Lunar Lander

Información técnica de la aplicación

Definición de tablas de base de datos

La base de datos “lunarlander” se compone de 3 tablas.

- **user**: Tabla en la que se almacenarán los datos de los usuarios registrados en la aplicación.
- **configuration**: Tabla en la que se almacenarán las configuraciones de los usuarios registrados en la aplicación.
- **score**: Tabla en la que se almacenarán las puntuaciones de los usuarios y con la cual se determinará la lista de usuarios activos jugando una partida en función del valor del campo “endTime”.

Tabla: user

Nombre del campo	Tipo de variable	Otra información
id	SERIAL	Primary key, not null
name	VARCHAR(50)	Not null, length >= 3
username	VARCHAR(50)	Unique, not null, length >= 4
password	VARCHAR(256)	Not null, SHA256
email	VARCHAR(100)	Not null, email format

Tabla: configuration

Nombre del campo	Tipo de variable	Otra información
id	SERIAL	Primary key, not null
user_id	INTEGER	Not null, reference user (id)
configName	VARCHAR(20)	Not null, length >= 4
diff_id	INTEGER	Not null
nave_id	INTEGER	Not null
planet_id	INTEGER	Not null

Table: score

Nombre del campo	Tipo de variable	Otra información
id	SERIAL	Primary key, not null
conf_id	INTEGER	Not null, reference configuration (id)
speed	FLOAT	Not null
fuel	FLOAT	Not null
trys	INTEGER	Not null
initTime	TIMESTAMP	Not null, default CURRENT TIMESTAMP
endTime	TIMESTAMP	Not null, default NULL

Información adicional:

El campo endTime se utilizará para saber que usuarios están jugando una partida. Cuando se inicie la partida se hará un insert y cuando acabe la partida se modificará el registro agregando en endTime correspondiente.

Backend

Servlets

CreateUserServlet.doPost

Añade un nuevo usuario en caso de que no exista en base de datos.

Parámetros recibidos:

- name: nombre real del usuario que se quiere registrar en la aplicación.
- userName: usuario que se quiere crear para iniciar sesión en la aplicación
- password: contraseña para el usuario que se quiere crear. Encriptada en SHA256.
- email: correo electrónico del usuario que se quiere registrar en la aplicación.

Definición del proceso:

Si existe el “username”:
    Devolver mensaje “Usuario ya existe en base de datos”
    Salir del servlet
Si existe el “email”:
    Devolver mensaje “Correo electrónico ya en uso para una cuenta”
    Salir del servlet
Crear el usuario en base de datos
Devolver mensaje “Usuario creado correctamente”

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error.

LoginServlet.doPost()

Comprueba las credenciales de usuario y devuelve un mensaje a corde con la validación.

Parámetros recibidos:

- userName: nombre del usuario el cual intenta hacer login.
- password: contraseña del usuario el cual intenta hacer login. Antes de comparar la contraseña, se encriptará en SHA256.

Definición del proceso:

Si el usuario no existe:
    Devolver mensaje de error “usuario no existe”
    Salir del servlet
Encriptar contraseña
Si la contraseña encriptada no coincide con la contraseña guardada en base de datos:
    Devolver mensaje de error “contraseña incorrecta”
    Salir del servlet
Devolver mensaje de confirmación “Validación ok”

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

CreateConfigurationUser.doPost()

Parámetros recibidos:

- userName: nombre de usuario al cual se le va a crear una nueva configuración asociada.
- configname: nombre de la configuración que se va a crear. No debe de existir para el usuario el cual la va a crear.
- diffId: valor que identifica la dificultad de la configuración.
- rocketId: valor que identifica el modelo de nave de la configuración.
- planetId: valor que identifica el modelo de luna de la configuración.

Definición del proceso:

Si el usuario no existe:
    Devolver mensaje de error “Usuario no existe”
    Salir del servlet
Si la configuración ya existe:
    Devolver mensaje de error “Configuración ya existe”
    Salir del servlet
Hacer un insert de la nueva configuración
Devolver mensaje de confirmación “Configuración creada correctamente”

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

GetConfigurationsUser.doPost()

Parámetros recibidos:

- userName: nombre del usuario del cual se desean obtener todas las configuraciones.

Definición del proceso:

Si el usuario no existe:
    Devolver mensaje de error “Usuario no existe”
Obtener todas las configuraciones de un usuario (List<Configuration>)
Crear un String con formato json mediante la librería gson y el objetos de la liste de configuraciones.
Devolver el string con formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

CreateScoreUser.doPost()

Parámetros recibidos:

- configurationId: id de la configuración con la cual el usuario va a jugar una partida.

