IIC 2143 Ingeniería de Software
Examen - Semestre 1 /2020
Secciones 01 y 02

En caso de responder este enunciado en papel, responda cada pregunta en una hoja separada. Recuerden que están bajo el código de honor.

Pregunta 1:

En una concesionaria de vehículos motorizados, existen vendedores especializados en la venta de automóviles o motocicletas. Se asume que el primero solo puede vender autos, y el segundo solo puede vender motos. Con el fin de promover la proactividad de los vendedores, la empresa ha decidido poner en marcha un sistema de comisiones que consta de dos partes.

En primer lugar, para promover la venta de vehículos, a cada vendedor se le otorga un bono cada vez que logra concretar una venta. En segundo lugar, para promover el reclutamiento de nuevos vendedores, cada vendedor puede recomendar a una persona en la empresa. Si el individuo resulta contratado, quien lo recomendó pasa a ser su "padrino", y este pasa a recibir una parte de las comisiones del recién contratado de ese momento en adelante. Nótese que esta segunda política es recursiva: al concretar una venta, eventuales padrinos de padrinos también reciben un bono.

Por cada venta de un automóvil, el total de bonos a repartir corresponde al 2% del valor del vehículo para automóviles con un valor inferior a 20 millones de pesos; y al 5% del valor del vehículo para automóviles con un valor superior o igual a 20 millones. Lo anterior tiene un tope de 2 millones en bonos a repartir. Por cada venta de una motocicleta, el total de bonos a repartir corresponde al 3% del valor del vehículo independiente del valor de la motocicleta y sin tope superior.

Del total de bonos a repartir, a un vendedor le corresponden 10 "partes" del bono, mientras que a todos sus eventuales padrinos el corresponderían 1 "parte". Por ejemplo, si un vendedor tiene 3 padrinos en su cadena de apadrinamiento, el bono se debe dividir en 10 + 3 = 13 partes. Luego, al vendedor le corresponderían 10/13, y a cada padrino le correspondería 1/13.

Su tarea consiste en crear una clase `SalesmanFactory` que implemente el método `create_salesman`. Este método debe retornar con un 50% de probabilidades a un vendedor de automóviles, y con un 50% de probabilidades a un vendedor de motocicletas. Adicionalmente, al momento de crear a un vendedor, el método `create_salesman` le asigna automáticamente como padrino a un vendedor creado previamente (si es que existe). Todo vendedor retornado por este método debe definir un atributo `@commissions` que guarde el total de comisiones a recibir por el individuo; y debe implementar el método `complete_sale(amount)`, el cual se ejecuta luego de concretar una venta por un vehículo con un costo igual a `amount`, y actualiza el valor de `@commissions` del vendedor y sus eventuales padrinos según la lógica aquí descrita.

Pregunta 2:

Una solución informática de deliveries a domicilio le permite a los usuarios realizar encargos a través de una aplicación móvil (para la cual existe una versión en Android y otra en iOS) o bien a través de una página web montada en AWS EC2. Independiente de la plataforma que se utilice, para realizar un encargo es necesario en primer lugar seleccionar los productos que a uno le interese comprar y luego pagar el precio correspondiente utilizando la plataforma Transbank como intermediario. Transbank a su vez valida las credenciales bancarias del cliente con su institución financiera correspondiente. Cuando se confirma una orden, esta se envía a un servidor optimizado para esta tarea que se encuentra montado en Microsoft Azure VM. Este servidor está más bien orientado a administradores y proveedores, y ofrece interfaces para gestionar encargos pendientes, inventario, registro de proveedores y de repartidores. Nótese que al llegar un nuevo encargo, este solo se añade al registro de encargos pendientes si es que se valida que el producto está disponible en el módulo de inventario.

La aplicación web del cliente almacena recursos estáticos en Amazon S3 y persiste su información en una base de datos Postgres RDS. La aplicación web de administración persiste información operacional en Azure SQL Database, y persiste información de reporting en Azure SQL Data Warehouse. Esta última se utiliza para brindar visualizaciones de datos con Power BI.

