Documentación Técnica

bingo Colaborativo

- Requisitos funcionales y no funcionales.

  - [x] ~~Estar en la nube con su dominio~~
  - [x] ~~Tener un aspecto visual agradable~~
  - [x] ~~Ser desarrollada de forma colaborativa~~
  - [x] ~~Tener test unitarios~~

Se presenta un solución al problema y se documenta como se ha resuelto.

Se considera el juego bingo como una clase que contendrá las funcionalidades para ejecutar una partida de este juego

Funcionalidades destacadas:

Iniciar y ejecutar toda la secuencia del juego hasta fin : initJuego()

generar una cantidad determinada de cartones por jugador getCartones()

Nombrar a los jugadores y asignar cartones correspondientes : __construct()

Simular lanzamiento de una bola con número del 1 al 99 sin repeticiones: getBola()

Verificar en los cartones generados el match con bola sacada verifica()

Verificar existencia de línea o Bingo en el total de cartones generados getPremio()

La lógica del funcionamiento de los métodos de la clase se detalla como comentario en el código fuente.

- Principales problemas encontrados y soluciones aportadas.

- [x] ~~Almacenaje y maquetación de los cartones solicitados.~~

El cartón se genera tras la lectura de 3 líneas del fichero cartones.dat, siendo cada línea un fila del cartón, cada línea será contenida en un array, los 3 array línea serán contenidos en un array cartón,

todos los cartones estarán dentro del array totalcartones, se creará una copia exacta de él para edición y verificación de LINEA y BINGO.

```
if($ncartonesporjug == 0 || $ncartonesporjug == 4){
            throw new DomainException();
        }
        $vueltas  = $ncartonesporjug*4;
        $clavejugador = $cont = 0;
        while( $vueltas  != 0){
            $carton = [];
            if( $ncartonesporjug==1 || $ncartonesporjug==2&&$cont%2==0 || $ncartonesporjug==3&&$cont%3==0){
                echo " <section class=sectioncartones1> Propiedad de {$this->jugadores[$clavejugador]} ";
                $clavejugador++;
            }
            for($i=0; $i<3; $i++){
                $linea = fgets($this-> fichero);
                $carton []= explode( ".", $linea );
            }
            $this-> totalcartones[] = $carton;
            echo "<div class=carton>";
            foreach ($carton as $key => $value ) {
                echo "<div class=cartonfila>";
                foreach  ($carton[$key] as $clave => $valor) {
                    echo "<div class=cartonfilacelda>".$valor."</div>";
                }
                echo "</div>";
            }
            echo "</div>";
            $cont++;
            if( $ncartonesporjug==1 || $ncartonesporjug==2&&$cont%2 ==0 || $ncartonesporjug==3&&$cont%3 ==0 ){
                echo "</section>";
            }
            $vueltas--;
        }
        $this-> verificartotalcartones = [];
        $this-> verificartotalcartones = $this-> totalcartones;
```

☑ ~~Verificación de línea y bingo.~~

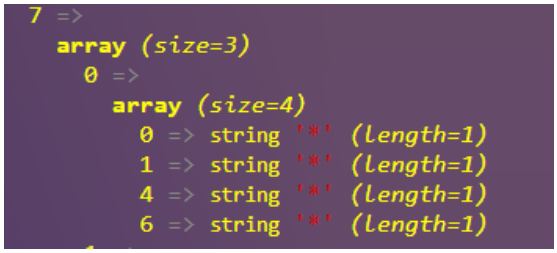
La verificación se realizará en el método verifica($bola) realiza la lectura array verificartotalcartones y si hace match numero contenido en $bola con valor numero en cartón borra clave y valor, imaginemos que ya se han borrado 5 claves y sus correspondientes valores. la función count() aplicada sobre el array línea devolverá 4 siendo estos los espacios en blanco por defecto en cada línea del cartón, si esto sucede cantaremos Línea, y parámetro lineaCantada será modificado para no volver a cantar.

