

INFORME

TRABAJO PRÁCTICO - PROGRAMACIÓN III

LIGHTS OUT



Universidad
Nacional de
General
Sarmiento



Integrantes del grupo:

Abel Aquino
Lautaro Moreno
Karin Pellegrini

Comisión : 01

Profesores:

Patricia Bagnes
Ignacio Sotelo
Gabriel Carrillo

Introducción

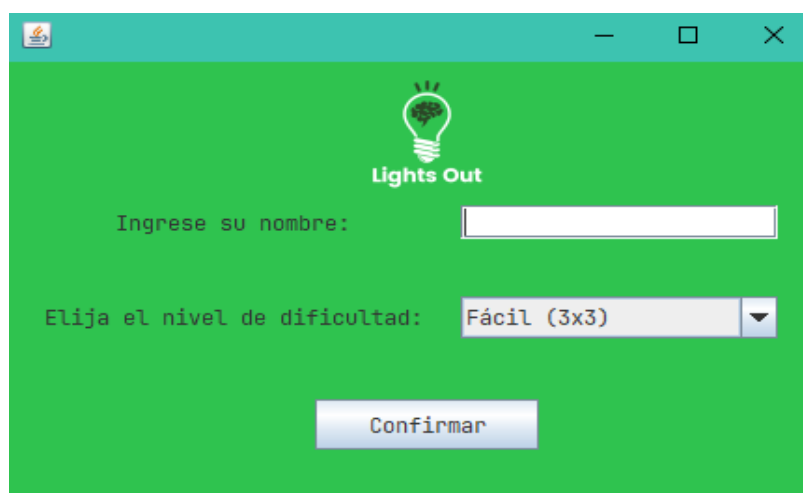
En este informe, se mostrará el programa realizado por los presentes alumnos para poder implementar una aplicación visual para el juego Lights Out. La presentación visual está dividida siguiendo la arquitectura MVP vista en clase. Usamos la librería de WindowBuilder para poder diseñar las interfaces llamadas "pantalla de inicio" y "pantalla de juego" en el plazo de 4 semanas desde que se anunció.

A continuación, vamos a mostrar las clases de las interfaces y sus casos de uso divididos en secciones.

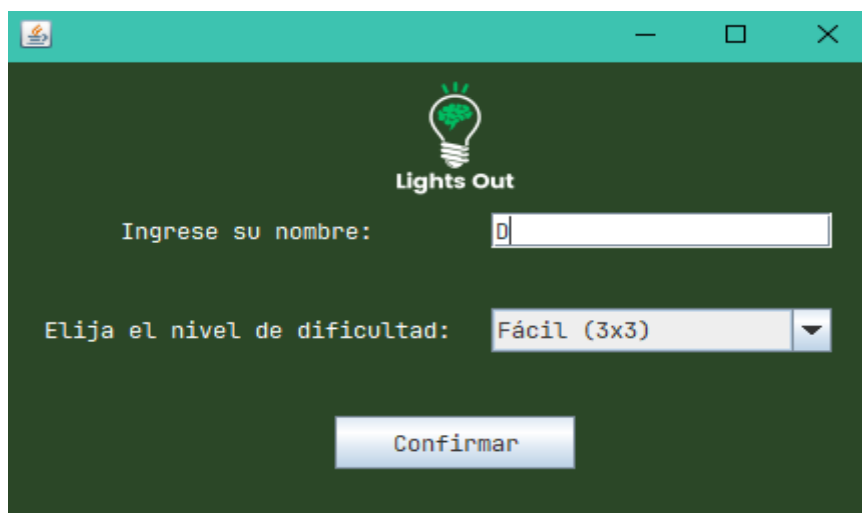
Pantalla de Inicio

En esta pantalla se pide el nombre del usuario y el nivel de dificultad que desea enfrentar mediante niveles donde se detallan el tamaño de las grillas, es decir, la cantidad de botones que tendrá cada fila y cada columna.