Definición del proceso:

Obtenemos la configuración recibida por parámetros.
Si la configuración no existe:
    Devolver mensaje de error “Configuración no encontrada”
    Salir del servlet.
Crear objeto Score con la configuración obtenida.
Realizar insert de la Score en base de datos.
Devolver la scoreId generada con formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

DestroyConfigurationUser.doPost()

Parámetros recibidos:

- configurationId: id de la configuración la cual se quiere borrar.

Definición del proceso:

Obtenemos la configuración recibida por parámetros.
Si la configuración no existe:
    Devolver mensaje de error “Configuración no existe”
    Salir del servlet.
Borrar todas las Scores vinculadas a esa configuración.
Borrar la configuración.
Devolver mensaje de OK.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

GetConfigurationsUser.doPost()

Parámetros recibidos:

- userName: Usuario del cual se desea obtener sus configuraciones.

Definición del proceso:

Obtenemos el usuario recibido por parámetros.
Si el usuario no existe:
    Devolver mensaje de error “Usuario no existe”
    Salir del servlet.
Devolvemos la lista de configuraciones del usuario en formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

GetScoresUser.doPost()

Parámetros recibidos:

- userName: Usuario del cual se van a obtener sus Scores.

Definición del proceso:

Obtener el usuario recibido por parámetros.
Si el usuario no existe:
    Devolver mensaje de error “Usuario no existe”
    Salir del servlet.
Devolver las scores del usuario en formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

GetTopRanking.doGet()

Definición del proceso:

Obtenemos de base de datos una lista de los 10 usuarios con más partidas jugadas.
Devolvemos la lista de usuarios en formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

Nota: solo se ha de devolver el username y el count(*) de las scores.

SetConfigurationUser.doPost()

Parámetros recibidos:

- configurationId: Id de la configuración a la cual se va a actualizar los datos de la partida.
- diffId: Dificultad con la que se ha jugado la partida.
- rocketId: Nave con la cual se ha jugado la partida.
- planetId: Planeta con la cual se ha jugado la partida.

Definición del proceso:

Obtener la configuración recibida por parámetros.
Si no existe la configuración:
    Devolver mensaje de error “Configuración no existe”.
    Salir del servlet.
Cambiar la configuración de base de datos con los parámetros de la dificultad, modelo nave y modelo luna.
Devolver mensaje de OK.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

SetScoreUser.doPost()

Parámetros recibidos:

- scoreId: Identificador de la score la cual se va a actualizar.
- fuel: Fuel al acabar la partida.
- speed: Velocidad al acabar la partida.

Definición del proceso:

Obtenemos la score recibida por parámetros.
Si no existe la score:
    Devolver mensaje de error “Score no existe”
    Salir del servlet.
Actualizar en base de datos la score con los parámetros del fuel y la velocidad.
Devolver mensaje de OK.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

UsersOnline.doGet()

Definición del proceso:

Obtenemos de base de datos una lista de los usuarios que tengan una score con el endTime null y un startTime inferior a 5 minutos.
Devolvemos la lista de usuarios en formato json.

En cualquier momento, si ocurre una excepción:
    Devolver mensaje de error

Frontend

Login/SignUp

Validaciones

Todas las validaciones se realizan desde el formulario en el html mediante atributos de las etiquetas. La única validación de campos que se realiza en el javascript es la de comparar que el campo de la contraseña coincida en ambos sitios cuando el usuario se registra en la página.

LocalStorage/SessionStorage

Optamos por utilizar localstorage y sessionstorage para comprobar que sólo puedan jugar los usuarios registrados en la página, además de utilizarlo para el “remember me”.

Cuando un usuario realice el login con la casilla de “remember me” activada se guardará en localstorage los datos de “username” y “password” para autocompletar en el login cuando el usuario vuelva a conectarse y además para validar que el usuario está registrado al entrar en la página del juego.

Si el usuario realiza el login sin la casilla de “remember me” activada no se guardará localstorage, en su lugar se guardará sessionstorage que se borrará cuando se cierre la pestaña del navegador.

Registrar usuario

Se realiza una petición ajax POST al servlet “CreateUserServlet”.

Se envían los siguientes parámetros:

- userName: Usuario que se quiere registrar.
- name: Nombre real del usuario que se quiere registrar.
- password: Contraseña del usuario que se quiere registrar. (no encriptada)
- email: Correo electrónico de la persona que se quiere registrar.

