**OBLIGATORIO (PARTE 1)**

| | | | | | |
|---|---|---|---|---|---|
| **EVALUACION** | OBLIGATORIO 1 | **GRUPO** | TODOS | **FECHA** | Octubre 2017 |
| **MATERIA** | Diseño y Desarrollo de Aplicaciones | | | | |
| **CARRERA** | Analista Programador | | | | |
| **CONDICIÓN** | Entrega 13/11/2017 | | | | |

**El objetivo del presente trabajo obligatorio es modelar e implementar la primera versión prototipo de un juego de buscaminas en red.**

El proyecto de este prototipo está dividido en dos etapas.

En esta primera versión el desarrollo se focalizará en:

1. Definir Usuarios.
2. Ingreso al juego.
3. Prototipo del juego.
4. Monitoreo y repetición de partidas

En esta versión no se implementará la persistencia de la información.

## 1) Definir Usuarios.

Información básica que se desea manejar sobre los usuarios: nombre de usuario (que debe ser único), contraseña y nombre completo.

Se deberán definir usuarios Administradores y Jugadores.

Los jugadores poseen un saldo, que indica el dinero que tienen acreditado en el sistema.

**NO** es necesario implementar una interfaz de usuario para el mantenimiento de usuarios. El sistema deberá tener pre-cargada la información sobre los mismos, de modo que al iniciarse ya cuente con un conjunto de usuarios definido.

## 2) Ingreso al juego.

Para ingresar al juego los jugadores deberán identificarse con su nombre de usuario y contraseña.

Un mismo jugador no podrá ingresar dos veces al mismo tiempo y los usuarios administradores no podrán ingresar al juego.

Luego de ingresar, el jugador deberá indicar el tamaño del tablero del juego y luego el sistema le indicará al jugador que está esperando que se inicie una nueva partida, o, si ya hay otro jugador esperando el inicio de una partida, se iniciará automáticamente una partida para ambos jugadores.

En una partida de buscaminas participan dos jugadores y puede haber N partidas al mismo tiempo.

No podrán ingresar al juego jugadores cuyo saldo sea menor a la apuesta inicial.

En cada partida de buscaminas el primer jugador que ingreso juega con un color y el segundo con otro distinto. (Los mismos colores se utilizan para todas las partidas)

## 3) Prototipo del juego de buscaminas.

***}***

***Dinámica del juego.***

En un juego de buscaminas hay un tablero de forma cuadrada compuesto por N x N casilleros.

(N debe ser mayor o igual a 3 y menor o igual a 10)

Al iniciar el juego todos los casilleros están “tapados” (por ejemplo, se muestran en color gris)

El primer jugador “descubre” uno de los casilleros, si no hay una mina en ese casillero, el casillero queda marcado con el color del jugador y el turno pasa al otro jugador. Así sucesivamente hasta que un jugador descubre un casillero donde hay una mina y pierde el juego.

Ningún jugador puede seleccionar un casillero que haya sido descubierto” anteriormente por el mismo o por el otro jugador.

Al iniciar el juego hay una sola mina en el tablero, luego de cada turno (ambos jugadores han descubierto un casillero) se agrega una segunda mina (en un casillero no descubierto) y así sucesivamente hasta que el juego finalice.

Al finalizar el juego se deben mostrar todas las minas ocultas en el tablero.

***Apuestas***

Al comienzo de cada partida cada jugador apuesta automáticamente el valor correspondiente a la apuesta inicial. En cualquier momento de la partida un jugador puede aumentar la apuesta. El otro jugador puede o bien aceptar la apuesta o bien rechazarla (no destapar un casillero), si la rechaza pierde la partida. Un mismo jugador no puede aumentar la apuesta dos veces consecutivas. Ningún jugador puede apostar un valor mayor al saldo de su rival. Al finalizar la partida el jugador ganador obtendrá el dinero apostado por ambos jugadores.

## 4) Monitoreo y visualización de partidas

Este caso de uso estará disponible solo para los usuarios administradores, que deberán identificarse mediante su usuario y contraseña.

Se debe mostrar en una lista con todas las partidas con la siguiente información y en el siguiente formato:

[Nº Partida] [En juego o finalizada] [Nº turno ] jugador 1 ($ saldo) - jugador 2 ($ saldo) : $Total apostado

El administrador podrá seleccionar una partida de la lista y visualizar la partida en una ventana diferente. Esta ventana debe mostrar la misma información que se muestra en la lista de partidas y el tablero de la partida. Si la partida está finalizada (o si finaliza mientras la esta visualizando), el administrador podrá repetir paso a paso el desarrollo de la partida. En la repetición no es necesario mostrar la dinámica de las apuestas. Solo el total apostado al momento de descubrirse cada casillero.