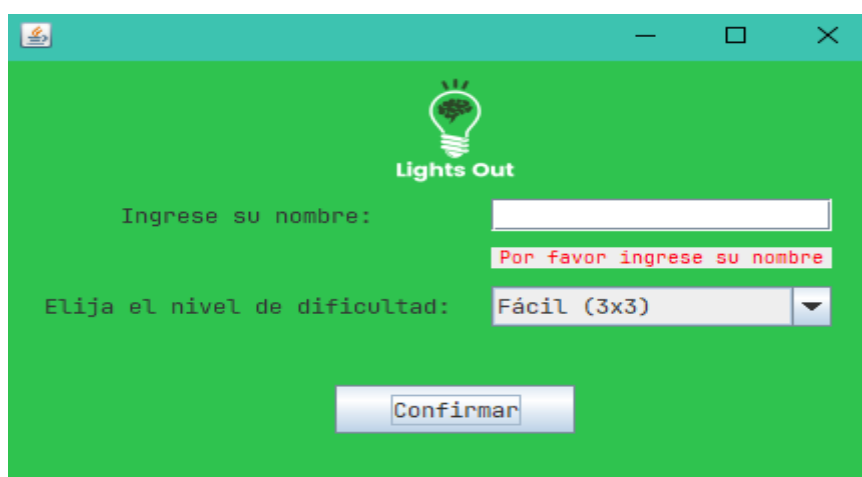
Al principio, la pantalla tendrá un color de fondo verde claro y un color de letra negro.



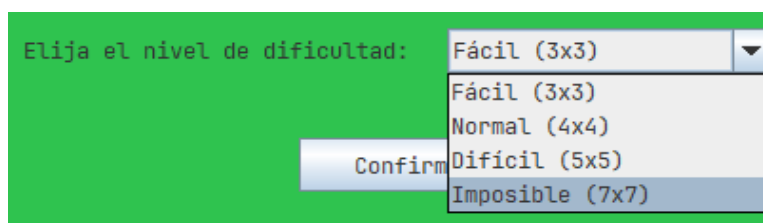
No obstante, el color de fondo y el de las letras puede cambiar si el usuario hace click en cualquier parte de la pantalla de inicio (excepto en los campos donde se pide que ingrese su nombre, el nivel de dificultad y el botón). Esto genera que el fondo de la pantalla sea un verde oscuro y el color de las letras sea blanco. A modo de representar la metodología del juego por cada botón.



Este registro cuenta con una validación que pide al usuario ingresar su nombre, caso contrario aparecerá un cartel de aviso con letras rojas y no nos dejará continuar con el juego si no se ingresa el nombre.

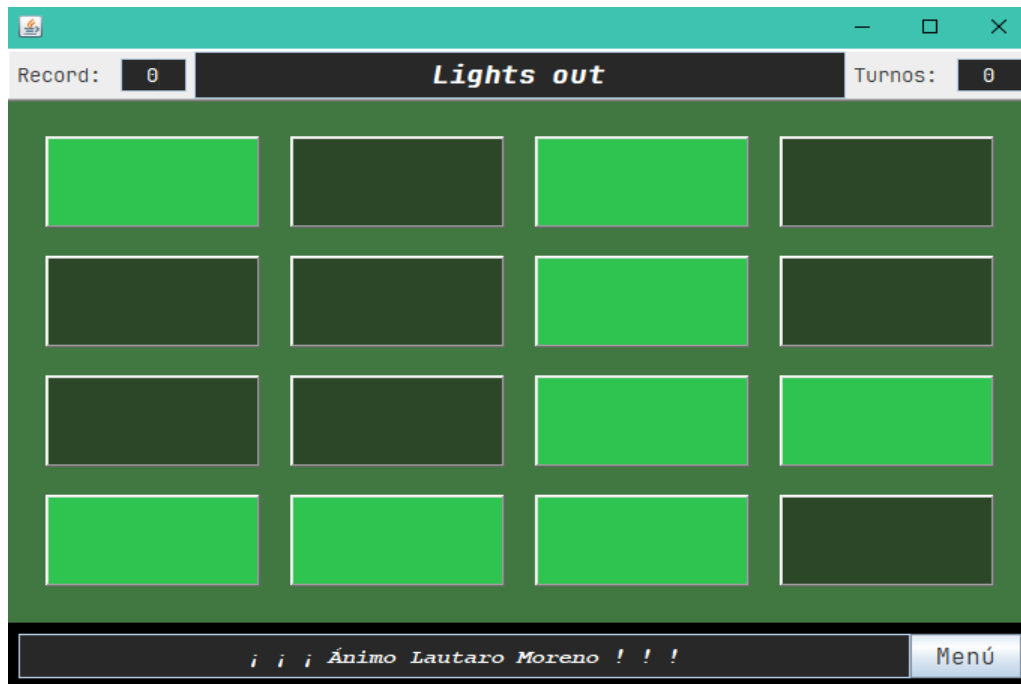


Para elegir el nivel de dificultad creamos una lista desplegable en donde tenemos 4 niveles para elegir, detallando la cantidad de botones que tendrá el juego por filas y columnas.



