Aplicaciones Web
Año 2017/2018
Grado en Ing. Software
Fdi - UCM

Ejercicio Entregable 5

Implementación de la capa de acceso a datos

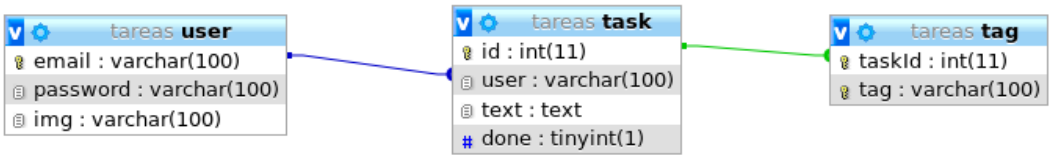
El ejercicio de esta hoja es entregable y puede realizarse en el laboratorio. Los ejercicios entregables realizados a lo largo del curso computan el 10 % de la nota final, pero su entrega no es obligatoria para aprobar la asignatura. La entrega se realiza por parejas.

Fecha límite de entrega: 23 de noviembre de 2017.

La entrega se realizará a través del Campus Virtual.

En este ejercicio vamos a continuar con la implementación de nuestra aplicación de gestión de tareas. Para ello abordaremos el acceso a la base de datos relacional que las almacenará.

Partimos del siguiente esquema relacional:



Por un lado tenemos la información de los usuarios. Para cada usuario almacenamos su dirección de correo electrónico (que es su identificador), una contraseña, y un atributo `img` con el nombre del fichero que contendrá la imagen de perfil, o `NULL` en caso de no tener imagen. De momento no te preocupes por los detalles de almacenamiento de la imagen; tan solo debes conocer que existe una columna `img` dentro de la tabla de usuarios.

Cada usuario tiene asociado un conjunto de tareas, que se representan en la tabla `task`. Cada tarea dispone de un texto, un bit `done` que indica si la tarea ha sido realizada, y una lista de *tags* que, al ser un atributo multivalorado, se almacenan en una tabla aparte (`tag`).

En este ejercicio se proporciona, además del fichero `.sql` que genera las tablas mostradas en la figura anterior, la implementación de dos clases Javascript: `DAOUsers` y `DAOTasks`. La primera clase implementa la funcionalidad relacionada con la gestión de usuarios en la BD, y la segunda implementa la gestión de tareas. Algunos métodos ya están implementados, salvo tres de ellos, que deberás implementar tú.

Las constructoras de las clases `DAOUsers` y `DAOTasks` reciben un pool de conexiones y lo almacenan en un atributo de la clase. Todas las operaciones a implementar deben obtener una conexión del pool, ejecutar una o varias consultas **paramétricas** y devolver la conexión al pool.

Todos los métodos a implementar son asíncronos. Esto implica que recibirán, como último parámetro, una función callback a la que pasarán dos argumentos. El primer argumento es un objeto de la clase `Error` que contendrá el error producido al ejecutar la consulta o modificación en la BD, o `null` en caso de no producirse ningún error. El segundo parámetro es el resultado de la operación realizada, o `undefined` si se ha producido algún error. Los métodos a implementar son:

- `DAOUsers.isUserCorrect(email, password, callback))`: Determina si existe un usuario en la BD con el `email` y `password` dados. La función callback, además del objeto error correspondiente, recibirá un valor booleano. Ejemplo de uso:

```
let daoUser = ...
daoUser.isUserCorrect("usuario@ucm.es", "mipass", (err, result) => {
    if (err) {
        console.error(err);
    } else if (result) {
        console.log("Usuario y contraseña correctos");
    } else {
        console.log("Usuario y/o contraseña incorrectos");
    }
});
```

- `DAOTasks.getAllTasks(email, callback)`: Devuelve todas las tareas asociadas a un determinado usuario de la BD. Se han de recuperar tanto las tareas como los *tags* asociados a cada una de ellas. Presta atención al problema de las $n+1$ consultas. Debes realizar este ejercicio utilizando una única consulta que relacione las tablas `tasks` y `tag`, y reconstruir el resultado a partir de esta consulta. El resultado de esta operación ha de ser un array de tareas, siendo cada una de ellas un objeto con cuatro propiedades: `id`, `text`, `done` y `tags`. La última de ellas es una lista de `string` con las etiquetas asociada a las tarea. Ejemplo de uso:

```
let daoTask = ...
daoTask.getAllTasks("usuario@ucm.es", (err, tasks) => {
    if (err) {
        console.error(err);
    } else {
        console.log(tasks);
    }
});
```

- `DAOTasks.insertTask(email, task, callback)`: Inserta una tarea dada en la BD, asociada al usuario cuyo identificador es `email`. La tarea recibida como parámetro es un objeto que contiene tres atributos: `text`, `done` y `tags`. Puedes implementar este método mediante dos consultas en la BD: una para insertar en la tabla `task`, y otra para insertar las etiquetas en la tabla `tag`. Para construir esta última consulta ten en cuenta que puedes insertar simultáneamente varias filas en la BD mediante una única sentencia `INSERT`:

```
INSERT INTO tag(taskId, tag) VALUES (?, ?), (?, ?), (?, ?), ...
```

  La función `callback` recibirá en este caso, un único parámetro con el objeto `Error`, en caso de producirse. Ejemplo:

```
daoTask.insertTask("usuario@ucm.es", newTask, (err) => {
    if (err) {
        console.error(err);
    } else {
        console.log("Elemento insertado correctamente");
    }
    pool.end();
});
```