mediante varios métodos, pero para efectos del curso lo mejor es usar un middleware que contenga esta lógica especial, y que pueda presentar una interfaz sencilla que el resto de devs pueda entender. Debe especificarse que esta interfaz interna debe ser compatible con el resto de sistemas disponibles (e.g. si especifica EDA deben estar asociadas a callbacks o broker) y que hay un costo
Debe mencionarse que se está usando encapsulamiento de complejidad, abstraccion de servicios comunes y uso de intermediario (literal, equivale a 1.5 ptos)

2.b) (3ptos) Debido a que la AI para validación de documentos se determinó como costosa, se demandó la presencia de un módulo para validar los documentos a un costo inferior. ¿Qué estilo(s) podrían usarse en un módulo así? Justifique.

Principalmente se podría usar cli/sv EDA para un módulo así. Depende mucho de la justificacion del alumno pero debería especificar un módulo que pueda ser utilizado por varios servicios, con una interfaz clara y sencilla.

2.c) (3 ptos) 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.

> ***Respuesta:*** *Se dará 1 pto por cada uno correctamente justificado. En caso de que se enumeren más, se distribuirá el puntaje entre todos los mencionados.*
>
> - *Son válidos:*
> - *Integrabilidad: Clave, debe ser mencionado si o si*
> - *Escalabilidad: Es posible que las busquedas y posicionamientos aumenten a medida que se usa y deja de usar el portal.*
> - *Confiabilidad: Hay tratamiento de info compleja*
> - *Elasticidad: drones subiendo videos pueden requerir peaks de demanda*
> - *Cualquier otro bien justificado*
> - *No gana por mencionar*
> - *Performance*

    - *Mantenibilidad*
    - *Portabilidad*
- *No aplican:*
    - *Usabilidad*
    - *Eficiencia*
    - *Soporte*

2.d) (4 ptos) 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).

> ***Respuesta:*** *Se dará 1 pto por cada uno correctamente justificado. En caso de que se enumeren más, se distribuirá el puntaje entre todos los mencionados.*
>
> *Son válidos:*
>
> - *EDA para coordinar consultas y middleware*
> - *Puede ser un monolito de capas pero debe ser muy bien modelado*
> - *Cli/sv en el front o entre servicios (solo ahi y no aporta + de 1 pto)*
> - *SoA*
> - *Microservicios, solo si justifica apropiadamente. No es necesario usarlo pero si lo hace, que respete los principios de microservicios*
>
> *No aplican:*
>
> - *Basado en reglas*
> - *Memoria compartida o agentico*
>
> *Cualquier otro no suma a menos que el ayudante discrimine que está bien integrado.*

2.e) (11 ptos) 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

> ***Respuesta:*** *Si no es un diagrama de componentes UML no se corrige. Como puede haber muchas soluciones al problema, se analizará en base a descuentos:*
>
> - *(-3 c/u) No usan los estilos o el componente que envía los datos que justificaron previamente*
> - *(-2 c/u) Falta un requisito funcional*
> - *(-1 c/u) Interfaces mal implementadas, max. 8 puntos de dscto*
> - *(-1 c/u) Componentes desagregados (sin conectores)*

- *(-2) No hay nombres en interfaces*

2.f) (4 ptos) Identifique 2 trade-off originados de la solución de su diagrama. Comente como las solucionaría y las implicancias de estas soluciones. No se corregirán aquellas que apunten al formato de cómo se diagramó o si faltó explotar algún componente

> ***Respuesta:*** *Se esperan trade-off asociados a su solución, puede ser por una decisión de implementar un estilo u herramienta o por el mismo diseño de su sistema. Sobre la distribución de puntaje:*
>
> - *(1pto) Identificar un trade-off correctamente*
> - *(1pto) Identificar correctamente una posible solución aplicable a su sistema*

3. Pregunta UML 2 (29 ptos)

La integración de los conocidos "*Cyber Day*" en la vida de las personas se ha vuelto un evento más esperado con el pasar de los años, en donde los usuarios esperan a esta fecha para cotizar y comprar productos que han necesitado a precios más baratos.

Es por esto por lo que se les ocurre crear una aplicación para comparar precios basados en los intereses de los usuarios a lo largo del tiempo, generando así algo similar a una lista de deseos de productos, y como su vida universitaria les impide hacer la aplicación directamente, deciden diseñar primero su idea usando sus conocimientos aprendidos de su curso favorito: ¡Arquitectura de Sistemas de Software! (o que sus profesores esperan que lo sea).

Así, piensan que la aplicación tendrá como funcionalidad principal el que los usuarios puedan iniciar sesión y, basados en un listado de productos podrán elegir los que les interese en la lista. Para esto deberá tener un buscador de productos, además de mostrarle su valor histórico de precios de cada uno de estos, un compilado y resumen de todas las opiniones que los distintos usuarios han dejado en su página o en la de las tiendas que los venden, y la opción para poder dejar el comentario respectivamente.

Para poder hacer esto, se les ocurre que su página esté constantemente buscando

información en internet para ir actualizando los productos y precios. Así, su sistema deberá ser capaz de ejecutar una vez al día una serie de *scrappers* que vayan navegando por internet en búsqueda de cambios en los precios de cada producto, que luego deberán ser notificados directamente a su sistema principal para actualizar la vista de los usuarios vía un *broker* pub/sub.

