

Documentación técnica magic

Proyecto

▼ Índice

- ▼ Programación basada en orientación a objetos.
- ▼ RWD.
- ▼ Batería de pruebas automatizadas.
- ▼ Modelado y diseño de base de datos.
- ▼ Presentación y maquetado profesional.
- ▼ Uso de herramientas de control de versiones para desarrollo del proyecto desde inicio.
- ▼ Publicación y puesta en producción en el subdominio magic.tudominio.ddd
- ▼ Estará debidamente documentado con bloques de documentación entendibles por phpDocumentor.
- ▼ La documentación del código estará generada y desplegada en producción 02-docphp/

- ▼ Debe tener un pull de base de 10 cartas diferentes.
- ▼ Poder dar de alta nuevas cartas a nuestro mazo.
- ▼ Saber cuantas cartas en total tenemos.
- ▼ Saber cuantas cartas diferentes tenemos.
- ▼ Para presentar las cartas, se visualizarán con un thumbnail y su nombre, al hacer clic se mostrará carta Magic.
- ▼ La cartas aparecerán, de primeras, ordenadas según esté establecido en la base de datos, pudiendo el usuario ordenarlas por nombre (asc/desc), por el orden establecido (asc/desc) y tipo (asc/desc), sin recargar la página.

Requisitos completados

No funcionales

- Programación basada en orientación a objetos.
- Modelado y diseño de base de datos.
- Presentación y maquetado profesional
- Batería de pruebas automatizadas.
- RWD.
- Uso de herramientas de control de versiones para desarrollo del proyecto desde inicio.

- Publicación y puesta en producción en el subdominio magic.newflow.tech
- Estará debidamente documentado con bloques de documentación entendibles por phpDocumentor.
- La documentación del código estará generada y desplegada en producción 02-decphp/

Funcionales

- Debe tener un pull de base de 10 cartas diferentes.
- Poder dar de alta nuevas cartas a nuestro mazo
- Saber cuantas cartas en total tenemos.
- Saber cuantas cartas diferentes tenemos.
- Para presentar las cartas, se visualizarán con un thumbnail y su nombre, al hacer clic se mostrará carta Magic.
- Las cartas aparecerán, de primeras, ordenadas según esté establecido en la base de datos, pudiendo el usuario ordenarlas por nombre (asc/desc), por el orden establecido (asc/desc) y tipo (asc/desc), sin recargar la página.

Soluciones aportadas y resolución de las mismas

En la historia de magic existen muchas colecciones de cartas, estas se lanzan al mercado marcadas o identificadas con un logotipo llamado expansión, en cada una de ellas las cartas tienen ciertas particularidades en su formato, esta aplicación está siendo desarrollada para mostrar lo más fielmente la expansión Conflux, no quiere decir que tenga punto y final esta será la primera versión y será diseñada tanto la base de datos, como todos ficheros para la incorporación de nuevas funcionalidades y nuevas colecciones de cartas.

Modelado y diseño de base de datos.

Las magics contienen multitud detalles que a continuación serán interpretados a favor del diseño de una óptima base de datos normalizada y estructurada.

