IIC 2143 Ingeniería de Software
Interrogación 2 - Semestre 1 /2019
Secciones 01 y 02

Responda cada pregunta en hoja separada
Entregue una hoja con su nombre para cada pregunta aunque sea en blanco
Tiempo: 2:00
Recuerden que están bajo el código de honor

Pregunta 1: (Modelos de Dominio)

La empresa tecnológica *EffectivePolls* acaba de comenzar a trabajar en un nuevo producto que se espera se transforme en el proyecto insigne de la compañía. El producto consiste en una herramienta web destinada a crear encuestas de satisfacción orientada a equipos de trabajo. En primer lugar, la herramienta debe permitir definir equipos de trabajo y sus integrantes, uno de los cuáles tendrá el rol de administrador. Por cada integrante, es necesario saber su nombre, mail y cargo. Una vez creada la encuesta por el administrador, se distribuye entre los miembros del equipo, se espera que respondan y luego se recopilan los resultados para ofrecerle al administrador del equipo una vista resumida de cuán satisfechos están los empleados de su empresa.

Todas las encuestas tienen un título, una descripción y un conjunto de preguntas asociadas. Todas las preguntas definen un texto, un orden (que define qué pregunta se muestra primero) y un flag que indique si acaso responderla es obligatorio o no. No obstante, las preguntas pueden admitir distintos tipos de respuesta: texto abierto, número entero en un intervalo específico o selección múltiple. Estas últimas definen varias alternativas. Se debe mantener flexibilidad para eventualmente agregar nuevos tipos de pregunta.

Una vez definida una encuesta, se debe poder crear una o múltiples instancias para responderla (la misma encuesta se podría repetir en varias instancias distintas para ver cómo a evolucionado la percepción de los empleados). Cada instancia tiene una fecha de inicio y una de término.

Una vez publicada una instancia de una encuesta, todos los miembros de un equipo de trabajo deben poder responderla: cada pregunta debe tener una respuesta asociada. Por último, la plataforma debe ser capaz de mostrar gráficos de tendencia con la información recopilada para mostrarle al administrador de equipo la evolución de la impresión de sus trabajadores. Por lo mismo, dado un set de respuestas de un miembro en particular, es imperativo ser capaz de determinar a qué instancia de encuesta corresponde, y por cada respuesta es necesario saber a qué pregunta está asociada.

Dibuje un modelo de dominio (usando la notación del diagrama de clases de UML) que ilustre las principales clases y relaciones (asociaciones y generalizaciones) entre clases que se desprenden del enunciado. En las clases, incluya todos los atributos que se desprendan del enunciado; para las relaciones, especifique etiquetas y multiplicidades; añada generalizaciones

donde sea pertinente. Especificar visibilidad y tipos de los atributos, además de navegabilidad de las relaciones es opcional.

Explicite todos los supuestos que consideró en la elaboración de su modelo. (6 puntos)

Pregunta 2: (Aspectos generales de ingeniería de software)

Responda las siguientes preguntas en no más de 5 líneas cada una:

a) En el contexto de casos de uso, explique los conceptos de flujo principal y flujos alternativos. (1 punto)

Considere el código Ruby a continuación.

```
module Main
  $file_name = ''
  $token = ''
  $result = []

  class Main
    def execute
      puts 'Input file name to read:'
      $file_name = gets.chomp

      puts 'Input token:'
      $token = gets.chomp

      parser = FileParser.new
      parser.parseFile
    end
  end

  class FileParser
    def parseFile
      reader = File.open($file_name, 'r')
      reader.lines.each_with_index do |line,
index|
        $result << index if line.include?
$token
      end
    end
  end

  main = Main.new
  main.execute
  puts $result.join(' - ')
end
```

b) Explique qué hace este código y evalúelo en términos de acoplamiento. (1 punto)

c) Explique el concepto de visibilidad aplicado a clases, métodos o atributos, y entregue dos ejemplos concretos de lo anterior. (1 punto)

d) En Scrum, una técnica para determinar los Story Points asociados a un relato de usuario es Planning Poker. ¿Por qué convendría usar esta práctica en vez de simplemente consensuar Story Points conversando? (1 punto)

e) Explique en qué consisten la ley de Parkinson y la ley de Goldratt. (1 punto)

f) Explique el concepto de deuda técnica (technical backlog). (1 punto)

Pregunta 3: (Diagrama de Estados)

Se pide modelar el comportamiento de un terminal de pago de Transbank (ver figura) como los que se usan en los restaurants, utilizando un diagrama de estados UML. A continuación, se describe el funcionamiento del dispositivo cuando el cliente solicita pagar.

