Pregunta 1

Epica 1: Crear un cuestionario adecuado para ser aplicado a un puesto de ingeniería

Relato 1: Navegar sobre base de datos de preguntas para ser usadas en el cuestionario
Como gerente de RRHH necesito navegar las preguntas disponibles para ver de qué dispongo para armar la base del cuestionario

Relato 2: Buscar preguntas
Como gerente de RRHH necesito poder buscar preguntas disponibles de cierto tópico, buscar preguntas usadas en un cuestionario dado o preguntas que nunca han sido utilizadas

Relato 3: Enviar a revisión borrador de cuestionario
Como gerente de RRHH necesito poder enviar el borrador de un cuestionario a ser usado en un caso concreto al gerente de ingeniería de modo que el pueda revisarlo y modificarlo si lo desea.

Relato 4: Revisar borrador de cuestionario
Como gerente de Ingeniería necesito revisar el borrador, y enviar la versión revisada de vuelta al gerente de RRHH.

Relato 5: Construir cuestionario final
Como gerente de RRHH necesito poder construir la versión definitiva del cuestionario incorporando el feedback y posibles correcciones y nuevas preguntas del gerente de ingeniería para ser aplicado al posulante.

Epica 2: Mantener la base de datos de preguntas

Relato 6: Comprar nuevas preguntas sobre un tópico
Como gerente de RRHH necesito poder comprar nuevos sets de preguntas sobre tópicos determinados de modo de poder producir los cuestionarios mas ajustados a las necesidades.

Relato 7: Mapear habilidades con tópicos de preguntas
Como gerente de RRHH necesito mapear habilidades solicitadas con tópicos de los cuestionarios para poder seleccionar preguntas desde la base de datos de cuestionarios

Relato 8: Generar reporte del estado de la base de datos con preguntas
Como gerente de RRHH necesito poder generar un reporte mensual con el número de preguntas en la base de datos, número de preguntas utilizadas, número de nuevas preguntas compradas

Epica 3: Seleccionar a los candidatos a ser entrevistados

Relato 9: Aplicar cuestionario a postulante
Como gerente de RRHH necesito aplicar cuestionarios a los postulantes para enviar los resultados al gerente de Ingeniería

Relato 10: Enviar resultados de evaluación
Como gerente de RRHH necesito enviar los resultados de los tests junto con posibles comentarios adicionales al gerente de Ingeniería para que decida a quienes llamar a entrevista cara a cara.

Relato 11: Comunicar lista de preseleccionados
Como Gerente de Ingeniería necesito comunicar a gerente de RRHH la lista de candidatos que deberán ser llamados a entrevista para poder pasar a la última etapa definitiva cara a cara

Relato 12: Concertar citas para entrevistas cara a cara
Como gerente de RRHH necesito poder concertar citas de cada uno de los postulantes preseleccionados con el gerente de Ingeniería para poder tomar una decisión final

Relatos no Asociados a Epicas

Relato 13: Generar reporte de cuestionarios construidos y aplicados
Como gerente de RRHH necesito poder generar un reporte mensual de número de cuestionarios construidos, número suministrado a postulantes, y puntajes obtenidos

Notas de Corrección

- Deberían haber reconocido por lo menos 10 de estos 13 relatos. Ojo que lo que se busca es la funcionalidad no el título
- Deberían haber reconocido al menos 2 de las 3 epicas
- No necesariamente las agrupaciones de relatos en épicas, debe ser ésta
- Descontar por malos títulos de los relatos
    - no comienza con un verbo
    - poco claro en cuanto al objetivo
- Descontar por no seguir el patrón estándar: "yo como X necesito Y de modo que Z"

Pregunta 2

Disponemos de 25 staff hours por sprint (los dos ingenieros)

Es razonable suponer que un 20% no será usado en PBIs de desarrollo sino en otras tareas. Consideraremos entonces 20 horas hombre por sprint

Dado que se ha estimado trabajo por 120 horas el número de sprints será 8 (siempre redondear hacia arriba)

No se entrega una priorización de los relatos, pero en base a la descripción es razonable suponer que los relatos 1, 3 y 4 son esenciales (must have). El relato 2 es menos importante porque por lo general el técnico sabe en los casos que más se repiten. El relato 5 definitivamente es el menos prioritario ya que no es esencial para llevar a cabo la tarea.