## Interfaz gráfica

El objetivo de este prototipo es emular una situación en la cual cada jugador interactúa con el sistema desde su computadora.

Para emular esta situación, se deberá implementar una ventana general de la aplicación y una interfaz para cada uno de los usuarios, de forma que cada usuario ingresara al sistema desde su propia interfaz.

En la ventana general de la aplicación estarán disponibles los siguientes casos de uso:

*-Crear una interfaz para un jugador.*

*-Crear una interfaz para un administrador.*

*-Finalizar la aplicación. (Solo si no hay juegos activos)*

En la interfaz de cada jugador estarán disponibles los siguientes casos de uso:

*-Ingresar al juego.*

*-Jugar al buscaminas*

En la interfaz de cada administrador estarán disponibles los siguientes casos de uso:

*-Ingresar al sistema.*

*-Monitorear y visualizar partidas*

**Durante una partida se debe mostrar en las ventanas de los jugadores los siguientes datos:**

Nombre completo del jugador
Nombre completo del jugador rival.
Saldo actual del jugador
Monto total apostado (por ambos jugadores)
Numero de turno
Tablero con los casilleros tapados y descubiertos por ambos jugadores (con los colores correspondientes)

Los casilleros deberán mostrarse gráficamente, no como texto. El usuario deberá poder seleccionar el casillero a descubrir haciendo click sobre el mismo.

**Importante: La información de todas las ventanas debe obligatoriamente actualizarse de manera automática, sin necesidad de que el usuario indique que desea actualizar la información.**

**<u>Requerimientos de diseño para esta versión:</u>**

1) Maximizar la modularidad y claridad del código. Para esto utilice la métrica que dice que ningún método debería tener más código que el que se puede visualizar en una pantalla.
2) Minimizar la duplicación de código. Evitar métodos o porciones de código que realizan la misma tarea.
3) División física de las clases en al menos 2 paquetes: Clases para la interfaz de usuario y clases de la lógica del negocio.
4) División lógica:

a. Interfaz de usuario: Se encarga de capturar la información que el usuario debe proveer y presentar la información que “el sistema” provee. Evite codificar partes de la lógica del sistema (reglas de negocio) en las clases que se encargan de realizar la interfaz con el usuario.
b. Lógica de negocio o modelo: Se encarga de validar y almacenar la información provista por los usuarios y de implementar la lógica necesaria para proveer de información a los mismos. Evite codificar la presentación de la información en las clases que se encargan de la lógica del sistema.

5) La información entre la capas o subsistemas se comunicará utilizando objetos de las clases del dominio del problema.
6) Uso del patrón de diseño “Fachada”. La política propuesta para este patrón es ocultar a las clases de la interfaz de usuario, todas las clases de la lógica del negocio o modelo que no pertenezcan al dominio del problema.
7) Maximizar la aplicación del GRASP Experto para asignar las responsabilidades.
8) Utilizar una arquitectura M.V.C.

---

**Notas**

- Las posibles omisiones, ambigüedades o contradicciones que surjan del estudio de los requerimientos detallados en este documento serán analizadas y corregidas en clase durante el curso.

**Se pide entregar**

***Implementación del sistema en Java con interfaz de usuario grafica cumpliendo con todos los requerimientos funcionales y de diseño solicitados.**

***2 Diagramas de Clases:**

-Un diagrama de clases conceptual modelando el dominio del problema.

-Un diagrama de clases de diseño (o mas de uno si lo considera apropiado) que incluya a todas las entidades que participan en la solución.

***Auto Evaluación:**

Descripción breve de aquellos requerimientos funcionales o de diseño que faltan o no funcionan correctamente y/o pueden ser mejorados y el motivo.

Datos de prueba: Listado impreso con el nombre de usuario, contraseña y saldo de los jugadores y administradores pre-cargados en el sistema.

**Entrega**

Se deberán entregar dos copias del sistema (incluyendo el código fuente) y la documentación en formato impreso.
Los diagramas UML solo se corregirán siempre y cuando hayan sido entregados en papel.
Es obligatorio utilizar estricta notación UML en todos los modelos.