El proceso se inicia cuando el mozo introduce el monto a pagar con las teclas numéricas y a continuación el tipo de tarjeta (débito con f1 o crédito con f2).

Si es débito el cliente debe presionar el botón verde (aceptar) y a continuación ingresar su clave de 4 dígitos. Si la clave es la correcta, la máquina imprime el recibo de la transacción para el mozo y luego, al presionar cualquier tecla, imprime una copia para el cliente.

Si se trata de crédito, después de que el cliente presiona la tecla de aceptar, se le pregunta si quiere pagar todo en una cuota (presionar 1) o en varias cuotas (presionar 2). Si el cliente presiona 1 la secuencia sigue igual que la tarjeta de débito. Si el cliente presiona 2 se le pregunta el número de cuotas y el cliente debe ingresar un número mayor o igual a 3 y menor o igual a 12. De ahí el proceso sigue igual que en los casos anteriores.

Observe que en la parte inferior de la máquina hay 3 teclas: la de mas a la izquierda de color rojo es para cancelar toda la operación y volver al comienzo. La del medio de color amarillo es para borrar la última tecla digitada y la de más a la derecha de color verde es la de aceptar (equivalente al enter) que es la que el cliente presiona cada vez que se le pide aceptar algo.

Puede hacer los supuestos que estime necesario siempre que sean razonables y no contradigan la información entregada. Deje todo claramente expresado por escrito. (6 puntos)

Pregunta 4: (Planificación)

Una pequeña empresa de desarrollo está considerando desarrollar una aplicación web para un cliente. La empresa dispone de un equipo de 3 personas full-time con contratos de trabajo de 44 horas a la semana.

El equipo ha estimado el esfuerzo de desarrollo en base a un levantamiento preliminar de relatos de usuario asignando valores de esfuerzo en horas-hombre en el mejor y el peor caso. El cliente ha revisado los relatos y ha asignado prioridades a dichas funcionalidades (baja, media y alta) en base a consideraciones del negocio.

Se pretende trabajar usando el modelo de desarrollo ágil Scrum con sprints de dos semanas de duración y el cliente ha dicho que necesita tener la aplicación en un plazo de 4 semanas.

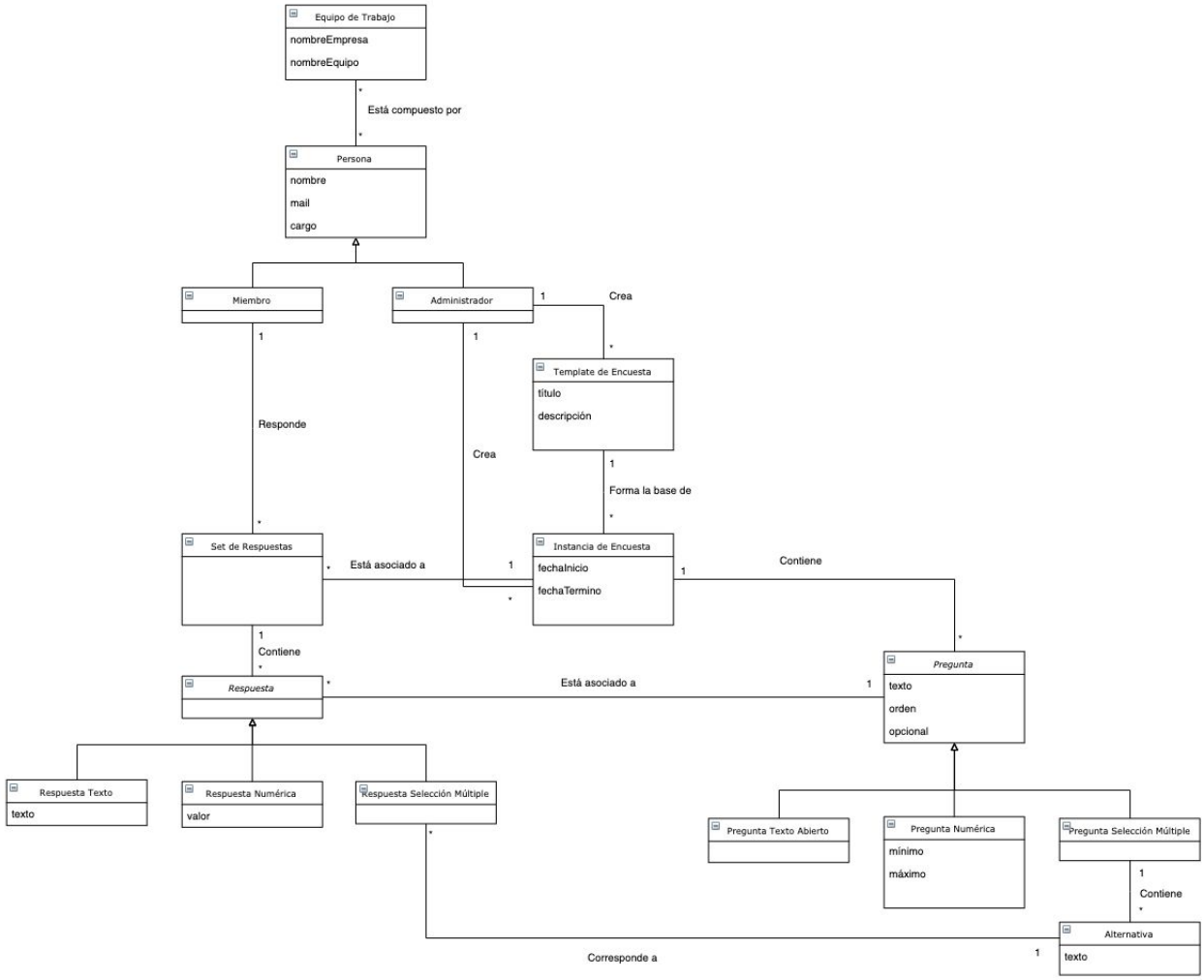
Relato	Prioridad	Best Case (horas hombre)	Worst Case (horas hombre)
1	baja	40	64
2	baja	48	64
3	media	64	80
4	media	24	48
5	baja	64	96
6	baja	16	48
7	alta	48	96
8	media	32	48
9	alta	64	80
10	baja	80	96

