Entrega5_BBDD

Versión: 3 de marzo de 2020

Objetivos

- Afianzar los conocimientos obtenidos sobre el desarrollo de programas interactivos con Node.js
- Afianzar los conocimientos obtenidos sobre el diseño de bases de datos relacionales y su implementación utilizando SQLite y Sequelize.

Descripción de la práctica

Esta práctica consiste en la ampliación del proyecto de gestión de usuarios y quizzes desarrollado en clase. Se añadirá la posibilidad de jugar a responder las preguntas almacenadas en el servicio de manera aleatoria y consecutiva hasta fallar alguna de ellas. Se implementará también una nueva funcionalidad para almacenar las puntuaciones conseguidas por cada usuario (número de aciertos) y para poder consultarlas en cualquier momento.

Descargar el código del proyecto

Es necesario utilizar la **versión 12 de Node.js** para el desarrollo de esta práctica. El proyecto debe clonarse en el ordenador desde el que se está trabajando:

```
$ git clone https://github.com/CORE-2020/Entrega5_BBDD
```

A continuación se debe acceder al directorio de trabajo, instalar las dependencias y configurar la base de datos (migraciones y seeders). Entonces puede arrancarse el programa.

```
$ cd Entrega5_BBDD
$
$ npm install
$
$ npm run migrate   ## En Windows: npm run migrate_win
$
$ npm run seed      ## En Windows: npm run seed_win
$
$ npm start     ##  or 'node main'
....
> h
  Commands (params are requested after):
```

```
> h              ## show help
>
> lu | ul | u    ## users: list all
> cu | uc        ## user: create
> ru | ur | r    ## user: read (show age)
> uu             ## user: update
> du | ud        ## user: delete
>
> lq | ql | q    ## quizzes: list all
> cq | qc        ## quiz: create
> tq | qt | t    ## quiz: test (play)
> uq | qu        ## quiz: update
> dq | qd        ## quiz: delete
>
> lf | fl | f    ## favourites: list all
> cf | fc        ## favourite: create
> df | fd        ## favourite: delete
>
> e              ## exit & return to shell
>
```

Tareas

El alumno debe implementar las siguientes nuevas funcionalidades sobre el proyecto proporcionado:

Funcionalidad play

Debe incluirse el nuevo comando `p` (play) que comienza una nueva ronda de preguntas. Al ejecutar este comando los quizzes almacenados en el sistema (en concreto el campo `question` de cada quiz) van mostrándose (usando la función `rl.questionP`) de manera aleatoria y consecutiva para tratar de contestarlos.

- Si se contesta correctamente a un quiz y hay más quizzes disponibles se muestra el mensaje `The answer "xxxxxxx" is right!` usando la función `rl.log` y después la pregunta siguiente (usando la función `rl.questionP`) siguiendo un orden aleatorio.
- Si se contesta correctamente a un quiz y no hay más quizzes disponibles se muestra el mensaje `The answer "xxxxxxx" is right!` usando la función `rl.log` y después la puntuación obtenida (número de aciertos) con el formato `Score: X` y usando la función `rl.log` .
- Si se contesta incorrectamente a un quiz se muestra el mensaje `The answer "xxxxxxx" is wrong!` usando la función `rl.log` y después se muestra por pantalla la puntuación obtenida (número de aciertos) con el formato `Score: X` y usando la función `rl.log` .