En caso de que el servlet devuelve “Success”:

- Mostrar mensaje toast por pantalla devuelto de servidor.
- Limpiar los campos de registro.
- Ocultar el div de registro y mostrar el div de login.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Login usuario

Se realiza una petición ajax POST al servlet “LoginServlet”.

Se envían los siguientes parámetros:

- userName: Usuario el cual quiere entrar en la aplicación.
- password: Contraseña del usuario.

Adicionalmente, la petición ajax ha de ser síncrona. Esto se debe a que si la petición se realiza dentro de un formulario que devuelve true o false en función de lo que devuelva el servidor. Si la petición ajax es asíncrona siempre dará problemas a la hora de realizar el login.

En caso de que el servlet devuelve “Success”:

- Guardar localstorage/sessionstorage.
- Cambiar a la página del juego.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Game

Controles de juego

En la versión de ordenador los controles del juego son los siguientes:

- Con la tecla “espacio” se activan los motores de la nave, al soltar la tecla se desactivan los motores.

- Con la tecla “P” se pausa o se reanuda la partida, también es posible pausar la partida clickando sobre el botón de pausa.

- Con la tecla “R” se reinicia la partida, también es posible pausando la partida y clickando sobre el botón de reiniciar.

En la versión de móvil los controles del juego son los siguientes:

- Pulsando sobre la pantalla se activan los motores de la nave, al soltar la tecla se desactivan los motores.

- Para pausar el juego se ha de pulsar sobre el botón de pausa, para reanudar la partida pulsar sobre el botón de play.

- Para reiniciar la partida se ha de pausar la partida y después hacer click sobre el botón de reiniciar.

Menú: Configuration

Permite elegir la configuración que el usuario quiere para jugar una partida.

Puede cargar una configuración, crear una nueva configuración o borrar una configuración existente.

Las configuraciones se obtienen mediante una petición ajax POST al servlet “GetConfigurationsUser”.

Se envían los siguientes parámetros:

- userName: Usuario del cual se quieren obtener las configuraciones.

En caso de que el servlet devuelve “Success”:

- Se cargará cada configuración dentro de la lista de configuraciones.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Una nueva configuración se guarda mediante la petición ajax POST al servlet “CreateConfigurationUser”.

Se envían los siguientes parámetros:

- userName: Usuario del cual se quiere crear una nueva configuración.
- configname: Nombre de la configuración que se quiere crear.
- diffId: Dificultad de la configuración.
- rocketId: Modelo de nave de la configuración.
- planetId: Modelo de planeta de la configuración.

En caso de que el servlet devuelve “Success”:

- Mostrar mensaje toast por pantalla devuelto de servidor.
- Cargar la configuración enviada dentro de la lista de configuraciones ya existente.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Una configuración se elimina mediante un apetición ajax POST al servlet “DestroyConfigurationUser”.

Se envían los siguientes parámetros:

- configurationId: Configuración la cual se va ha eliminar.

En caso de que el servlet devuelve “Success”:

- Mostrar mensaje toast por pantalla devuelto de servidor.
- Eliminar la configuración enviada de la lista de configuraciones.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Nota: Cuando se vaya a eliminar una configuración se pedirá una confirmación al usuario, ya que al eliminar las configuraciones también se van a eliminar las Scores vinculadas a dicha configuración.

Menú: Scores

Permite ver las puntuaciones de un jugador.

Las Scores se obtienen mediante una petición ajax POST al servlet “GetScoresUser”.

Se envían los siguientes parámetros:

- userName: Usuario del cual se desean obtener las puntuaciones.

En caso de que el servlet devuelve “Success”:

- Cargar las puntuaciones obtenidas en una tabla.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Menú: Players Online

Permite ver una lista con los jugadores conectados que están jugando actualmente una partida.

La lista de jugadores online se consigue mediante una petición ajax GET al servlet “UsersOnline”.

En caso de que el servlet devuelve “Success”:

- Se carga una lista con los “username” de los usuarios online dentro de un lapso de 2 minutos jugando.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.

Menú: Ranking

Permite ver una lista con los usuarios más viciados del juego (top 10).

La lista de usuarios más viciados se obtiene mediante una petición ajax GET al servlet “GetTopRanking”.

En caso de que el servlet devuelve “Success”:

- Se carga una lista de usuarios con más partidas en orden descendiente con la cantidad de partidas que ha jugado cada jugador.

En caso de que el servlet devuelve “Error”:

- Mostrar mensaje toast por pantalla devuelto de servidor o bien “Error desconocido” en caso de que el error no esté definido.