a) Haga una planificación para el release del producto que incluya todas las funcionalidades solicitadas. ¿Es posible la entrega en 4 semanas? Si no es así, ¿cuál es la extensión de tiempo que tendría que solicitar? (3 puntos)

b) Suponiendo que el cliente no acepta extender el plazo y el software debe estar listo en 4 semanas. Haga una planificación del desarrollo e indique la forma de negociar con el cliente. (3 puntos)

Si necesita hacer algunos supuestos déjelos claramente establecidos por escrito.

Pauta Pregunta 1:



Nota de Pauta: La solución propuesta para este ejercicio es solo una de múltiples posibles soluciones. Sin embargo, es importante verificar que todos los aspectos que figuran acá deben ser incluidos en la solución del alumno.

Se proponen varias generalizaciones posibles en esta solución. El alumno debe incorporar al menos una en su propuesta. Descontar 0.5 puntos en caso contrario.

Se debe distinguir claramente un Template para hacer encuestas y una instancia particular de esta. Descontar 1 punto en caso contrario.

Dada una respuesta, debe ser posible determinar la instancia a la que pertenece, al igual que la pregunta a la que está asociada. En este ejemplo se utiliza una clase intermedia “Set de Respuestas” pero no es necesaria. Descontar 1 punto en caso contrario.

Los distintos tipos de preguntas y respuestas deben estar representados como clases distintas. Si no se hace la distinción para las preguntas, descontar 0.5; si no se hace la distinción para respuestas, descontar 0.5.

Descontar 0.25 por cada etiqueta de relación omitida o que no tenga sentido.

Selección múltiple debe contener alternativas, y la respuesta a selección múltiple debe estar asociada a dicha alternativa. En caso contrario, descontar 0.5.

Por cada multiplicidad mal puesta u omitida, descontar 0.25. Se consideran aparte ambos extremos de la multiplicidad (si un alumno omite una multiplicidad para una relación, hay que descontar 0.5 debido a los dos extremos).

Todos los atributos de las clases definidos en las clases de este modelo deben figurar en alguna parte en la solución del alumno (aunque no necesariamente dentro de las mismas clases que salen acá). Descontar 0.25 por cada uno faltante. No obstante, los alumnos pueden agregar más si lo estiman pertinente.

La distinción entre Miembro y Administrador como clases aparte no es necesaria (pueden distinguirse con un atributo). En esta solución propuesta se asume que el administrador no responde la encuesta, pero es aceptable asumir que también la puede responder.