Funcionalidad scores

Las puntuaciones obtenidas por un usuario registrado deben almacenarse en la base de datos al terminar una ronda de preguntas. Además las puntuaciones de los usuarios deben poder consultarse en cualquier momento usando el nuevo comando `ls` (list score). Para ello deben realizarse los siguientes pasos:

1. Incluir un nuevo modelo `Score` así como su relación con la tabla de usuarios. Hay que tener en cuenta que para un mismo usuario se almacenarán varias puntuaciones, una por cada vez que juegue (debe utilizarse el alias `scores` para referenciar las puntuaciones asociadas a un usuario). La tabla `Scores` almacenará los siguientes atributos:
    - Atributo `wins` de tipo entero que indica el número de quizzes contestados correctamente. No puede ser nulo y debe ser mayor o igual que 0.
    - Referencia `userId` al identificador de usuario de la tabla User.
2. Implementar una migración de la base de datos para crear la nueva tabla `Scores` .
3. Ampliar la funcionalidad play para solicitar el nombre de usuario (usando la función `rl.questionP`) al finalizar el juego y para almacenar la puntuación asociada a dicho usuario en la tabla `Scores` de la base de datos. En caso de que el usuario introducido no exista se creará un nuevo usuario con el nombre introducido y edad 0.
4. Implementar la funcionalidad del nuevo comando `ls` (list score) que pinta una lista (cada línea debe pintarse con la función `rl.log`) de las puntaciones almacenadas en la base de datos ordenadas de mayor a menor con el siguiente formato (para dar formato a la fecha se debe utilizar el método `toUTCString()` del objeto `Date`):

```
Peter|3|Tue, 18 Feb 2020 14:20:27 GMT
Susan|2|Tue, 18 Feb 2020 14:20:27 GMT
Kike|2|Tue, 18 Feb 2020 14:20:27 GMT
Patri|1|Tue, 18 Feb 2020 14:20:27 GMT
```

¡¡Nota importante!!: Si durante el desarrollo de la práctica crees que has podido "romper" la base de datos o crear alguna inconsistencia siempre puedes reiniciar su estado incial eliminando el fichero `db.sqlite` y ejecutando de nuevo los comandos `npm run migrate` y `npm run seed`

Prueba de la práctica

Para ayudar al desarrollo, se provee una herramienta de autocorrección que prueba las distintas funcionalidades que se piden en el enunciado. Para utilizar esta herramienta debes tener node.js (y npm) (https://nodejs.org/es/) y Git instalados.

Para instalar y hacer uso de la herramienta de autocorrección en el ordenador local, ejecuta los siguientes comandos en el directorio del proyecto:

```
$ npm install -g autocorector     ## Instala el programa de test
$ autocorector                    ## Pasa los tests al fichero a entregar
.............................     ## en el directorio de trabajo
... (resultado de los tests)
```

También se puede instalar como paquete local, en el caso de que no se dispongas de permisos en el ordenador desde el que estás trabajando:

```
$ npm install autocorector     ## Instala el programa de test
$ npx autocorector             ## Pasa los tests al fichero a entregar
.............................  ## en el directorio de trabajo
... (resultado de los tests)
```

Se puede pasar la herramienta de autocorrección tantas veces como se desee sin ninguna repercusión en la calificación.

¡¡Nota importante!!: Tenga en cuenta que en esta práctica el autocorrector comprueba, entre otras cosas, la entrada y salida de la consola en su programa para validar los resultados. Tenga cuidado por lo tanto con los logs que utilice a modo de depuración durante el desarrollo ya que podrían interferir en dicha salida y entrada alterando el resultado de las pruebas. Ante la duda comente los logs a la hora de pasar los tests.

Instrucciones para la Entrega y Evaluación.

Una vez satisfecho con su calificación, el alumno puede subir su entrega a Moodle con el siguiente comando:

```
$ autocorector --upload
```

o, si se ha instalado como paquete local:

```
$ npx autocorector --upload
```

La herramienta de autocorrección preguntará por el correo del alumno y el token de Moodle. En el enlace https://www.npmjs.com/package/autocorector se proveen instrucciones para encontrar dicho token.

RÚBRICA: Se puntuará el ejercicio a corregir sumando el % indicado a la nota total si la parte indicada es correcta:

- **40%:** Funcionalidad play
- **60%:** Funcionalidad scores

Si pasa todos los tests se dará la máxima puntuación.