Lunar Lander

Changelog

Changelog v0 - Proyecto base

Versión 0 del proyecto “Lunar Lander Teamwork”.

En esta versión inicial del proyecto se ha hecho un fork al proyecto “lunar-lander-javascript” de Urbinapro.

A partir de aquí los miembros del equipo se han asignado las áreas en las que trabajará cada uno:

Backend → arsg93
Frontend → SOSandreu1095
Control de versiones → JDTheRipperPC

Una vez asignadas las áreas de cada miembro, se han definido los diferentes objetivos a ir logrando progresivamente:

- Creación de un proceso para dar de alta a un usuario.
- Creación de un proceso para entrar en la aplicación con un usuario ya existente.
- Mejora gráfica y del código del juego.
- Menú de configuraciones de usuario.
- Listado de usuarios en línea.
- Listado de puntuaciones e información de usuario.
- PAAS para las pruebas del proyecto en la nube.

Repositorio en GITHUB del proyecto Lunar Lander Teamwork.

Adicionalmente se han pensado en ideas opcionales a llevar a cabo en caso de disponer tiempo suficiente para llevarlas a cabo tras pulir los objetivos fijos:

- Login con Facebook.
- Compartir puntuaciones al acabar una partida en las redes sociales.
- Multijugador el local.
- Enviar email tras el registro de usuario para notificar la nueva cuenta creada.

Changelog v1 - Definición de tablas

Versión 1 definición de tablas de base de datos “lunarlander”.

Para esta primera versión se ha definido un modelo entidad relación con el cual se ha escrito un script sql para la generación de las tablas sobre una base de datos PostgreSQL.

En el script sql se definen 3 tablas: user, configuration y score.

Una vez creadas las tablas se generan las clases que hacen referencia a las entidades de base de datos utilizando las herramientas JPA. También se generan los controladores para cada entidad. Al generar las entidades se ha seleccionado como relación LAZY.

Lista de entidades:

- User
- Configuration
- Score

Lista de controladores:

- UserJpaController
- ConfiguracionJpaController
- ScoreJpaController

Por el momento se codifica a mano en cada una de las 3 entidades el schema “public”. Posteriormente esto se cambiará modificando el fichero persistence.xml agregando un fichero orm.xml que disponga de la configuración del schema sin necesidad de tener que modificar el código de las entidades.

Listener del servidor para utilizar en EntityManagerFactory:

- PostgresqlListener

Changelog v2 - Sign up

Versión 2 añadiendo el proceso de registro de un nuevo usuario a la aplicación.

Backend

Se han creado los siguientes servlets para controlar el proceso de registro de usuario en el lado servidor:

- CreateUserServlet: Añade un usuario a la tabla “user” si no existe.

CreateUserServlet.doPost()

El servlet recibe los siguientes parámetros:

- name: nombre real de la persona que se registra en la aplicación.
- userName: usuario de la persona con el cual se podrá iniciar sesión en la aplicación.
- password: contraseña del usuario con el cual se podrá iniciar sesión en la aplicación.
- email: correo electrónico del usuario que se registra en la aplicación.

Se ha definido un nuevo método en el controlador UserJpaController “existByUsername” para comprobar si existe el nombre de usuario “userName” recibido por parámetros o no.

Frontend

Se han creado los siguientes ficheros html/css/js para controlar el proceso de registro de usuario en el lado cliente:

- login.html: Página html que controla el registro de usuario, posteriormente se modificará para añadir el login de usuario.
- login.js: Código javascript que controla la petición ajax post al servlet CreateUserServlet, posteriormente se modificará para añadir el login de usuario.
- login.css: Página de estilos vinculada a login.html.
- jquery.toast.min.js: Plugin de jquery que permite utilizar mensajes toast para notificar al usuario.
- jquery.toast.min.css: Página de estilos del plugin jquery.toast.min.js.

Repositorio del plugin toast en GITHUB.

Changelog v3 - Login

Versión 3 añadiendo el proceso de inicio de sesión en la aplicación.

Backend

Se han creado los siguientes servlets para controlar el proceso de inicio de sesión en el lado servidor:

- LoginServlet: Comprueba las credenciales de usuario.

LoginServlet.doPost()

El servlet recibe los siguientes parámetros:

- userName: Usuario con el cual se desea iniciar sesión en la aplicación.
- password: Contraseña del usuario a validar en base de datos.

Se ha definido un nuevo método en el controlador UserJpaController “findUserByUsername” para obtener el usuario si existe el nombre de usuario “userName” recibido por parámetros en base de datos o null si no existe.

Frontend

Se han modificado los siguientes ficheros html/css/js para controlar el proceso de registro de usuario en el lado cliente:

- login.html: Página html donde se ha añadido el formulario de inicio de sesión.
- login.js: Código javascript donde se ha añadido el proceso de inicio de sesión.
- login.css: Hoja de estilos modificada para mejorar la apariencia de la página.

En la petición ajax del login se ha cambiado para que se realice de manera síncrona, ya que si no daba problemas a la hora de cambiar del login.html al game.html.

