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

IIC2513 – Tecnologías y Aplicaciones Web (II/2018)

Profesor: Raúl Montes

Examen

29 de noviembre de 2018

Pregunta 1 (25%): Conceptos

Responde en forma precisa y concisa las siguientes preguntas (**6 preguntas**, 1 punto cada una):

1. ¿Cuál es la diferencia entre los valores `relative`, `absolute` y `fixed` de la propiedad `position` de CSS?
2. ¿Dónde y por qué razón suele ser recomendable ubicar elementos `link` y `script` dentro del documento HTML?
3. ¿Callbacks o promesas? ¿Por qué?
4. En el contexto del client side ¿Qué significa tener un "layout fluido" (o líquido)? ¿Para qué sirven los "breakpoints"?
5. ¿Qué características tiene HTTP que facilitan el que puedas tener una arquitectura de servidores con un "load balancer" que recibe todos los requests y luego los reparte entre N nodos que corren tu aplicación y generan la respuesta a enviar al cliente?
6. Te encuentras un sitio web con un formulario para publicar mensajes, y te das cuenta de que **todo** lo que escribes ahí, caracter por caracter, sin escapar caracteres de ninguna forma, se muestra directamente en un feed visible por muchos otros usuarios. ¿Qué riesgos corren los usuarios de esa aplicación? Sé específico, basándote en ejemplos concretos.

Pregunta 2 (25%): Sé un buen consejero (4 items, 1.5 puntos cada uno)

A continuación verás una serie de situaciones que gente con menos experiencia que tú en el mundo Web podría tener y no saber cómo resolver. Tu misión es describir cómo se pueden solucionar estos problemas, suponiendo que tu destinatario no es completamente ignorante respecto a estos temas (ha estado aprendiendo de koa, por ejemplo) pero no es experto en ningún caso, por lo que debes explicarle detalladamente y con algunas "manzanitas y peritas" (puedes suponer un nivel similar al tuyo). No es requisito que escribas código (a menos que se pida explícitamente), pero si crees que te facilita la explicación y puedes ayudar a esta persona de mejor manera, siéntete libre de incluir código o pseudo-código. Además, puedes suponer que todos estos casos son usuarios que están utilizando el mismo template de aplicación que tú usaste durante el semestre.

1. Tengo un sitio de adopción de mascotas y tengo dos modelos, Pet (mascota) y User, y quiero que cada usuario pueda adoptar a una mascota, guardando información como la fecha en que sucedió. ¿Cómo tengo que cambiar mi **modelo** para lograrlo?

2. Tengo 5 funciones asíncronas (`fa`, `fb`, `fc`, `fd` y `fe`) que retornan promesas por un número, pero con las siguientes restricciones:

   - `fb` necesita recibir como argumento el número de `fa` (el número de la promesa resuelta, no la promesa)
   - si `fc` entrega un número par, entonces `fe` necesita recibir el número de `fd`, pero si no es así no necesita recibir ningún argumento

   Lo que necesito es una función que me entregue la suma de `fb`, `fd` y `fe` considerando las restricciones anteriores y que debe entregar ese resultado lo más rápido que sea posible. ¿Puedes escribir un trozo de código que logre esto?

3. En varios *route handlers* de mi aplicación necesito realizar una operación que puede fallar con el mismo error, un error de la clase `OopsError`. Cada vez que esto sucede necesito reaccionar de la misma forma, mostrando una página con un texto "Oops, I did it again". ¿Cómo puedo hacer esto sin tener que repetir tanto código?

4. Tengo un token que me da permisos para utilizar una API pagada que me entrega datos que muestro junto a otra información importante en una página de mi aplicación. Sin embargo, estos requests a esta API están tomando bastante tiempo, lo que está causando problemas pues ha hecho que la página completa sea muy lenta. ¿Cómo puedo solucionar esto fácilmente, de manera tal que pueda mostrar al menos el resto de la información que no necesita de esta API y luego cuando tenga los datos de la API agregarlos a esa página?

Pregunta 3 (25 %): Red social

En la lejana Tierra Media surgió un nuevo emprendimiento tecnológico: una red social. Sin embargo, como las distintas razas no están muy bien integradas aún, decidieron partir teniendo una faceta diferente de la red social para cada una de las razas. Inicialmente lanzarán esto entre elfos, enanos, orcos, hombres y, por supuesto, también hobbits.

Están desarrolando una aplicación con koa, y tienen el siguiente requerimiento que no saben cómo solucionar. Necesitan que cuando llegas a la aplicación, independiente de la URL con la que llegaste, si es que no saben de qué raza eres se te muestre una pantalla de selección de raza. Una vez selecciones una de las razas, la aplicación te llevará de nuevo a la URL original con la que llegaste a la aplicación.

Además, cuando la aplicación sabe ya de qué raza es el usuario, deberá contar con esta información en cada uno de los *route handlers*, pues en muchos de ellos necesitarán personalizar cierta información de acuerdo a la raza.

Explica cómo resolverías este problema, y luego implementa (sí, significa escribir código) tu solución. En particular, la(s) ruta(s) y página (EJS, no es necesario CSS) asociada a la selección de entre las cinco razas, y lo necesario para el funcionamiento descrito anteriormente (redirección a esta selección de raza cuando sea necesario y exponer la raza conocida a los *route handlers*).

Pregunta 4 (25 %): La mejor tienda online de la galaxia

Hace mucho tiempo, en una galaxia muy muy lejana, la República ha logrado vencer al Imperio Galáctico, trayendo paz a toda la galaxia. Sin embargo, para Droids & Clones Corporation (también conocida como DCC), que se benefició tanto de los bélicos tiempos anteriores, esta nueva realidad los obliga a pivotear en su negocio: crearán una tienda online de manualidades hechas por sus ejércitos de clones y droides.

Para estar al día con las tecnologías, deciden crear una *single page application*, y te piden ayuda pues sus informantes les comentaron que has estado aprendiendo bastante sobre React y sobre APIs. Concretamente, necesitan que los ayudes con:

1. (3 puntos) un API endpoint para obtener una manualidad, incluyendo datos de su autor y de la categoría a la que pertenece. Una restricción importante aquí es que por estándares de la galaxia necesitan que esta API sea RESTful.
2. (3 puntos) un componente React que se encargue de mostrar una manualidad. Éste recibirá un id de manualidad y deberá cargarla con el endpoint del punto anterior, esperando con un "loading" mientras eso sucede. Lo que necesita mostrar es el `title`, `photoUrl` (como una imagen), `description`, `createdAt` (fecha en que se agregó esta manualidad), el nombre de la categoría a la que pertenece y algunos datos del autor como `name`, `type` (Droid o Clone) y `age` (su edad en años).

El modelo (ya existente) que necesitarás utilizar es:

- `Craft` (manualidad) con atributos `id`, `title`, `description`, `photoUrl`, `internalCode` (un código sólo para uso interno, que usuarios no deberían conocer), `purchasesCount`, `updatedAt`, `createdAt` más los atributos de las relaciones `belongsTo` a Crafter (el autor) y Category
- `Crafter` (autor de manualidades) con atributos `id`, `name`, `age`, `type` (droid o clone), `ssn` (Soldier Security Number, es como el identificador del clon o droide, para uso interno)
- `Category` (categoría) con atributos `id` y `name`

Importante: En esta pregunta se evaluará tanto tu resultado como la forma de lograrlo (buenas prácticas vistas durante el semestre).