Documentación Bingo

Índice

Planteamiento del Proyecto

Cuando recibimos la tarea del proyecto comenzamos a plantear ideas, salió la posibilidad de hacer el proyecto con txt como base de datos, también apareció la opción de utilizar MYSQL, al final después de mucho discutir vimos viable una opción híbrida ya que había datos que necesitábamos guardar pero no

tenían la suficiente relevancia como para crear una tabla en MYSQL para ellos como por ejemplo el número de bola que saca el bombo para que no se volviera a repetir.

También decidimos utilizar POO y TDD, los cuales vamos a explicar mas adelante.

POO

Para crear el juego del Bingo hemos pensado en utilizar Programación Orientada a Objetos, las clases que hemos utilizado son:

- Bingo

  La clase Bingo tiene 4 atributos para determinar quien es el jugador 1, el jugador 2, jugador 3 y jugador 4.

  Esta clase va a recibir los nombres, el número de cartones que va a tener el jugador y la url de la imagen de cada jugador. A su vez el atributo que determina a cada jugador va a tener el valor de un nuevo objeto que se crea en la clase jugador.

  Al poner que los atributos de Bingo son objetos de otra clase para acceder a ellos tenemos que hacer 2 funciones una para obtener a 1 jugador o a todos los jugadores.

- Jugador

  La clase Jugador tiene 5 atributos, en el primero se determina el nombre del jugador, en el segundo como en el caso de bingo hay un atributo carton1 que es una objeto de la clase cartón, en tercer lugar tenemos un atributo para la imagen, y posteriormente hay dos atributos para el 2 cartón y el tercero, estos atributos no se indican que tienen que ser de tipo clase cartón ya que pueden existir o no y si se indica nos obliga a que exista siempre y no funciona correctamente.

El construct de esta clase crea a cada jugador asignándoles los cartones que se asignan desde la clase cartón a cada cartón del jugador (no puede dar cartones repetidos) e introduce dentro de la BD el nombre del jugador y su imagen en la tabla jugadores y una vez ese jugador tiene unos cartones asignados añade el id del jugador el id único del cartón y la secuencia en la que se encuentra el cartón por defecto, esta secuencia se utiliza para saber si una casilla es entera de un color, si no esta tachado en número o si esta tachado.

Además tiene una función para obtener el nombre del jugador y otra función para obtener todos los cartones de un jugador

▼ Cartón

Esta clase tiene 2 atributos, el primero un array de cartón y otro atributo que va a ser el id del cartón. Además hay una constante ce CARTONES = 60 ya que nuestro máximo van a ser 60 cartones ni más ni menos.

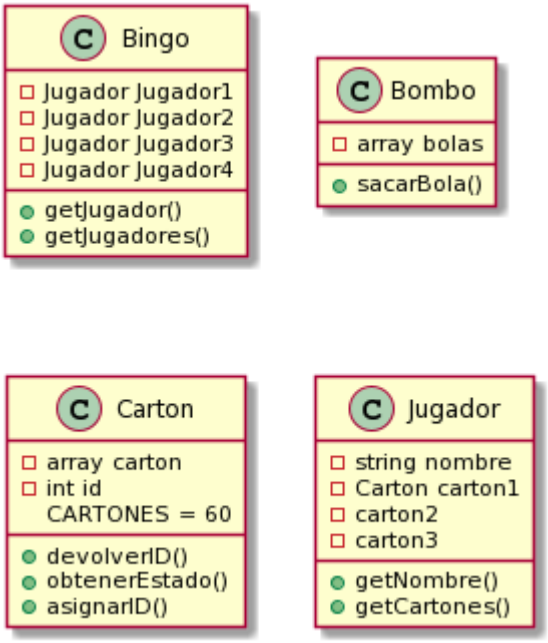
Esta clase lo que hace es comparar todos los números del 1 al 60 que hay en un array con un fichero txt donde se guardan los cartones que ya se han repartido. Si no se ha repartido le asigna el cartón al jugador y escribe el id del cartón en el fichero txt, si el numero se encuentra en el txt se quita del array todos los números que ya han salido para que no vuelvan a salir y se reordenan los indices para poder dar un cartón aleatoriamente entre los cartones disponibles.

También dispone de 2 funciones en las cuales en una devuelve el estado del cartón y en la otra se asigna el id.

▼ Bombo

La clase Bombo como su propio nombre indica hace de bombo que contiene los números del bingo, solo tiene un atributo que es un array llamado bolas. Al crearse el bombo la primera vez se va a llenar con las 90 bolas posibles del bingo, pero a medida que vaya soltando bolas va a ir leyendo las bolas que ha sacado de un txt y se las va a quitar del bombo para que no puedan volver a salir.

Diagrama de clases



Página de entrada

Esta página es la entrada a nuestra aplicación web de bingo por ello como ya sabemos solamente pueden jugar, las personas que son mayores de edad, estuvimos pensando en hacer un aviso para saber si eres mayor de edad o si eres menor de edad te redirigirá a otro sitio web

Si eres mayor de edad se ocultará este aviso.