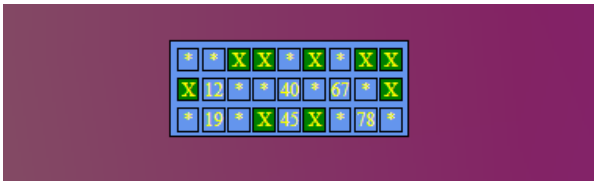
A cada verificación de fila completa correspondiente a array línea del cartón, de testea con getPremio() si es línea cantada o línea completa si se canto anteriormente.

```
public function verifica($bola){
$ncartonesporjug = count($this->totalcartones)/4;
        $clavejugador = $cont = 0;
```

```
        foreach ($this->verificartotalcartones as $key => $value) {
            foreach ($this->verificartotalcartones[$key] as $clave => $valor) {
                // var_dump($this->totalcartones[$key]);  echo"<br><br><br><br>";
                foreach ($this->verificartotalcartones[$key][$clave] as $indi => $caracter) {
                    // var_dump($this->totalcartones[$key][$clave]);  echo"<br><br><br><br>";
                    if($caracter == $bola){
                        // echo "match en carton: $key , Linea: $clave , clave: $indi , idcelda: $celda<br> <hr>";
                        $ndecarton = $key;
                        $ndecarton +=1;
                        $ndelinea = $clave;
                        $ndelinea +=1;

                        echo "Carton : ".$ndecarton." fila: ".$ndelinea;
                        echo " Numero: $caracter  -  ¡Match!</br>";

                        unset($this->verificartotalcartones[$key][$clave][$indi]);
                        $this->totalcartones[$key][$clave][$indi] = "X";
                    }
                }
          !!!! echo $this->getPremio($this->verificartotalcartones[$key][$clave]); !!!!
            }
            echo " <span class='bola1'>$bola</span></br></br> ";
        }
```

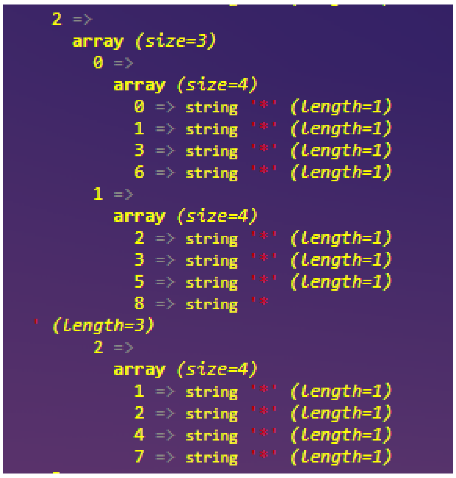
- Ejemplo de Línea

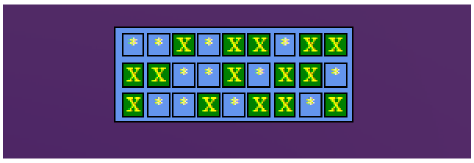




De la misma manera que para línea.. cuando se leen 3 líneas completas seguidas, OJO pertenecientes al mismo cartón se canta bingo y finaliza el juego.

- Ejemplo de Bingo





- Código del método getPremio()

```
public function getPremio(array $linea){

        if( $this->linealeida == 3 ){
            $this->linealeida = 0;
            $this->lineaCompleta = 0;
        }
        $this->linealeida++;
        $size= count($linea);
        if ($size == 4 ) {
            $this->lineaCompleta =  $this->lineaCompleta +1;
            if($this->lineaCantada == 0){
                $this->lineaCantada = 1;
                $mensaje = "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  LÍNEA  ¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡";
                return $mensaje;
            }
            if($this->linealeida == 3 && $this->lineaCompleta == 3 &&  $this-> bingoCantado == 0){
                $this-> bingoCantado = 1;
                $mensaje = "!!!!!!!!!!!!!!!!!!!!!!!   BINGOOOO   ¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡";
                return $mensaje;
            }
        }

    }
```

- [x] ~~Iluminar o colorear celdas con números coincidentes con bola.~~

- Dentro de la clase verificaBola(), tras el proceso de verificación de bola en array verificartotalcartones el array totalcartones utilizado para lo impresión de los cartones matcheados a cada ronda de bola sacada, asi que es editado en la misma posición que hace match la bola, su valor numerio se sustituye por una X.

```
foreach ($this->verificartotalcartones[$key][$clave] as $indi => $caracter) {
                    // var_dump($this->totalcartones[$key][$clave]);  echo"<br><br><br><br>";
                    if($caracter == $bola){
                        // echo "match en carton: $key , Linea: $clave , clave: $indi , idcelda: $celda<br> <hr>";
                        $ndecarton = $key;
                        $ndecarton +=1;
                        $ndelinea = $clave;
                        $ndelinea +=1;

                        echo "Carton : ".$ndecarton." fila: ".$ndelinea;
                        echo " Numero: $caracter  -  ¡Match!</br>";

                        unset($this->verificartotalcartones[$key][$clave][$indi]);
                    !!!!     $this->totalcartones[$key][$clave][$indi] = "X";   !!!!
                    }
```