Otros aspectos del diagrama de clases UML como especificar tipo de asociación (dependencia, agregación, composición), tipo de los atributos u operaciones no se pedían en este ejercicio. Si el alumno las agregó de todas formas, revisar y descontar puntos a criterio del ayudante en caso de percibir un error.

Descuentos adicionales según otros errores que pudiera haber cometido el alumno quedan a criterio del corrector.

Pauta Pregunta 2:

a) En el contexto de casos de uso, explique los conceptos de flujo principal y flujos alternativos. (1 punto)

El flujo principal es la secuencia o interacción actor-sistema que describe el escenario se da en la mayor parte de los casos. Por ello se le suele llamar el "happy-path"

Los flujos alternativos describen variaciones al escenario más frecuente que se traducen en desvíos desde el flujo principal para retomarlo más adelante o simplemente para terminar en otro punto.

Nota de pauta: Asignar 0.5 puntos a cada definición.

b) Explique qué hace este código y evalúelo en términos de acoplamiento. (1 punto)

El código toma de entrada el nombre de un archivo y un token, luego retorna todas las líneas del archivo en que se encuentre dicho token. En términos de acoplamiento, el código muestra un fuerte grado de acoplamiento debido al uso de variables globales para compartir información (acoplamiento por variables globales). Esto en general es una característica indeseable.

Nota de pauta: Asignar 0.4 puntos a la descripción del programa y 0.6 puntos a la evaluación por acoplamiento.

c) Explique el concepto de visibilidad aplicado a clases, métodos o atributos, y entregue dos ejemplos concretos de lo anterior. (1 punto)

La visibilidad de clases, métodos o atributos corresponde a la posibilidad de ciertos lenguajes de programación de explicitar qué entidades pueden acceder a los miembros de otras entidades. Es un concepto fuertemente ligado a la encapsulación del paradigma orientado a objetos.

Los cuatro tipos de visibilidad son:

- Public: accesible por todos
- Protected: accesible por si mismo y clases derivadas
- Private: accesible solo por si mismo
- Package: accesible solo por entidades ubicadas en el mismo módulo o paquete (la semántica exacta de esta visibilidad varía según el lenguaje de programación).

d) En Scrum, una técnica para determinar los Story Points asociados a un relato de usuario es Planning Poker. ¿Por qué convendría usar esta práctica en vez de simplemente consensuar Story Points conversando? (1 punto)

Se recomienda utilizar planning poker para determinar los story points con el fin de fomentar la participación de todos los miembros del equipo y evitar el fenómeno de groupthinking.

e) Explique en qué consisten la ley de Parkinson y la ley de Goldratt. (1 punto)

Ley de Parkinson: El trabajo se expandirá hasta usar todo el tiempo disponible.

Ley de Goldratt: Si hay demasiado tiempo, se malgasta al comienzo del proyecto.

Nota de pauta: Asignar 0.5 puntos a cada definición.

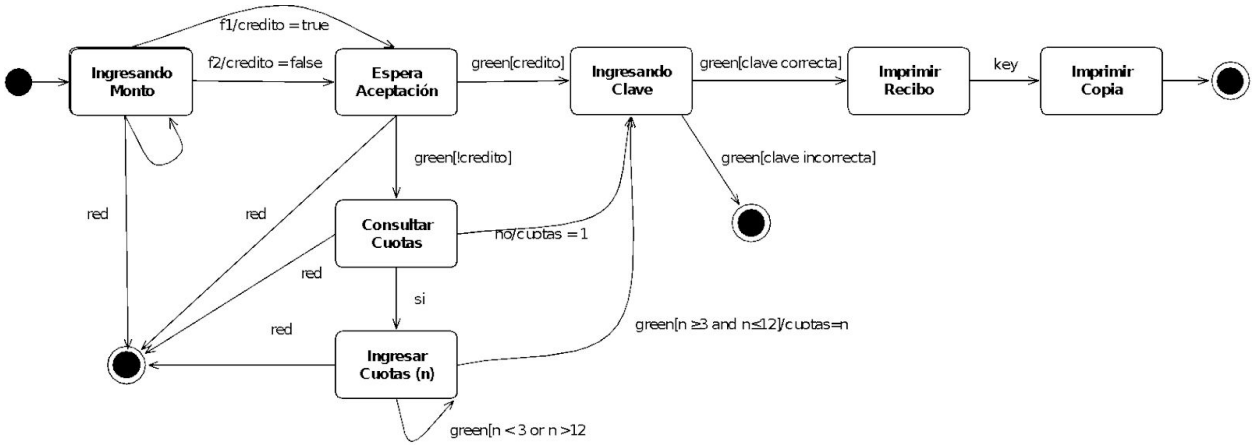
Si están invertidas asignar 0.5 a la pregunta