Veremos una animación de bolas del bingo por detrás del logotipo y un botón para entrar.

Selección de jugadores

Nos mostrará está página en el que podemos ver que tenemos un menú arriba en el cual podemos darle a **todos aleatorios** que significa que usará los 4 jugadores que es el limite de jugadores y todos aleatorios, luego tenemos el botón de **jugar** que es para redirigirnos ya a la partida y otro botón por si hemos salido del navegador y queremos **continuar la partida**, por último tenemos el botón de **salir**.

En cada jugador tenemos el siguiente formulario, seleccionar imagen que si le damos se nos abrirá un modal con imágenes para seleccionar la que queremos.

Luego el otro campo que tenemos es para ponerle un nombre al jugador y el otro campo para los cartones.

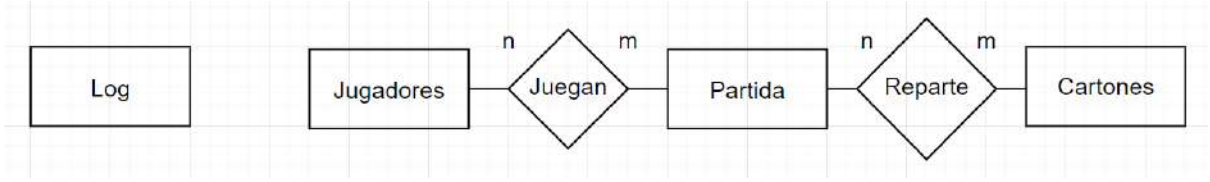
Por otro lado tenemos el botón aleatorio y este jugador nos dará todos los valores aleatorios.


JUGADOR 1
Seleccionar imagen
Javi
Cartones
2
Aleatorio

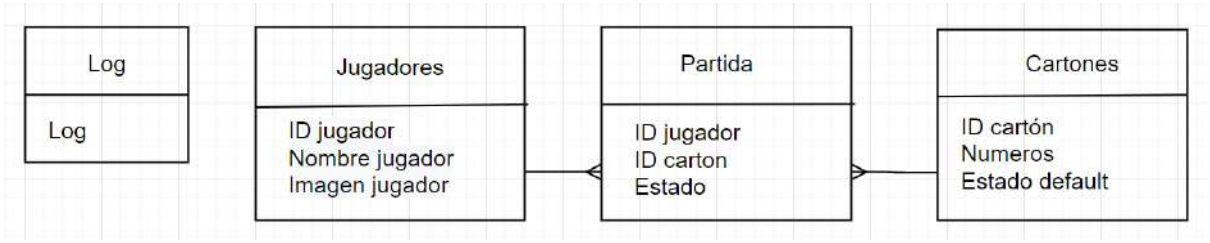
y podemos seleccionar tantos jugadores como queramos como máximo , podemos eliminar los jugadores también.

Modelos de la base de datos

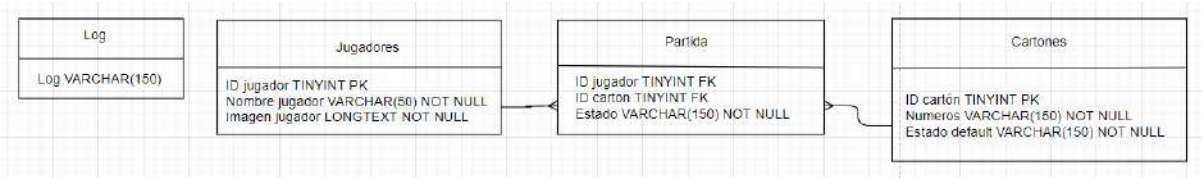
Modelo conceptual



Modelo lógico



Modelo físico



Problemas encontrados

Nuestra primera idea era crear el bingo con POO ya que a la hora de entender el funcionamiento tener separado el programa en clases ayuda a la hora de mantenimiento y entendimiento/funcionamiento, todo funcionaba bien ya que tan solo con pasarle a la clase bingo los nombres de los jugadores, el numero de cartones que habían pedido y la url de la imagen todo el funcionamiento interno del juego se preparaba y creaba automáticamente.

Entonces cuando se ejecuta la primera vez y se crea el objeto bingo todo funciona correctamente ya que en ese momento se conocen todos los valores que tienen los atributos de los objetos, el problema ocurre a la hora de recargar, ya que nuestra principal idea era no arrojar todos los datos de la partida de una vez debido a que no es una experiencia muy limpia para el usuario que la ejecute y no es "controlable".

Nuestra principal idea era que hubiera un botón de sacar bola y que se fuera jugando ronda por ronda, la idea estaba bien pero el problema que nos encontramos que es que al recargar todos los valores que tenían las clases se pierden por lo que hemos tenido que utilizar funciones normales para coger los datos que necesitábamos de la BD y utilizarlos para que pudiera funcionar el programa.

Futuras necesidades

Como hemos comentado en los problemas encontrados nos ha surgido la necesidad de aprender a como poder guardar datos,atributos etc... en PHP para poder seguir utilizando las clases sin que se pierdan sus datos o arrastrar valores directamente al recargar.