- Tras la edición de todos los valores matcheados en todos los cartones el array totalcartones, se imprime y formatea para mostrar los cartones matcheados de cada jugador.

```
// Impresión de cartones Matcheados, aplica clase .celdacolor a celdas con valor matcheado en bola.
        foreach($this->totalcartones as $key => $value){
            if( $ncartonesporjug==1 || $ncartonesporjug==2&&$cont%2==0 || $ncartonesporjug==3&&$cont%3==0){
                echo " <section class='sectioncartones'> Propiedad de {$this->jugadores[$clavejugador]} ";
                $clavejugador++;
            }
            echo "<div class=carton>";
            foreach ($this->totalcartones[$key] as $clave => $valor ) {
                echo "<div class=cartonfila>";
                foreach  ($this->totalcartones[$key][$clave] as $in => $numero) {
                    if($numero == "X"){
                        echo "<div class=\"cartonfilacelda celdacolor\">".$numero."</div>";
                    }
                    else{
                        echo "<div class=cartonfilacelda>".$numero."</div>";
                    }
                }
                echo "</div>";
            }
            echo "</div>";
            $cont++;
            if( $ncartonesporjug==1 || $ncartonesporjug==2&&$cont%2 ==0 || $ncartonesporjug==3&&$cont%3 ==0 ){
                echo "</section>";
            }
        }
```

- Ejemplo de impresión de cartones marcados tras una sacada de bola, verificación y una sustitución de valores matcheados.

Se crean Clases y se instancian objetos de dichas clases.

- Las clases existentes son Bingo y Útiles

```php
<?php
    include_once('../vendor/autoload.php');
    use Moi\Bingo\Bingo;

    $nombre1=$_POST['nombre1'];
    $nombre2=$_POST['nombre2'];
    $nombre3=$_POST['nombre3'];
    $nombre4=$_POST['nombre4'];
    $ncartones=$_POST['ncartones'];

    $nuevaPartida = new Bingo($nombre1, $nombre2, $nombre3, $nombre4);

    $nuevaPartida -> getCartones($ncartones);

    $nuevaPartida -> initJuego();

    echo " <script src=js/bingo.js language=javascript type=text/javascript></script>";
?>
```

Se crean pruebas unitarias que verifican el correcto funcionamiento del código.

Test lanzados a la clase Bingo

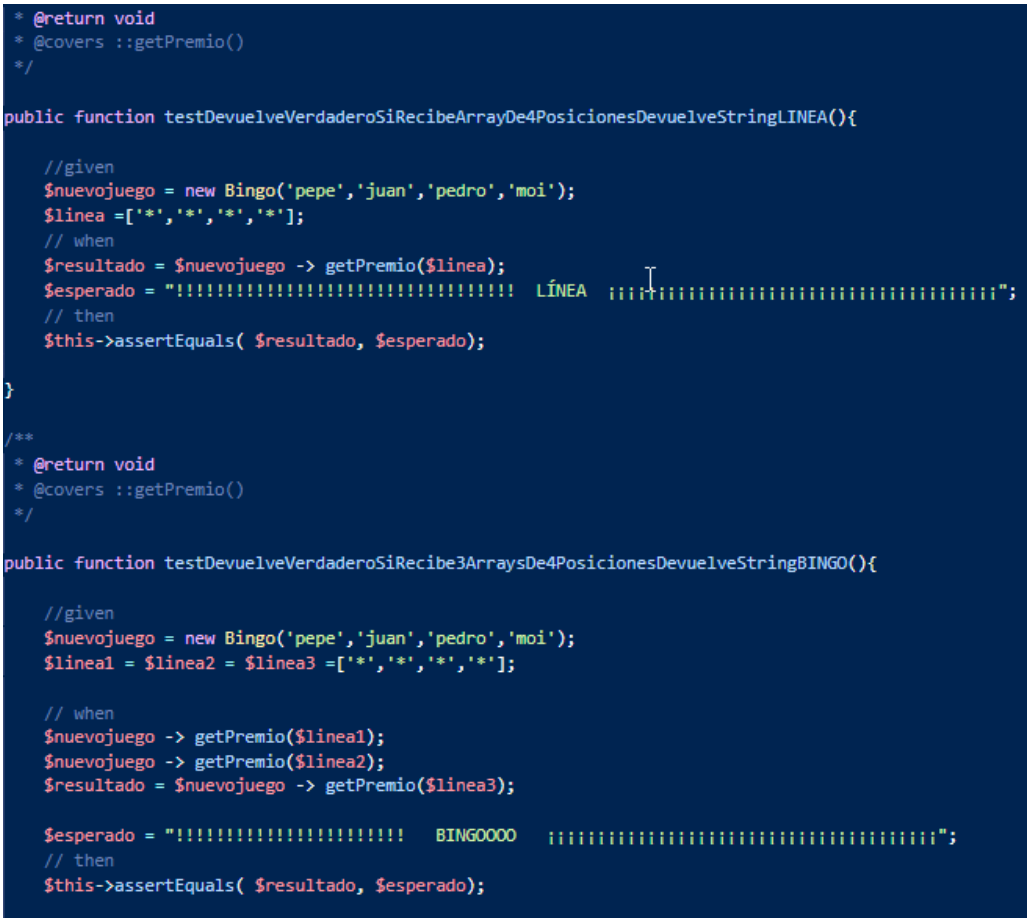
- método getBola()