Luego, al apretar en el botón confirmar vamos a pasar a la ventana del juego.

Pantalla de Juego



Esta pantalla es donde el usuario podrá jugar a LightsOut, su objetivo será apagar todas las celdas de la pantalla (es decir, que aparezcan con el color verde oscuro).

Al inicio de cada partida, las celdas se crean con un estado que puede ser “encendida” o “apagada”, definido de forma aleatoria. Cuando el usuario toca una celda modifica el estado de todas las que se encuentran en la misma fila y columna, es decir que las que están prendidas se apagan y viceversa.

La pantalla se encuentra dividida en 6 secciones las cuales son:

- **Récord:** Muestra el récord actual, es decir la menor cantidad de turnos en la que el nivel fue completado en el turno anterior. Este récord varía según el nivel seleccionado.
- **Nombre del juego**
- **Turnos:** Cada vez que se hace click en una celda se incrementa el contador de turnos que le toma al usuario hasta completar el juego.
- **Cártel de ánimos**
- **Botón de menú:** Hace que el usuario vuelva a la pantalla de inicio en caso de que quiera cambiar el nombre o quiera seleccionar otro nivel. Se debe tener en cuenta que al hacer esto se reinicia la partida.

Niveles

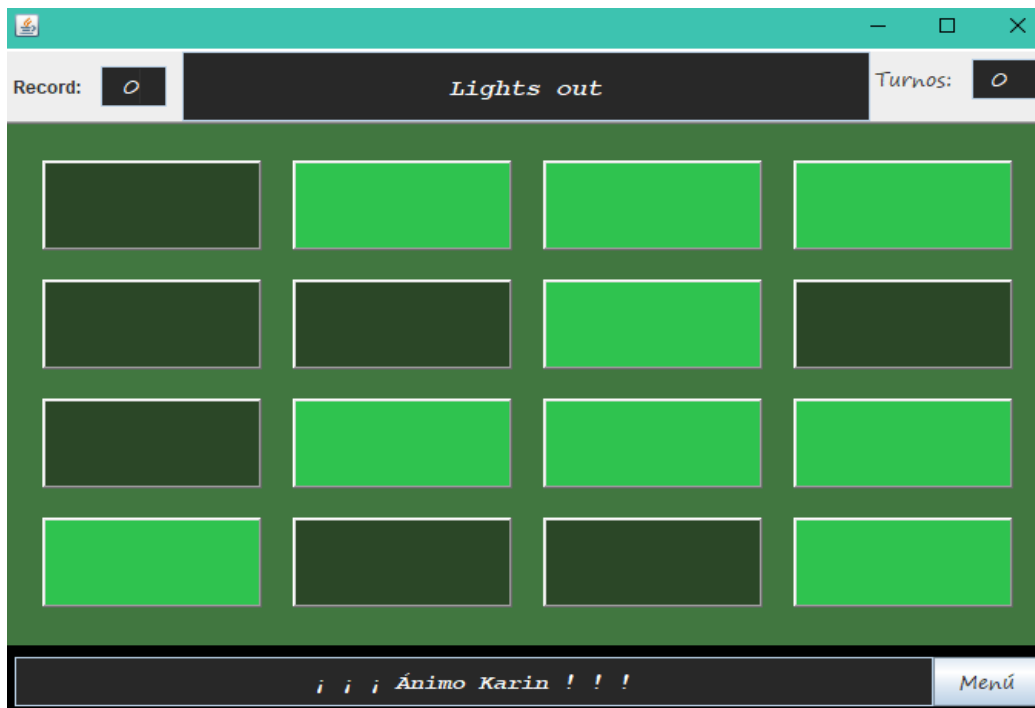
★ Caso de juego: partida ganada con nivel fácil



Dependiendo de la dificultad del nivel que el usuario haya seleccionado en la pantalla de inicio, el juego tendrá 4 posibles grillas con celdas, la primera (nivel fácil) es la anterior mostrada, con 3 filas por 3 columnas para apagar.