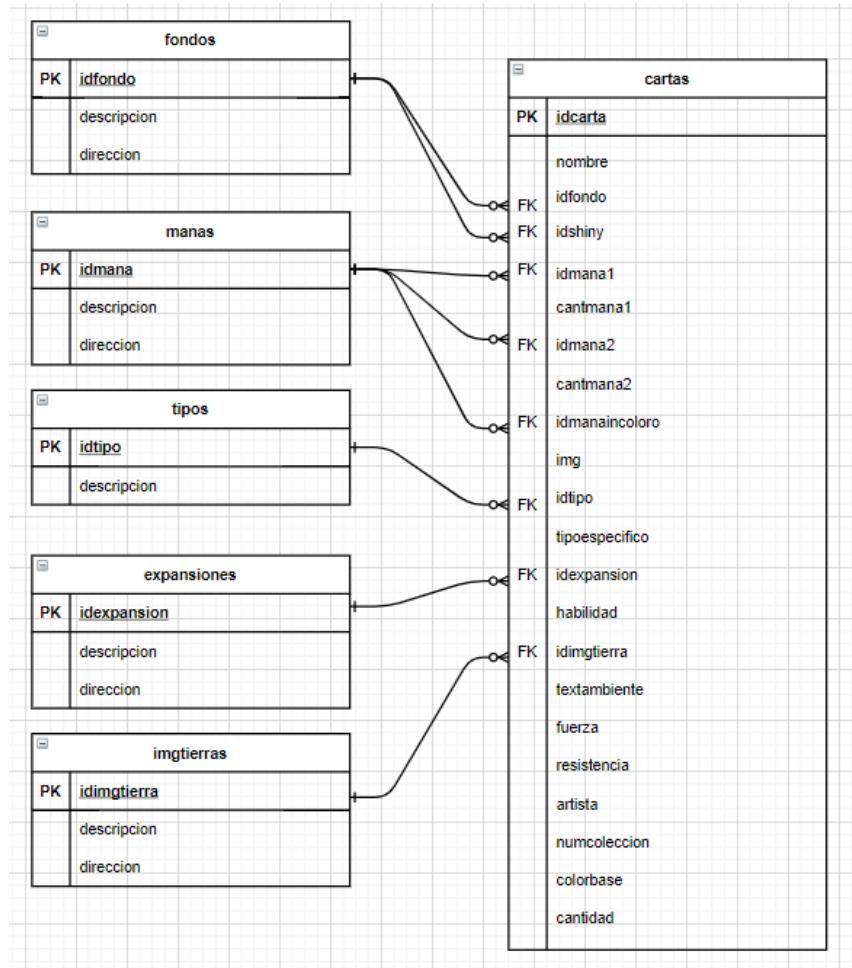
Considero la entidad principal carta ya que cada magic es única y genuina por la ilustración única que desvelan, son útiles en cualquier baraja pertenezcan a la expansión que pertenezcan.

Los fondos, manas, tipos, expansiones e imágenes para las carta tierra, serán comunes la totalidad de las cartas sean de cualquier expansión por lo que normalizo estos datos en una tabla cada concepto.