Estos test verifican que la devolución del método sea un entero, y que este comprendido entre 1 y 90 ambos incluidos.

```php
namespace Moi\Tests\Bingo;
use Moi\Bingo\Bingo;
use PHPUnit\Framework\TestCase;
use DomainException;

class BingoTests extends TestCase{

    /**
     * @return void
     * @covers ::getBola()
     */
    public function testDevuelveVerdaderoSiBolaEsNumeroEntero(){
        //given
        $nuevojuego = new Bingo('pepe','juan','pedro','moi');
        // when
        $resultado = $nuevojuego -> getBola();
        // then
        $this->assertIsInt($resultado);
    }

    /**
     * @return void
     * @covers ::getBola()
     */
    public function testDevuelveVerdaderoSiBolaEsMayorOIgualA1(){

        //given
        $nuevojuego = new Bingo('pepe','juan','pedro','moi');
        // when
        $resultado = $nuevojuego -> getBola();
        // then
        $this->assertGreaterThanOrEqual(1,$resultado);

    }

     /**
     * @return void
     * @covers ::getBola()
     */
    public function testDevuelveVerdaderoSiBolaEsMenorOIgualA90(){

        //given
        $nuevojuego = new Bingo('pepe','juan','pedro','moi');
        // when
        $resultado = $nuevojuego -> getBola();
        // then
        $this->assertLessThanOrEqual(90,$resultado);

    }
```

- método getPremio()

Estos test verifican la devolución variable siendo esta de tipo string y su contenido sea LINEA!!!!... o BINGO!!!... en el caso correspondiente.



```
 * @return void
 * @covers ::getPremio()
 */

public function testDevuelveVerdaderoSiRecibeArrayDe4PosicionesDevuelveStringLINEA(){

    //given
    $nuevojuego = new Bingo('pepe','juan','pedro','moi');
    $linea =['*','*','*','*'];
    // when
    $resultado = $nuevojuego -> getPremio($linea);
    $esperado = "!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  LÍNEA  ¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡";
    // then
    $this->assertEquals( $resultado, $esperado);

}

/**
 * @return void
 * @covers ::getPremio()
 */

public function testDevuelveVerdaderoSiRecibe3ArraysDe4PosicionesDevuelveStringBINGO(){

    //given
    $nuevojuego = new Bingo('pepe','juan','pedro','moi');
    $linea1 = $linea2 = $linea3 =['*','*','*','*'];

    // when
    $nuevojuego -> getPremio($linea1);
    $nuevojuego -> getPremio($linea2);
    $resultado = $nuevojuego -> getPremio($linea3);

    $esperado = "!!!!!!!!!!!!!!!!!!!!!!!!   BINGOOOO   ¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡";
    // then
    $this->assertEquals( $resultado, $esperado);
```

- método getCartones()

Estos test verifican el arrojo de Excepción si los cartones solicitados son 0 o 4 para cada jugador valores inválidos.

```php
 * @return void
 * @covers ::getCartones()
 */
public function testDevuelveDomainExceptionSiCartonesSon0PorJugador(){

    $this -> expectException(DomainException::class);
    //given
    $nuevojuego = new Bingo('pepe','juan','pedro','moi');

    // when then
    $nuevojuego -> getCartones(0);

}
/**
 * @return void
 * @covers ::getCartones()
 */
public function testDevuelveDomainExceptionSiCartonesSon4PorJugador(){

    $this -> expectException(DomainException::class);
    //given
    $nuevojuego = new Bingo('pepe','juan','pedro','moi');

    // when then
    $nuevojuego -> getCartones(4);

}
```