Changelog v4 - Juego html/css/javascript

Versión 4 añadiendo el juego funcional.

Frontend

Se ha modificado el game.html del proyecto base y se han añadido imágenes, cambiado la distribución de los elementos del html, añadido css y cambiado el javascript (orientado a objetos + jquery)

Los objetos de javascript creados son los siguientes:

- rocket: Objeto que permite manipular los atributos de la nave.
- ConfigurationClass: Objeto que permite crear nuevos objetos de tipo configuración, ya sea cuando se cargue una lista de configuraciones o bien se cree una nueva configuración.

Changelog v5 - Configuraciones

Versión 5 añadiendo todo el tratamiento referente a las configuraciones.

Backend

Se han creado los siguientes servlets para controlar el proceso de registro de usuario en el lado servidor:

- GetConfigurationsUser: Obtiene las configuraciones de un usuario.
- CreateConfigurationUser: Crea una nueva configuración.
- DestroyConfigurationUser: Elimina una configuración y todas sus scores asociadas.
- SetConfigurationUser: Modifica una configuración. **(deprecated)**

GetConfigutationsUser.doPost()

El servlet recibe los siguientes parámetros:

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

CreateConfigurationUser.doPost()

El servlet recibe los siguientes parámetros:

- configname: Nombre de la configuración a crear.
- diffId: Dificultad de la configuración.
- rocketId: Modelo de nave.
- planetId: Modelo de luna.

Se ha definido un nuevo método en el controlador ConfigurationJpaController “existByConfigName” para comprobar si existe una configuración con el mismo nombre que la configuración que se quiere crear

DestroyConfigurationUser.doPost()

El servlet recibe los siguientes parámetros:

- configurationId: Identificador de la configuración que se quiere eliminar.

SetConfigurationUser.doPost() (deprecated)

El servlet recibe los siguientes parámetros:

- configurationId: Identificador de la configuración que se quiere modificar.
- diffId: Nuevo valor de dificultad.
- rocketId: Nuevo valor de modelo nave.
- planetId: Nuevo valor de modelo luna.

NOTA: Esta servlet está en desuso ya que no está la opción a modificar una configuración porque al modificarla también afecta al cálculo de las puntuaciones y se haría trampa.

Frontend

Se ha añadido un menú para manejar las configuraciones dentro del game.html.

Ventana modal donde un tab “configuration” nos permite seleccionar con que configuración jugar, crear una nueva configuración o bien eliminar una configuración junto con las scores vinculadas.

Un usuario no puede jugar si no tiene una configuración creada.

Changelog v6 - Scores

Versión 6 añadiendo el tratamiento de las scores de inicio y fin de partida.

Backend

Se han creado los siguientes servlets para controlar el proceso de registro de usuario en el lado servidor:

- GetScoresUser: Obtiene las puntuaciones de un usuario.
- CreateScoreUser: Crea una nueva puntuación al inicio de partida.
- SetScoreUser: Actualiza la puntuación al acabar la partida.

GetScoresUser.doPost()

El servlet recibe los siguientes parámetros:

- userName: Usuario el cual se quieren obtener todas sus scores asociadas.

CreateScoreUser.doPost()

El servlet recibe los siguientes parámetros:

- configurationId: Configuración con la cual se va a jugar la partida.

SetScoreUser.doPost()

El servlet recibe los siguientes parámetros:

- scoreId: Identificador de la score que se va a actualizar.
- fuel: cantidad de fuel al acabar una partida.
- speed: velocidad de aterrizaje.

Frontend

Se ha añadido un nuevo tab al menú “Scores” donde se cargará la lista de puntuaciones.

Al iniciar la partida se llama al servlet CreateScoreUser y al acabar al SetScoreUser.

Changelog v7 - Listados

Versión 7 añadiendo los listados de los jugadores online y el top 10 jugadores más viciados.

Backend

Se han creado los siguientes servlets para controlar el proceso de registro de usuario en el lado servidor:

- UsersOnline: Obtiene la lista de los usuarios conectados jugando una partida.
- GetTopRanking: Obtiene la lista de los 10 jugadores más viciados.

UsersOnline.doGet()

Se define un nuevo método en el controlador UserJpaController “getUsersEndTime” para obtener los jugadores online jugando una partida.

GetTopRanking.doGet()

Obtiene únicamente los 10 jugadores con mayor número de partidas jugadas.

Frontend

Se han añadido 2 tabs “Players Online” y “Ranking”.

En “Players Online” se encuentran los jugadores conectados jugando una partida.

En “Ranking” se encuentran los 10 jugadores con mayor número de partidas jugadas.

Changelog v8 - Últimas mejoras

Versión 8 añadiendo las últimas mejoras al proyecto.

Frontend

Se han añadido los tabs de “Instructions” y “About”.

En el tab about redirige a una nueva página “about.html”

Se han añadido las teclas “P” y “R” como atajos rápìdos para pausar el juego y reiniciar la partida respectivamente.

Se han corregido errores en el juego y se ha pulido la interfaz de usuario.

Se han añadido nuevas imágenes al proyecto.