Cuando se ingresa un pedido, la plataforma procede a enviar una notificación Push a repartidores designados dotados de otra aplicación móvil. Esta segunda aplicación (también disponible en Android y iOS) se encarga de listar todos los pedidos que un repartidor debe enviar y rastrear su posición en tiempo real vía GPS. El estado del envío y la posición del repartidor pueden ser simultáneamente revisados a través de las aplicaciones del cliente. Note que toda notificación Push proveniente de un servidor debe pasar a través de los servidores de Google y Apple respectivamente para ser ruteada a los dispositivos móviles finales.

Construya los diagramas de contexto, contenedores y componentes que se pueden desprender de este enunciado según el modelo C4.

Pregunta 3:

La pandemia que ha azotado este año al mundo ha disparado la utilización de aplicaciones que permiten hacer conferencias (Zoom, Meet). Muchas empresas están trabajando en nuevas herramientas que permitan ir más allá de lo que hoy es posible. Entre las funcionalidades que se desean incluir están las siguientes:

1. Poder compartir no solo uno sino múltiples documentos pertenecientes a más de un participante.
2. Envío privado de documentos entre los participantes durante la sesión (sin compartir).
3. Poder interactuar con la aplicación enteramente por voz (agendar, entrar, salir, poner o sacar cámara, mutear, grabar, pausar, etc).
4. Poder tener cuartos temporales donde dos o más miembros de la conferencia puedan conversar algo en privado sin abandonar la conferencia.
5. Muting automático inteligente (cuando se detecta que hay ruido de fondo, gritos de niños, llamada telefónica, etc. por sobre la voz del participante).
6. poder hacer contacto visual entre personas (detección del movimiento de ojos).

Se espera tener un producto con todas esas funcionalidades en 6 meses. Se ha hecho un levantamiento en forma de relatos de usuario para el primer release que debe estar terminado en 1 mes. Este debe incluir todo lo básico (lo que hoy día tienen estas aplicaciones), más todo lo asociado a la funcionalidad 1, con lo que se obtiene lo siguiente:

Relato	Esfuerzo (staff hours)		Prioridad
	Optimista	Pesimista	
1	12	24	media
2	18	30	alta
3	6	18	alta
4	18	24	media
5	16	36	alta
6	18	30	baja
7	6	18	alta
8	12	24	media
9	10	30	media
10	12	30	baja
11	18	32	alta
12	18	40	baja

La implementación de las funcionalidades asociadas a las siguientes etapas se espera sea entregada en 3 releases adicionales durante los 5 meses siguientes a la entrega del release 1. Los esfuerzos para los grupos de funcionalidades se han estimado en:

Funcionalidades	Esfuerzo (staff hours)	
	Optimista	Pesimista
2	150	250
3	200	300
4	100	200
5	250	340
6	70	90

El equipo de desarrollo es de 3 personas una de las cuales trabaja en jornada completa (44 horas semanales) y los otros dos a media jornada (22 horas).

Se pide:

a) Hacer una planificación a nivel de release para el primer release de la aplicación

b) Hacer una planificación a nivel de producto para los siguientes 5 meses que considere todo lo que se le quiere incorporar.

Pregunta 4:

En una clase que modela una factura de venta se tiene un atributo que corresponde al monto neto (iva incluido) y un método que calcula el monto total a partir del monto neto en base a tres posibles descuentos: un descuento por el monto de la factura, uno que incentiva la primera compra y uno que premia la lealtad de los clientes más antiguos.

Descuento por monto - Toma en cuenta el monto neto de la factura
monto_neto < 10.000 - no hay descuento
10.000 ≤ monto_neto < 100.000 - descuento 5%
monto_neto ≥ 100.000 - descuento 10%

Descuento por primera compra - Opera como incentivo a la primera compra
Es la primera compra - 20% de descuento
No es la primera compra - no hay descuento

Descuento por antiguedad del cliente - Premia la lealtad del cliente
Cliente de hasta 3 meses de antigüedad - no hay descuento
Cliente de 3 meses o más de antigüedad - 2% de descuento
Cliente 6 meses o más de antigüedad - 4% de descuento

Los descuentos aplican siempre sobre la base del monto_neto. Por ejemplo, para un cliente de más de 6 meses de antigüedad que compro 200.000 el monto total va a incluir el 10% de descuento por monto y el 4% de descuento por antigüedad.

monto_total = monto_neto - monto_neto*0.10 - monto_neto*0.04

Elabore un set de pruebas de caja negra razonable que permita testear con seguridad el método que calcula el descuento. Para cada caso de prueba basta con especificar el input del test, es decir, puede omitir señalar el output esperado.