Tenemos entonces la lista priorizada siguiente

Prioridad	Relato	Horas hombre
1	1	10
2	3	20
3	4	10
4	2	40
5	5	40

Sprint:1
Relatos: 1 y 3
Necesario: 20
Disponible: 20

Sprint: 2
Relato: 2
Necesario: 20
Disponible: 20

Sprints: 3 y 4
Relato: 2
Necesario: 40
Disponible: 40

Sprints: 5 y 6
Relato: 5
Necesario: 40
Disponible: 40

a) De acuerdo a la planificación anterior, el release puede ser comprometido para 6 semanas e incluirá <u>todas</u> las funcionalidades

b) Si se debe entregar en 4 semanas se puede comprometer la entrega pero sin que incluya el relato 5 (ajuste por alcance). Dado que ese relato no tiene demasiada relevancia en el negocio es muy posible que no haya ningún problema en esta propuesta.

Notas de Corrección

- Deberían haber sido capaces de priorizar los relatos en base a la descripción. Si no los ordenaron o no hicieron esta consideración debería haber un descuento importante
- Deberían haber estimado la cantidad de horas hombre por sprint con un número algo menor que 25 (que es el monto bruto). Si en lugar de 20 dejan un número algo menos o mayor igual puede aceptarse
- La planificación a) debe considerar la realización de todos los relatos en orden de prioridad considerando la disponibilidad del sprint
- Debe indicarse en a) el número de semanas en que se puede comprometer el release
- La planificación b) debería dejar afuera lo que no se puede hacer en 4 sprints (se espera que sea lo de menor prioridad)

Pregunta 3

a) (6 pts) El primer hash, hash1, no funciona porque está definido con la notación tipo JSON. Esta notación convertirá las keys a symbols. Como no existe una conversión de un entero o un hash a symbol, el interpretador no logra ejecutar la declaración. Por otro lado, el segundo hash funciona porque está declarado con la notación tipo rocket. Con esta notación es posible guardar cómo key de un objeto cualquier otro objeto, entre ellos un entero y un hash como en este caso en particular. La notación JSON es más cómoda y ahorra espacio en memoria, mientras que con la notación rocket se pueden guardar objetos más complejos.

2 pts: Por decir cuál funciona.
2 pts: Por decir la razón correcta.
2 pts: Por especificar las diferencias entre ambas notaciones.

b) (6 pts) El cliente, es decir, quien toma el rol de cliente en el patrón cliente servidor de la web, es el que hace la request. Esta request llega al servidor para ser procesada. Si el servidor encuentra algún error con esta request enviará un código del tipo 400s, que significa que hubo un error al crear la request. Quien cumple el rol del cliente en la web no es el usuario final, sino el programa que corre y crea una request. Puede ser el browser, una aplicación web del lado del cliente o una aplicación móvil. Es esa aplicación la responsable de manejar este tipo de errores. Un usuario cualquiera de la web no sabe qué hacer cuando se le muestra un código de error, muchas veces no sabe qué significa ese código y qué acciones tomar al respecto. Es por eso que es el cliente el que debe manejarlo y darle un mensaje entendible al usuario sobre qué ocurrió.

3 pts: Explicar de forma correcta la diferencia entre cliente y usuario.
3 pts: Explicar la razón por la que el código no debe mostrarse al usuario.

c) (8 pts) Con respecto al caso de los accidentes del Boeing 737 se discutió:
Fallas de ingeniería:
- No se construyó un sistema confiable. El software MCAS para nivelar y bajar la punta de la nariz del avión se activaba en momentos que no debía hacerlo debido a malas lecturas del sensor. Estas posibles fallas se debieron prever y construir un sistema de software más robusto.
- No se respetó el proceso de testing, en especial con los vuelos de test en que pilotos señalaron problemas al intentar subir la nariz del avión. Un buen proceso de testing debió recoger este feedback y trabajar para mejorarlo.
- Se comprometió la calidad de software por presiones externas. Ya sea propios de la empresa como distintos departamentos (operaciones, marketing, gerencia) como externos a ella, como la presión que ejercían los conflictos de interés entre Boeing y la FA.

No hubo un sólo culpable, sino que todos participaron en no corregir la situación hasta que acabara con un accidente grave. En particular hay culpabilidad del lado del desarrollador porque no pudo garantizar la calidad de software.

El testing es uno de los procesos que nos permiten garantizar la calidad de software. Siguiendo un buen proceso de testing podemos encontrar fallas y vulnerabilidades a los sistemas. En este caso podrían haber encontrado los errores con los sensores del MCAS, o mejorar la usabilidad para los pilotos. También pudo reforzar procesos como los necesarios para que los pilotos se capacitaran en esta nueva tecnología.