Si aparecen ambas, pero sin asignarse a los nombres asignar 0.5 a la pregunta

f) Explique el concepto de deuda técnica (technical backlog), cuándo puede ocurrir y sus posibles consecuencias. (1 punto)

Corresponde a las consecuencias de largo plazo de malas decisiones de diseño y prácticas de desarrollo. Ocurren con frecuencia cuando producto del retraso de un proyecto el equipo decide sacrificar criterios de calidad con el fin de acelerar el proceso de desarrollo. Lamentablemente, esto puede tener un efecto importante en la calidad de la solución final y el tiempo perdido en arreglar defectos producto de la deuda técnica puede ser superior al tiempo ahorrado por el equipo. En el peor de los casos, un cliente puede rechazar una solución si la deuda técnica es muy alta.

Pauta Pregunta 3:



Nota de pauta:

Los aspectos formales del diagrama correctos (2 puntos)

- los nombres en los rectángulos identifiquen claramente estados

- los eventos que ocasionan las transiciones deben ser las etiquetas de las flechas

- las condiciones (guardias) deben aparecer entre paréntesis cuadrados a continuación de la etiqueta del evento

- las acciones asociadas a la transición a continuación del evento o del guardia separado por un /

- uso de símbolos correctos para el comienzo y el fin

Modelo captura correctamente el problema (4 puntos)

Hay varias posibles variaciones como por ejemplo ...

Manejo de clave incorrecta regresa a ingresando clave (requiere una transición desde ingresando clave a fin con etiqueta roja)

Lo importante es chequear que captura la descripción del problema (que no hay cosas importantes que o no se tratan o se tratan mal)

Pauta Pregunta 4:

El esfuerzo total de desarrollo es:

- *best case*: 480 horas hombre
- *worst case*: 720 horas hombre

Contamos con un equipo de 3 personas full-time (contrato de 44 horas a la semana) lo que nos daría un total de 132 horas a la semana. Sin embargo, solo una parte del tiempo disponible de cada persona se podrá destinar al desarrollo propiamente tal porque hay otras tareas (reuniones diarias, ajuste de herramientas, tiempo libre, etc.). El tiempo efectivo para trabajar que pueden considerar los alumnos es entre un 50% y un 80% del valor anterior.

Nota de pauta: Si un alumno asume disponibilidad por sobre el 80%, descontar 2 puntos.

El análisis de este ejercicio puede proceder de 2 maneras. Se puede considerar el nivel de esfuerzo más probable en base al intervalo propuesto, o bien, se puede analizar por separado el best-case scenario y el worst-case scenario.

Alternativa 1:

a) Haga una planificación para el release del producto que incluya todas las funcionalidades solicitadas. ¿Es posible la entrega en 4 semanas? Si no es así, ¿cuál es la extensión de tiempo que tendría que solicitar?

Es bien sabido que cuando se estiman estos dos escenarios extremos probablemente el real termine más cerca del peor que del mejor.

Supondremos que el valor probable está a la mitad de distancia del peor que del mejor (ver gráfico)

Podríamos comenzar nuestra planificación basados en un esfuerzo total de 640 horas hombre de esfuerzo.

Supondremos que aproximadamente el 60% del tiempo puede ser usado efectivamente en desarrollo => 26 horas a la semana. Dado que son 3 desarrolladores tenemos una disponibilidad de 79 horas hombre a la semana, o bien 158 horas hombre por sprint.

El tiempo estimado de desarrollo es entonces: 640/79 = 8.1 semanas

Para realizar todo el proyecto necesitamos 4 sprints de 2 semanas. El proyecto no estará terminado antes de 8 semanas por lo que se debe renegociar una extensión de otras 4 semanas adicionales.

Nota de Pauta:
Se acepta un estimado probable entre 480 y 720 distinto a 640 siempre que sea mas cercano a 720 que a 480. Incluso se acepta trabajar en base a 720. Si es un promedio simple descontar 1 punto
Se acepta un tiempo efectivo distinto al 60% pero debería estar entre 50 y 80%. Un valor mayor que 80% debe castigarse con 1 punto.

b) Suponiendo que el cliente no acepta extender el plazo y el software debe estar listo en 4 semanas. Haga una planificación del desarrollo e indique la forma de negociar con el cliente.