Pauta Pregunta 1:

```ruby
class Salesman
  def initialize
    @commission = 0
  end

  def parent=(parent)
    @parent = parent
  end

  def number_of_ancestors
    if @parent.nil?
      1
    else
      1 + @parent.number_of_ancestors
    end
  end

  def complete_sale(amount)
    parts = 10
    parts += @parent.number_of_ancestors unless @parent.nil?

    @commission += sale_commission(amount) / parts * 10
    @parent.on_sale_completed(sale_commission(amount) / parts) unless @parent.nil?
  end

  def on_sale_completed(commission)
    @commission += commission
    @parent.on_sale_completed(commission) unless @parent.nil?
  end

end

class CarSalesman < Salesman
  def sale_commission(total_amount)
    commission = if total_amount < 20_000_000
                   total_amount * 0.02
                 else
                   total_amount * 0.05
                 end
    [commission, 2_000_000].min
  end
end

class BikeSalesman < Salesman
  def sale_commission(total_amount)
    total_amount * 0.03
  end
end
```

```ruby
class SalesmanFactory
  def initialize
    @salesmen = []
  end

  def create_salesman
    random = rand(2)
    if random == 0
      salesman = CarSalesman.new
    else
      salesman = BikeSalesman.new
    end

    unless @salesmen.empty?
      random = rand(@salesmen.length)
      salesman.parent= @salesmen[random]
    end

    @salesmen << salesman
    salesman
  end
end
```

Nota de Pauta:

Esta pregunta busca evaluar los patrones Factory y Observer. A pesar de que el segundo no se explicita, hay que notificar a los padrinos de las ventas de alguna forma (en esta pauta, se utiliza un simple atributo parent para referenciarlos).

Desglose de puntaje:

- Jerarquía Salesman, CarSalesman, BikeSalesman
  - 1 punto
- Implementación de SalesmanFactory y método create_salesman
  - 1 punto
- Cálculo de comisiones y actualización de atributo @commisions
  - 2 puntos
- Lógica para notificar a padrinos de las comisiones que les corresponden
  - 2 puntos

Pauta Pregunta 2:

Diagrama de Contexto:

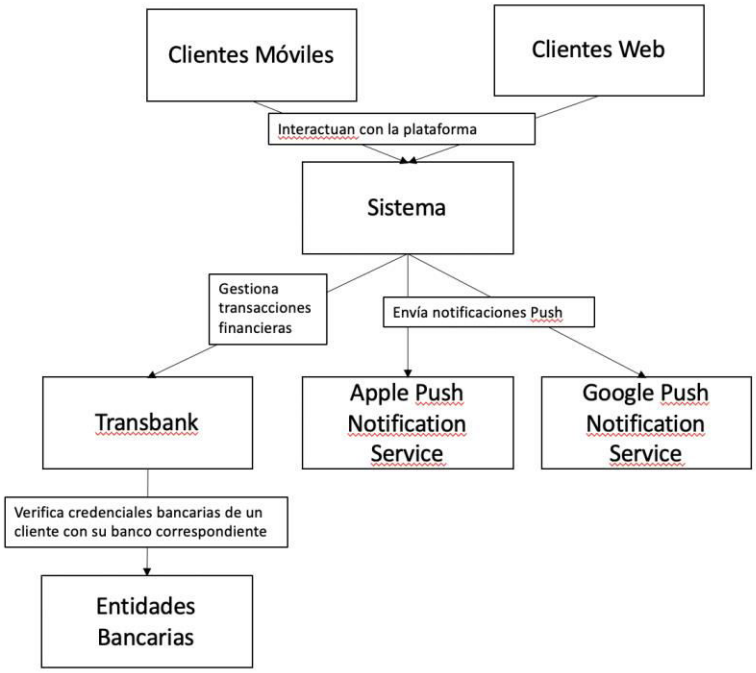


Diagrama de Contenedores:

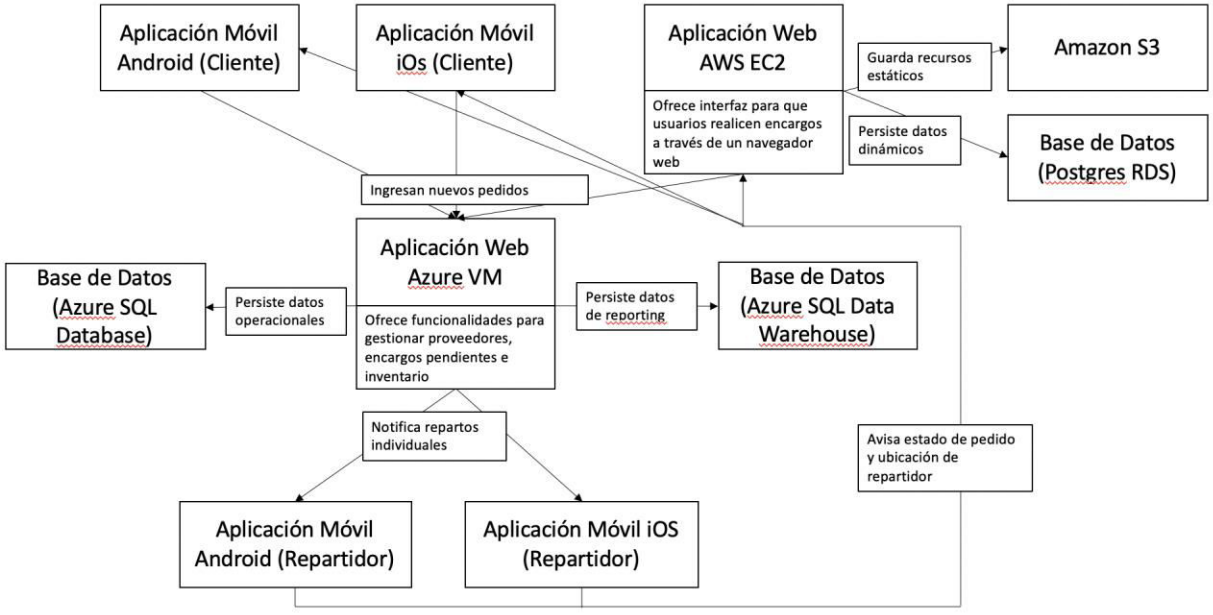
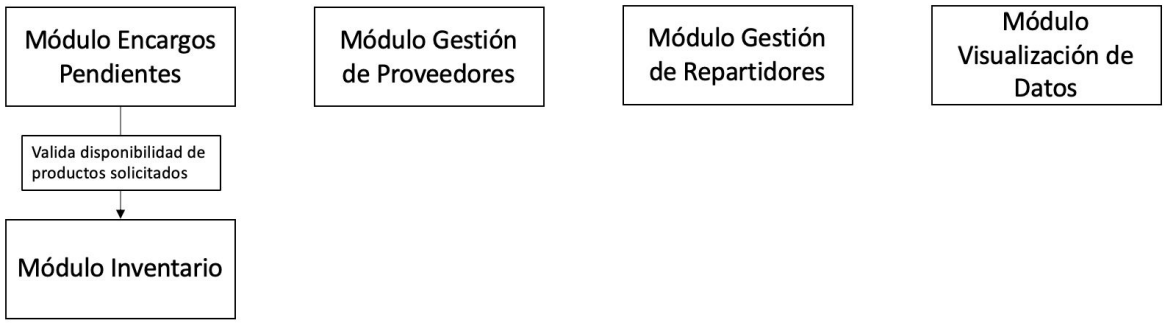


Diagrama de Componentes de la Aplicación Web en Azure VM:



Nota de Pauta:

- Cada uno de los elementos en cada uno de los tres diagramas debe consistir en un título más una breve descripción (en esta pauta solo se incluyen unas pocas descripciones por brevedad, pero el modelo C4 requiere un mayor nivel de detalle).
- Cada una de las dependencias debe también estar acompañada de una descripción.
- No se especifica el contenedor del cual se debe armar un diagrama de componentes, pero el único del cual se ofrecen suficientes detalles en el enunciado para armarlo es la aplicación web en azure VM, por lo que se espera que el alumno sea capaz de deducir por si solo lo anterior.
- Para el diagrama de contenedores, se acepta que el alumno agregue algunos contextos con los cuáles los primeros interactúan; para el diagrama de componentes, se acepta que el alumno agregue algunos contenedores con los cuales interactúan. Sin embargo, no es necesario para tener todo el puntaje.

Desglose de puntaje:

- Diagrama de Contexto
  - 2 puntos
- Diagrama de Contenedores
  - 3 puntos
- Diagrama de Componentes
  - 1 punto

Pauta Pregunta 3:

Horas totales por semana 88 horas
Considerando un 80% de horas efectivas de producción nos quedan 70.4 horas disponibles
Si hacemos sprints de dos semanas el total máximo por sprint es 140.8 hrs

Ordenando los relatos en orden descendente de prioridad tenemos

Relato	Esfuerzo (staff hours)		Prioridad
	Optimista	Pesimista	
2	18	30	alta
3	6	18	alta
5	16	36	alta
7	6	18	alta
11	18	32	alta
1	12	24	media
4	18	24	media
8	12	24	media
9	10	30	media
6	18	30	baja
10	12	30	baja
12	18	40	baja

Generalmente el esfuerzo real está mas cercano al valor pesimista. Podríamos usar esos valores (poniéndose en el peor caso) pero también podríamos agrega una columna con valores probables por ejemplo pesando los pesimistas el doble de los optimistas.

Nota de Pauta:
Lo importante aquí es:
1) Que se haga la consideración
2) Que se usen o bien los valores pesimistas o probables, nunca solo los optimistas

Trabajaremos con el peor caso aquí.

Dado que tenemos 140 horas disponibles por sprint podemos hacer
Sprint 1 : Relatos 2, 3, 5, 7, 11
Total: 134 horas
Sprint 2: Relatos 1, 4, 8, 9, 6, 10
Total: 162 horas
Dado que la planificación se hizo con los valores pesimistas es factible que pueda lograrse a pesar de sobrepasar las 140 horas. Si no es así el relato 10 quedará pendiente al igual que el 12 pero son ambos de baja prioridad.

b) Veamos ahora la planificación de los siguientes releases del producto:

Si mantenemos el mismo equipo con capacidad de 140 horas por sprint, podríamos tener el siguiente plan de entregas:

release 2: funcionalidades 2 y 3
Esfuerzo total 550 horas
Sprints: 4
Tiempo: dos meses

release 3: funcionalidades 4 y 5
Esfuerzo total: 540 horas
Sprints: 4
Tiempo: dos meses

release 4: funcionalidades 6
Esfuerzo total: 260
Sprints: 2
Tiempo: un mes

En resumen la planificación a nivel de producto queda

	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
Release 1 (funcionalidades básicas)						
Release 2 (agrega grupos 2 y 3)						
Release 3 (agrega grupos 4 y 5)						
Release 4 (agrega grupo 6)						

Nota de Pauta:
En la planificación del producto bien podría haberse elegido otra forma de agrupar los releases 2 al 4.

Pauta Pregunta 4:

Hay 3 dimensiones de posibles inputs: monto_neto, primera compra y antiguedad.

- monto_neto tiene 3 rangos relevantes
- primera compra solo 2 valores distintos
- antiguedad tiene tambien 3 rangos relevantes

El número de clases de equivalencia es entonces : 3 x 2 x 3 = 18

El número de puntos de borde es dos en la primera dimensión (10.000 y 100.000) y dos en la tercera dimensión (3 meses y 6 meses).