3 puntos: 1 punto por cada falla de ingeniería bien identificada.
2 puntos: Por mencionar quienes son los culpables.
3 puntos: Por relacionar correctamente el testing dentro del contexto de estos accidentes.

Pregunta 4

(30ptos).

- (2 pts) Entregar un archivo index.html.erb o el código de la vista.
- (4 pts) Agregar HTML y ruby a la vista index mostrando todos los campos de la base de datos.
- (2 pts) Entregar un archivo new.html.erb o el código de la vista.
- (5 pts) Crear formulario con todos los campos necesarios para los campos en la base de datos. Usa correctamente los campos para fecha, entero, y string.
- (2 pts) Crea clase controlador.
- (1 pt) La clase controlador tiene el método index bien programado.
- (1 pt) La clase controlador tiene el método new bien programado.
- (2 pts) La clase controlador tiene el método create bien programado.
- (2 pts) Utiliza un método post_param para pasar los parámetros de forma segura.
- (2 pts) Entrega archivo de rutas.
- (2 pts) El archivo de rutas tiene definida una ruta GET para el método index.
- (2 pts) El archivo de rutas tiene definida una ruta GET para el método new.
- (2 pts) El archivo de rutas tiene definida una ruta POST para el método create.

Posible solución:

index.html.erb

```
<h1>Publicaciones</h1>
 <%= link_to 'Crear Publicación', new_post_path %>
 <br>
 <% @posts.each do |post| %>
 <div>
   <h2><%= post.title %></h2>
   <p><%= post.picture_date.strftime("%d/%m/%Y")%></p>
   <%= image_tag(post.picture_url) %>
   <p><%= post.description%></p>
   <p>Total de objetos: <%= post.objects_total %></p>
   <p>Longitud: <%= post.longitude %> - Latitud: <%= post.latitude %></p>
   <p>DEC: <%= post.dec %> - AR: <%= post.ar %></p>
```

```
   <p>Autor: <%= post.author_name %> <%= post.author_last_name %> - <%=
post.author_email %></p>
 </div>
 <hr />
 <% end %>
</div>
```

new.html.erb

```
<h1>Nueva publicación</h1>
<%= form_with model: @post, local: true do |form| %>
 <h3> Sobre la imagen </h3>
 <p>
   Titulo:
   <br>
   <%= form.text_field :title %>
 </p>
 <p>
   Descripción:
   <br>
   <%= form.text_field :description %>
 </p>
 <p>
   Número de objetos:
   <br>
   <%= form.number_field :objects_total %>
 </p>
 <p>
   Url de imagen:
   <br>
   <%= form.url_field :picture_url %>
 </p>
 <p>
   Fecha de la imagen:
   <br>
   <%= form.datetime_field :picture_date %>
 </p>
 <p>
   Latitud:
   <br>
   <%= form.text_field :latitude %>
 </p>
 <p>
   Longitud:
   <br>
```

```
    <%= form.text_field :longitude %>
  </p>
  <p>
    DEC:
    <br>
    <%= form.text_field :dec %>
  </p>
  <p>
    AR:
    <br>
    <%= form.text_field :ar %>
  </p>
   <h3> Autor: </h3>
  <p>
    Nombre:
    <br>
    <%= form.text_field :author_name %>
  </p>
  <p>
    Apellido:
    <br>
    <%= form.text_field :author_last_name %>
  </p>
  <p>
    Email:
    <br>
    <%= form.email_field :author_email %>
  </p>
  <p>
    <%= form.submit 'Crear Publicación' %>
  </p>
<% end %>

<%= link_to 'Volver', posts_path %>
```

routes.rb

```
Rails.application.routes.draw do
  # For details on the DSL available within this file, see
http://guides.rubyonrails.org/routing.html
  resources :posts, only: [:index, :new, :create]
end
```

posts_controllers.rb

```ruby
class PostsController < ApplicationController
 def index
   @posts = Post.all()
 end

 def new
   @post = Post.new
 end

 def create
   @post = Post.new(post_params)
   if @post.save
     redirect_to posts_path
   else
     render 'new'
   end
 end

 def post_params
   params.require(:post).permit(
     :picture_url,
     :title,
     :description,
     :latitude,
     :longitude,
     :objects_total,
     :picture_date,
     :dec,
     :ar,
     :author_name,
     :author_last_name,
     :author_email
   )
 end
end
```