PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2513 – Tecnologías y Aplicaciones Web (II/2015)

Profesor: Raúl Montes

Interrogación 1

10 de septiembre de 2015

Ejercicio 1 (35 %): Preguntas conceptuales

Responde en forma precisa y concisa las siguientes preguntas:

1. (1 pto) Explica el concepto de *convention over configuration* y da 3 ejemplos de cómo se aplica en Rails.
2. (1 pto) ¿Qué es *MVC* y cómo se aplica en Rails?
3. (1 pto) Son las 4 A.M. y con tanto estudio para esta prueba te pones a soñar que te conviertes en ¡un computador que actúa de servidor Web! Tienes instalada una aplicación Rails que permite revisar un listado de especies de escarabajo y luego, mediante un link, ver una página con el detalle de un escarabajo en particular. Todo estaba tranquilo hasta que, de pronto, un usuario se dispone a cargar esa lista y luego hacer clic en el primer escarabajo. En ese momento, todo pasa a verse en "cámara lenta" en tu sueño. Describe, en detalle, desde tú perspectiva como servidor Web, qué sucede con esta interacción del usuario, paso a paso. Mantente sobre el nivel TCP/IP (Internet), pero sé detallista, incluyendo lo que sucede "dentro" de la aplicación Rails.
4. (1 pto) El gran logro inicial de la Web fue darnos la posibilidad de dejar disponibles en Internet documentos estáticos "*linkeables*" entre sí de manera que cualquier computador conectado a esta red pudiese obtenerlo y desplegarlo al usuario final. Hoy en día, usando básicamente la misma infraestructura, podemos dejar disponibles a todo el mundo verdaderas aplicaciones, que ya no tienen mucho que enviadarle a sus contrapartes "nativas" (o "de escritorio"). Pero esto no sucedió de la noche a la mañana... explica cómo ocurrió esta evolución, incluyendo una mención y pequeña explicación de los estándares y tecnologías involucrados.
5. (1 pto) En el contexto de Ruby, explica los usos que tiene el concepto de *módulo* (*module*). Da un ejemplo de cada uno con código.
6. (1 pto) En Ruby, ¿qué es un *símbolo* y cuál es su importancia?

Ejercicio 2 (35 %): Demuestra que el Ruby es tu piedra preciosa favorita

Las siguientes dos preguntas sobre Ruby esperan que saques provecho de sus características diferenciadoras, no sólo que

1. (3 ptos) Escribe una clase `MathSequence`, cuyo constructor recibe el nombre de una serie numérica y además tiene un método, `traverse`, que dado un número `n` permite recorrer los primeros n números de esa serie y ejecutar código para cada número y su índice. Las series soportadas son las de *Fibonacci* (1, 1, 2, 3, 5, ...), *Cuadrados* (1, 4, 9, 16, ...) y los números triangulares (1, 3, 6, 10, ... $triangle_n = \frac{n \times (n+1)}{2}$). Tu clase debe poder usarse de la siguiente manera:

```
seq = MathSequence.new(:fibonacci)
seq.traverse(100) do |i, number_i|
  puts "El␣#{i}-ésimo␣número␣es␣#{number_i}"
end
```

2. (3 puntos) Crea una clase `Person` que tenga atributos para nombre, apellido, edad y sexo; todos ellos se especifican sólo al momentos de construir un objecto, pero sus valores pueden leerse de manera pública.
   Luego crea una clase `Hero`, que hereda de `Person`, y que tiene las siguientes características adicionales:

   - un atributo que representa un listado de "superpoderes" (cada uno representado simplemente como un símbolo). Un `Hero` es creado, inicialmente, sin superpoderes.
   - un atributo que represente su nombre de súper héroe (se especifica, junto con los atributos de persona, al momento de construcción).
   - un método para agregar un nuevo superpoder.
   - un método que recibe un superpoder y que, si está disponible en su lista de superpoderes, lo utiliza (mostrando algo en consola simplemente).
     Pero si el superpoder no está disponible, debe lanzar la excepción `IDontHaveThatPowerException`.
   - que dos héroes se puedan sumar (`hero1 + hero2`) y que el resultado de esa suma sea un nuevo súper héroe que tenga una combinación de ambos poderes y cuyo nombre sea una combinación de ambos (lo que pasa con los atributos de persona queda a tu criterio).

Ejercicio 3 (30 %): Partiendo con Rails

1. (3 puntos) Explica de manera corta cómo funciona el sistema de *migraciones* de Rails y escribe una migración para representar a la entidad Universidad, que tiene los atributos nombre, url (de su sitio web), si es o no estatal y la cantidad de alumnos que posee. Detalla cómo quedará la tabla en la base de datos luego de ejecutarla.

2. (3 puntos) Escribe una clase ActiveRecord para trabajar con la tabla anteriormente creada, incluyendo validaciones para:

   - nombre: no puede estar en blanco, y debe al menos tener 3 caracteres.
   - url: debe ser de la forma `www.<letras o números>.<entre 2 y 5 letras>`.