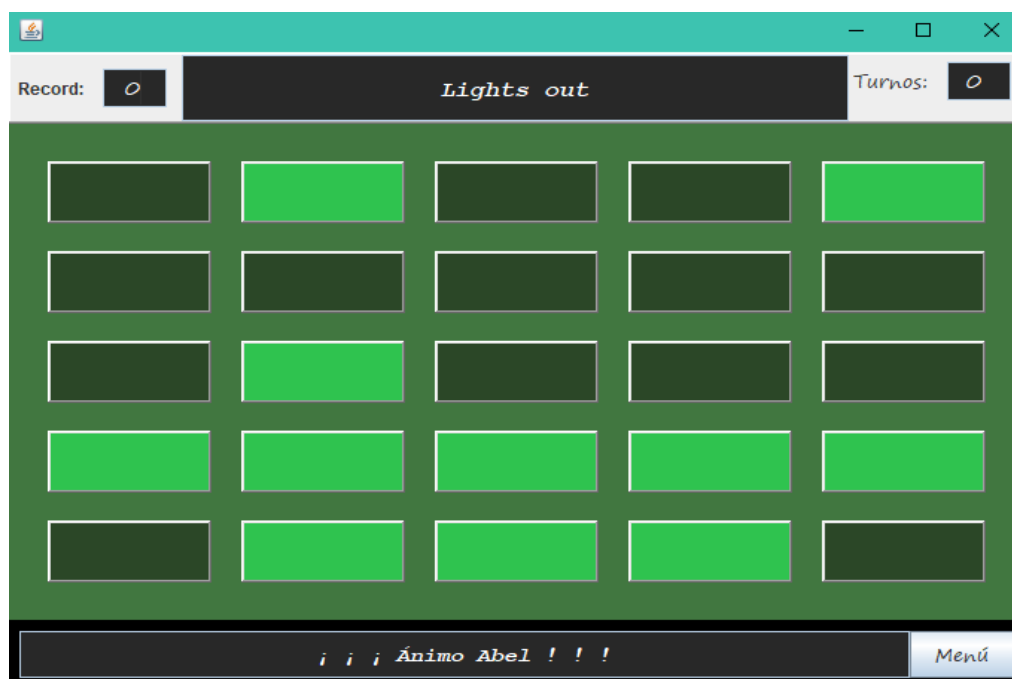
★ Caso de juego: nivel normal

Si el usuario quiere un equilibrio entre desafío y dificultad, sin dudas, el nivel medio es para él. En este nivel el tablero del juego goza de un tablero con 4 celdas y 4 filas para complejizar levemente la partida, tal y como se puede observar en las siguientes imágenes.



★ Caso de juego: nivel difícil

Para los jugadores que están familiarizados con el concepto del juego o simplemente les encantan los desafíos complejos, el nivel difícil, con un tablero aún más grande que el del anterior, es decir, 5 filas y 5 celdas es el indicado para marear incluso al más experimentado de los jugadores.



★ Caso de juego: nivel imposible

Por último, el juego cuenta con un nivel especialmente para el usuario más capaz o de plano al más temerario, con un tablero de 7x7, es capaz de lograr que ni sus propios desarrolladores puedan ganarlo.