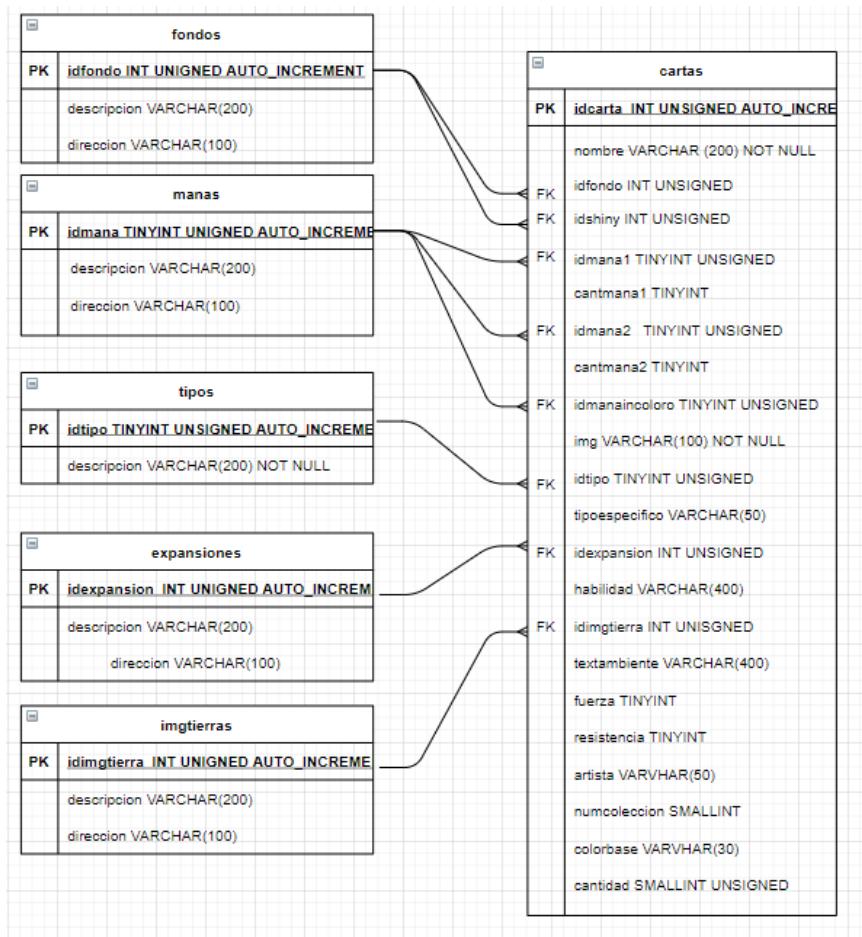
- Modelo Conceptual



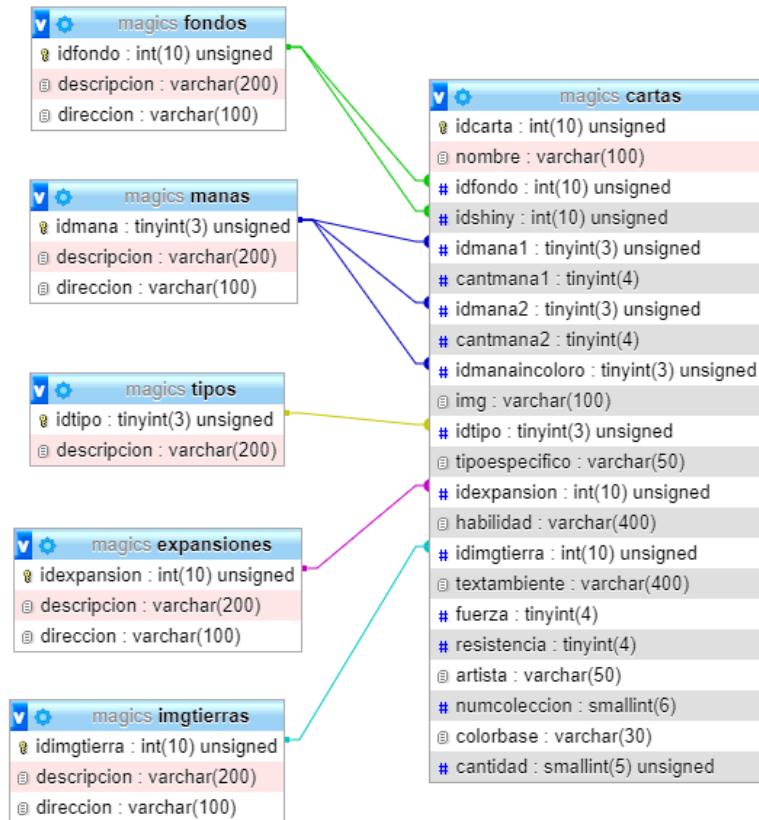
- Modelo Lógico



- Modelo Físico



- Visión del diseñador una vez creada la base de datos.



Programación basada en orientación a objetos.

La carta en la aplicación será tratada como un objeto, para esto dispondremos una clase llamada Carta que aceptará como parámetros todos los posibles datos que pueda tener una carta.

```

public function __construct(
    $idcarta, $nombre, $fondo, $shiny, $mana1, $cantmana1, $mana2, $cantmana2, $cantmanainc,
    $img, $tipo, $tipoespecifico, $expansion, $habilidad, $imgtierra, $textambiente, $fuerza,
    $resistencia, $artista, $numcolección, $colorbase, $cantidad)
{
    $this-> idcarta = $idcarta; $this-> nombre = $nombre; $this-> fondo = $fondo;
    $this-> shiny = $shiny;
    $this-> mana1 = $mana1; $this-> cantmana1 = $cantmana1; $this-> mana2 = $mana2;
    $this-> cantmana2 = $cantmana2; $this-> cantmanainc = $cantmanainc; $this-> img = $img;
    $this-> tipo = $tipo; $this-> tipoespecifico = $tipoespecifico;
    $this-> expansion = $expansion; $this-> habilidad = $habilidad; $this-> imgtierra = $imgtierra;
    $this-> textambiente = $textambiente; $this-> fuerza = $fuerza; $this-> resistencia = $resistencia;
    $this-> artista = $artista; $this-> numcolección = $numcolección; $this-> colorbase = $colorbase;
    $this-> cantidad = $cantidad;
}