Un set de pruebas completo debería incluir

- un caso de pruebas para cada clase de equivalencia
- un caso de prueba para cada valor límite

El primer grupo ya sabemos que comprende 18 casos.

Para el segundo grupo, un análisis simple nos daría 4 casos adicionales (uno para cada valor límite). Sin embargo, cada valor límite debería ser probado con valores en los distintos rangos de las otras dimensiones. Esto implica que para un testeo exhaustivo cada valor generará 2 x 3 = 6 casos que en total nos da 24 casos adicionales.

En la tabla se muestran separados los primeros 18 casos y luego los 24 casos correspondientes a cada uno de los valores de borde.

1	monto < 10.000	primera	antiguedad < 3 meses	
2	monto < 10.000	primera	3≤ antiguedad < 6 meses	
3	monto < 10.000	primera	antiguedad ≥ 6 meses	
4	monto < 10.000	no primera	antiguedad < 3 meses	
5	monto < 10.000	no primera	3≤ antiguedad < 6 meses	
6	monto < 10.000	no primera	antiguedad ≥ 6 meses	
7	10.000 ≤ monto <100.000	primera	antiguedad < 3 meses	
8	10.000 ≤ monto <100.000	primera	3≤ antiguedad < 6 meses	
9	10.000 ≤ monto <100.000	primera	antiguedad ≥ 6 meses	
10	10.000 ≤ monto <100.000	no primera	antiguedad < 3 meses	
11	10.000 ≤ monto <100.000	no primera	3≤ antiguedad < 6 meses	
12	10.000 ≤ monto <100.000	no primera	antiguedad ≥ 6 meses	
13	monto ≥ 100.000	primera	antiguedad < 3 meses	
14	monto ≥ 100.000	primera	3≤ antiguedad < 6 meses	
15	monto ≥ 100.000	primera	antiguedad ≥ 6 meses	
16	monto ≥ 100.000	no primera	antiguedad < 3 meses	
17	monto ≥ 100.000	no primera	3≤ antiguedad < 6 meses	
18	monto ≥ 100.000	no primera	antiguedad ≥ 6 meses	
19	**monto = 10.000**	primera	antiguedad < 3 meses	
20	**monto = 10.000**	primera	3≤ antiguedad < 6 meses	
21	**monto = 10.000**	primera	antiguedad ≥ 6 meses	
22	**monto = 10.000**	no primera	antiguedad < 3 meses	
23	**monto = 10.000**	no primera	3≤ antiguedad < 6 meses	
24	**monto = 10.000**	no primera	antiguedad ≥ 6 meses	
25	**monto = 100.000**	primera	antiguedad < 3 meses	
26	**monto = 100.000**	primera	3≤ antiguedad < 6 meses	
27	**monto = 100.000**	primera	antiguedad ≥ 6 meses	
28	**monto = 100.000**	no primera	antiguedad < 3 meses	
29	**monto = 100.000**	no primera	3≤ antiguedad < 6 meses	
30	**monto = 100.000**	no primera	antiguedad ≥ 6 meses	
31	monto < 10.000	primera	**antiguedad = 3**	
32	monto < 10.000	no primera	**antiguedad = 3**	
33	10.000 ≤ monto <100.000	primera	**antiguedad = 3**	

34	10.000 ≤ monto <100.000	no primera	**antiguedad = 3**	
35	monto > 10.000	primera	**antiguedad = 3**	
36	monto > 10.000	no primera	**antiguedad = 3**	
37	monto < 10.000	primera	**antiguedad = 6**	
38	monto < 10.000	no primera	**antiguedad = 6**	
39	10.000 ≤ monto <100.000	primera	antiguedad = 6	
40	10.000 ≤ monto <100.000	no primera	antiguedad = 6	
41	monto > 10.000	primera	antiguedad = 6	
42	monto > 10.000	no primera	antiguedad = 6	

Nota de Pauta

Identificación correcta de las 18 clases de equivalencia: 2
Descripción de los 18 primeros casos de prueba (las clases): 2
-- Análisis de los bordes 4 casos adicionales 1 o bien
-- Análisis de los bordes 12 casos adicionales 2
Punto base