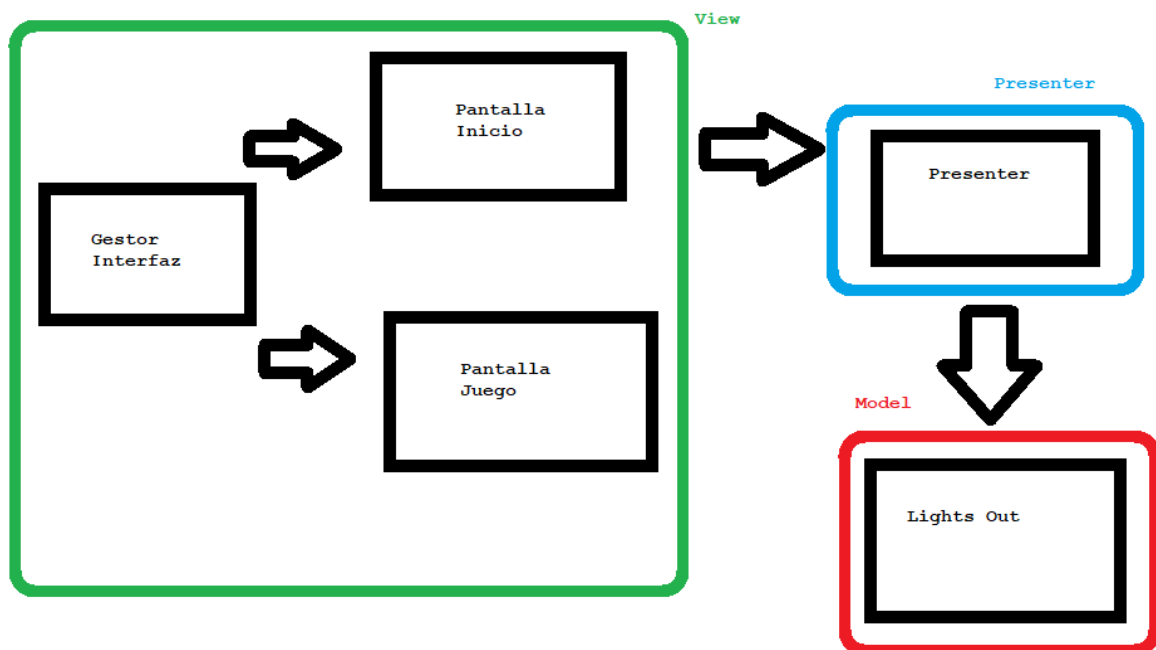
En cuanto al código para la implementación, el único método importante (fuera de estilizar las secciones del título y el mensaje de ánimo) se llama `crearTableroConCeldas()`. Este método es el responsable de que se creen las celdas en el tablero, que se defina la estructura, se enciendan o apaguen los botones de forma aleatoria, y se le asignen eventos para detectar las acciones del usuario a cada celda. Para lograr todas estas cosas, se dividió el método en varias funciones que hacen una acción en específico, las cuales son :

- `colorearBotonSegunElEstadoRandom(int fila,int columna); //`
Permite colorear los botones (ubicándolos por medio de la fila y la columna donde están) de forma aleatoria al inicio del juego.
- `asignarEventosDeCelda(int fila,int columna); //` Permite asignar lo que pasa cuando el usuario hace click sobre un botón (enciende o apaga las celdas de esa misma fila y

columna), además de cambiar levemente los colores de las celdas cuando el mouse pasa por arriba de ellas.

Arquitectura técnica

El proyecto está construido sobre la arquitectura MVP, en la que se encuentra dividido en 3 módulos, la “*vista*”, el “*controlador*”, y por último, el “*modelo*”. Cada módulo cuenta con sus clases y cada cual tiene su responsabilidad, logrando un código más amigable para futuras funcionalidades.



Módulo Model

Esta clase, viene a representar la parte lógica del juego, la encargada de cada dato que se registre en el juego, así como el estado de las celdas y si el usuario logró finalizar con el objetivo del juego. A continuación, damos una descripción mejor de los métodos que tiene esta clase y su función:

- `crearTablero(int filaColumna)` // Me permite crear el tablero de la partida con la cantidad de filas y columnas que reciba como parámetro.
- `validarTablero()` // Funciona como validador, en caso de que el tablero tenga todas las celdas apagadas, crea otro tablero. Utiliza el siguiente método para saber si están todas las celdas apagadas.