Si solo se dispone de 1 mes de tiempo lo que se debe hacer es hacer una planificación basada en limitar el alcance del proyecto y desarrollar en orden de prioridades.

Usando la misma proporción de 1/3 - 2/3 para estimar esfuerzos probables tenemos los siguientes esfuerzos estimados:

Relato	Prioridad	Probable (horas hombre)
1	baja	56
2	baja	58
3	media	76
4	media	40
5	baja	84
6	baja	38
7	alta	80
8	media	42
9	alta	76
10	baja	90

Sprint 1: Relatos 7 y 9 de prioridad alta con un esfuerzo estimado de 156 horas hombre
Sprint 2: Relatos 3, 4 y 8 de prioridad media con un esfuerzo de 158 horas hombre

Podemos prometer que el software será entregado en un mes de plazo y que todas las funcionalidades de prioridad media y alta estarán implementadas. Las funcionalidades de prioridad baja se construirían en un segundo release del producto.

<u>Alternativa 2:</u>

a) Haga una planificación para el release del producto que incluya todas las funcionalidades solicitadas. ¿Es posible la entrega en 4 semanas? Si no es así, ¿cuál es la extensión de tiempo que tendría que solicitar?

Supondremos que aproximadamente el 60% del tiempo puede ser usado efectivamente en desarrollo => 26 horas a la semana. Dado que son 3 desarrolladores tenemos una disponibilidad de 79 horas hombre a la semana, o bien 158 horas hombre por sprint.

En el mejor de los casos, tardaríamos 480 horas hombre en completar el desarrollo. Necesitaríamos entonces 480/79 = 6.07 semanas.

En el peor de los casos, tardaríamos 720 horas hombre en completar el desarrollo. Necesitaríamos 720/79 = 9.11 semanas.

En ninguno de los casos es posible realizar la entrega en 4 semanas. Si queremos garantizar la entrega de todas las funcionalidades, necesitamos situarnos en el peor caso, en el cual necesitaríamos 5.11 semanas adicionales. Debemos traducir la cifra anterior a número de sprints. Esto correspondería a 3 sprints adicionales (ceil(5.11/2) = 3), por lo que específicamente necesitamos solicitarle al cliente 6 semanas adicionales para completar la solución.

b) Suponiendo que el cliente no acepta extender el plazo y el software debe estar listo en 4 semanas. Haga una planificación del desarrollo e indique la forma de negociar con el cliente.

Si solo se dispone de 1 mes de tiempo lo que se debe hacer es hacer una planificación basada en limitar el alcance del proyecto y desarrollar en orden de prioridades.

En el mejor de los casos, podemos ofrecer la siguiente planificación:

Sprint 1: Relatos 7, 9 (prioridad alta) y 8 (prioridad media) con esfuerzo estimado de 144 horas.
Sprint 2: Relatos 3, 4 (prioridad media) y 5 (prioridad baja) con esfuerzo estimado de 152 horas.

En el peor de los casos, podemos ofrecer la siguiente planificación:

Sprint 1: Relatos 9 (prioridad alta) y 3 (prioridad media) con esfuerzo estimado de 160 horas.
Sprint 2: Relatos 7 (prioridad alta) y 4 (prioridad media) con esfuerzo estimado de 144 horas.

Podemos prometer que el software será entregado en un mes de plazo, que todas las funcionalidades de prioridad alta y al menos 2 de las funcionalidades de prioridad media estarán implementadas. En el mejor de los casos, podríamos alcanzar a implementar la funcionalidad de prioridad media restante y una funcionalidad de prioridad baja adicional, pero no está garantizado. El resto de las funcionalidades se construirían en un segundo release del producto.

Nota de pauta: En la planificación propuesta por el alumno, es aceptable que el esfuerzo estimado de las tareas para un sprint sobrepase muy levemente la capacidad del equipo. Es aceptable el hacer una tarea de prioridad media antes de las de mayor prioridad siempre y cuando se garantice que en el release todas las tareas de mayor prioridad van a completarse

(puede ocurrir si con los supuestos de los alumnos el poner todas las tareas de prioridad alta en el primer sprint sobrepasa de sobremanera la capacidad del equipo y, para no perder el tiempo, resulta prudente poner una tarea de menor prioridad en el entre tanto). No es aceptable poner en la planificación alguna tarea de prioridad baja si podía remplazarse por otra de mayor prioridad sin exceder de sobremanera la capacidad del equipo.