Finalmente, para agregarle un valor extra a su página y que se diferencie de la competencia, deciden publicar esta información vía una API de pago. Para esto deberán integrar el sistema de pagos que se muestra a continuación, de forma que todas las *requests* se verifiquen que han sido pagadas antes de ejecutarse.



3.a) (4 ptos) Dado que su aplicación tiene scrappers que se demoran en ejecutar, deciden separarlo de su aplicación principal para levantarlo en una máquina aparte. Esto les genera un problema de tomar la decisión entre querer que esta parte del sistema se coordine vía coreografía o mediante orquestación. Indique (a) la diferencia entre ambos, (b) cuál quisieran usar, (c) una desventaja de su decisión y (d) cómo lo resolverían. Para esta pregunta pueden apoyarse de links externos indicando su fuente.

> ***Respuesta:*** *Para tener el puntaje completo se deben responder todas las preguntas:*
>
> *A) (+2 ptos) Coreografía: Los sistemas se coordinan entre ellos directamente y se notifican de errores o de términos de flujos para continuar.*
> *Orquestación: Hay un servicio encargado de hacer seguimiento de los demás sistemas para coordinarlos.*
>
> *B) Solo es indicar cuál se usará, no tiene puntaje.*
>
> *C) (+1 pto) En coreografía se pierde mantenibilidad ya que la coordinación es más difícil. En orquestación se pierde performance y escalabilidad ya que tenemos un SPOF.*
>
> *D) (+1 pto) Depende de la justificación del alumno el cómo lo quiera solucionar. Debe estar bien justificado y ser una solución que aplique al problema para tener el punto completo.*

3.b) (2 ptos) Como su sistema debe integrarse a varios sistemas vía API, indique si REST es una buena opción para esto. En caso positivo indique el por qué, en otro caso indique que otra tecnología de integración vista en clases usaría y por qué sería mejor que REST.

> ***Respuesta:*** *Se entrega puntaje en base a la justificación. Se espera que se use gRPC debido al sistema de notificaciones en tiempo real o MOM, pero para tener todo el puntaje se puede justificar refiriéndose a la performance y al tipo de envío de datos.*
>
> *Si se dice que si es buena opción, se debe indicar al menos por qué su uso de endpoints serviría y la facilidad (1 pto), pero se deberá indicar una medida para hacer el envío en tiempo real para tener el puntaje completo (1 pto).*

3.c) (3 ptos) 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.

> ***Respuesta:*** *Se dará 1 pto por cada uno correctamente justificado. En caso de que se enumeren más, se distribuirá el puntaje entre todos los mencionados.*
>
> Son válidos:
>
> - *Performance*
> - *Integrabilidad*
> - *Escalabilidad*

- *Usabilidad. Si se justifica que puede ser usado por diversos usuarios no nativos digitales.*
- *Seguridad. Por manejo de transacciones*
- *Confiabilidad. Por manejo de servicios por pagos.*
- *Otro bien justificado*

No aplican:

- *Mantenibilidad. No se pide manejo de errores. A excepción si en la pregunta 3.a se indique que se solventará la mantenibilidad y se justifica con eso.*
- *Flexibilidad. No hay extensión de funcionalidades*
- *Reusabilidad. No se usarán las piezas del código*
- *Portabilidad. No se indica que se ejecutará en múltiples ambientes para que sea necesario priorizarlo*

3.d) (4 ptos) 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).

Respuesta: *Se dará 1 pto por cada uno correctamente justificado. En caso de que se enumeren más, se distribuirá el puntaje entre todos los mencionados.*

Son válidos:

- *EDA*
- *Pub/Sub*
- *Cliente Servidor. Solo para la API de integración con otros sistemas*
- *En capas*
- *Pipe & Filter. Para verificar que un usuario está suscrito antes de permitirle conectarse*
- *SOA. Siempre y cuando se enumeren cuáles serán los servicios a desglosar.*
- *Microservicios. Siempre y cuando se enumeren cuáles serán los servicios a desglosar.*
- *Otro bien justificado*

No aplican:

- *Código móvil. No es necesario ejecuciones de código ni envío de este como dato.*
- *Pizarra. No es necesario tener múltiples fuentes de datos*

- *Main y Subrutinas. El que sea bloqueante dificulta su uso*
- *Peer to Peer. No es necesario su uso y añadería complejidad innecesaria.*

3.e) (11 ptos) 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

> ***Respuesta:*** *Si no es un diagrama de componentes UML no se corrige. Como puede haber muchas soluciones al problema, se analizará en base a descuentos:*
>
> - *(-3 c/u) No usan los estilos o el componente que envía los datos que justificaron previamente*
> - *(-2 c/u) Falta un requisito funcional*
> - *(-1 c/u) Interfaces mal implementadas, max. 8 puntos de dscto*
> - *(-1 c/u) Componentes desagregados (sin conectores)*
> - *(-2) No hay nombres en interfaces*
>
> *Diagrama de referencia:*

3.f) (4 ptos) Identifique 2 trade-off originados de la solución de su diagrama. Comente como las solucionaría y las implicancias de estas soluciones. No se corregirán aquellas que apunten al formato de cómo se diagramó o si faltó explotar algún componente

> ***Respuesta:*** *Se esperan trade-off asociados a su solución, puede ser por una decisión de implementar un estilo u herramienta o por el mismo diseño de su sistema. Sobre la distribución de puntaje:*
>
> - *(1pto) Identificar un trade-off correctamente*
> - *(1pto) Identificar correctamente una posible solución aplicable a su sistema*