- `todoTableroEnFalse()` // Permite ver si el tablero se creó con todas las celdas apagadas, lo cual no debe pasar
- `inicializarJuego()` // Recorre cada elemento del tablero y le asigna un valor aleatorio para estar encendida o apagada.
- `estaEncendido(int fila, int columna)` // Se fija si la celda, que se ubica con la fila y la columna, pasados por parámetro, si está encendido o apagado. Retornando ese valor.
- `cambiarEstado(int fila, int columna)` // Cambia el estado de una celda dentro del tablero.
- `registrarRecord(int nuevoRecord)` // Recibe el record actual y lo registra según el largo del tablero (por ejemplo, si es un tablero de 3x3 entonces es el nivel fácil y lo registra como el record de ese nivel).
- `consultarRecordHistorico()` // Me devuelve el récord registrado según el nivel en el que estoy (igual que en el método pasado, si el tablero es de 3x3 entonces estoy jugando en el nivel fácil).

Módulo Presenter

Esta clase se enfoca en ser un intermediario entre las pantallas, vistas previamente, con la lógica del juego. Es decir, es el encargado de proveer los datos del juego, el estado de las celdas (si están encendidas o apagadas), a través del método `cambiarEstado(int fila, int col)` que recibe la fila y la columna donde está ubicada la celda puede hacer la orden para que se cambie su estado (en el caso de que en la interfaz se haya hecho un click sobre una celda), `iniciarJuego(int numeroNivel)` hace la orden para crear un tablero según el nivel de dificultad que el usuario haya elegido dentro de la pantalla de inicio, `registrarRecord(int nuevoRecord)` puede registrar un récord nuevo siempre que esté previamente haya superado al anterior (recordemos que cuando decimos “superar” nos referimos a si la cantidad de turnos es menor que el récord guardado) y sabe si el juego está acabado o no por medio del método `verificarEstadoDelTablero()`.

Cabe resaltar, que los métodos mencionados son los más usados e importantes dentro de esta clase.

Módulo View

Para implementar correctamente el mvp y que la conexión entre la vista y el presentador sea de lo más óptimo, se optó por crear tres clases para gestionar correctamente el momento de mostrar una u otra pantalla. Con esta decisión se logra mantener una única instancia de la clase “*Presenter*” (para más información consultar la sección “*Módulo Presenter*”).

Dentro de este módulo encontramos las clases “*GestorInterfaz*”, “*PantallaInicio*” y “*PantallaJuego*”.

Estos comportamientos los maneja el gestor mediante los métodos:

- `irAMenu()` // Cierra la pantalla donde el usuario ingresa sus preferencias e invoca a la pantalla donde se visualiza el juego.
- `abrirAJuego()` // Previo a usar este método, la pantalla de inicio se cierra. Se encarga de iniciar una pantalla de Juego con el nombre del usuario (ingresado previamente), el presenter (creado en el *GestorInterfaz*) y el mismo *GestorPantalla*, y la hace visible.

Los métodos, se envía el presentador por parámetro y así es como se logra que sea global.

Al ejecutar la segunda, se procede a realizar el camino de vuelta, se finaliza la pantalla del juego y se abre el menú.

Conclusión

Para concluir con este informe, podemos decir que el proyecto se pudo desarrollar cumpliendo con los objetivos pedidos por la consigna, a lo largo de las cuatro semanas el grupo fue dando ideas, algunas implementadas y otras descartadas.

Entendemos que se logró resolver los desafíos como:

- Implementar la arquitectura MVP.
- Reducir en la medida de lo posible los llamados entre los distintos módulos.
- Mantener una única instancia de la clase ***Presenter*** y ***LightsOut***.
- Añadir más niveles y su respectivo récord histórico.
- Mejorar la experiencia de usuario, como por ejemplo agregando sonidos para una mejor inmersión en la aplicación.

Así como tuvimos problemas que logramos solucionar, también surgieron contratiempos que nos impidieron implementar más funcionalidades. Dentro de estos pendientes encontramos, por mencionar algunos:

- Sugerencia de próximo movimiento.

- Música ambiental para el juego.
- Limitar la creación continua de pantallas de inicio y de juego (con esto, nos referimos a cuando se “abre” o se “cierra” una pantalla y llamamos a otra. Lo que pasa realmente es que dejan de ser visibles pero siguen estando ahí. Para cuestiones del juego, esto no tiene mucha importancia, pero en cuestiones de la memoria y los recursos que se usan es algo que requiere mejorar para usar sólo una pantalla de inicio y una de juego).

Como todo proyecto, aún puede ser mejorado, sin embargo, el equipo quedó muy conforme con el resultado de lo trabajado.