```

- Dispondrá de un método que imprima la estructura html que soportará todos los datos de la carta, considero la carta como un artículo completo ya que cada una tiene sentido por si misma no necesita de las demás, será formado por un encabezado varias secciones y un pie de carta

El encabezado mostrará el nombre y el costo de mana de la carta.

```
<header class=encabezadocarta borderojo style=background-color:{$this->colorbase}>
    <h1> {$this-> nombre} </h1>
    <div>;
        if($this-> cantmanainc != ""){
            echo "<img class=imgmana src={$this-> cantmanainc}>";
        }
        if($this-> mana1 != ""){
            for($i=0; $i<$this-> cantmana1; $i++){
                echo "<img class=imgmana src={$this-> mana1}>";
            }
        }
        if($this-> mana2 != ""){
            for($i=0; $i<$this-> cantmana2; $i++){
                echo "<img class=imgmana src={$this-> mana2}>";
            }
        }
        echo "
    </div>
</header>
```

Primera sección contiene imagen.

```
<section class=seccionimagen>
    <img class=imgcarta src={$this-> img}>
</section>
```

Segunda sección contiene tipo y tipo específico de la carta, tendrá el color de fondo especificado, si contiene tipo específico imprimiremos un guión para diferenciación.

```
<section class=seccióntiposubtipocarta style=background-color:{$this->colorbase} >
    <ol>
        <li class=itemtipo>
            {$this-> tipo}
        </li>;
        if($this-> tipoespecifico ){
            echo "-";
        }
        echo "
        <li class=itemsubtipo>
            {$this-> tipoespecifico}
        </li>
    </ol>
    <img class=imgexpansion src={$this-> expansion}>
</section>
```

Tercera sección habilidades de la carta y texto de ambientación, tendrá el color de fondo especificado.

```
<section class=seccióndescripcióncarta style=background-color:{$this->colorbase}>
    <ol>
        <li class=itemhabilidad>
            {$this-> habilidad}
        </li>
```

```

        <li class=itemtextambiente>
            {$this-> textambiente}
        </li>
    </ol>
</section>

```

Por último existirán dos footer imprimiremos uno u otro según el tipo que sea la carta, el primer footer tendrá estructura para fuerza y resistencia, el segundo no.

Esta estructura también tendrá el color de fondo especificado.

```

if( $this->tipo !=="Instantáneo"&&$this->tipo!=="Encantamiento"
    &&$this->tipo!=="Conjuro"&&$this->tipo!=="Artefacto"&&$this->tipo!=="Tierra" ){
    echo "
        <footer class=piecarta>
            <section class=seccionfuerzaresistencia style=background-color:{$this->colorbase} >
                <ol>
                    <li class=itemfuerza>
                        {$this-> fuerza}
                    </li>
                    /
                    <li class=itemresistencia>
                        {$this-> resistencia}
                    </li>
                </ol>
            </section>
            <ol>
                <li class=itemabilidad>
                    {$this-> artista}
                </li>
                <li class=itemnumcolección>
                    <p>numero:{$this-> numcolección}</p>
                </li>
            </ol>
        </footer>
    ";
}
else{
    echo "
        <footer class=piecartav2>
            <ol>
                <li class=itemabilidad>
                    {$this-> artista}
                </li>
                <li class=itemnumcolección>
                    <p>numero:{$this-> numcolección}</p>
                </li>
            </ol>
        </footer>
    ";
}

```

- Todas estas estructuras irán dentro de una clase articulo, esta a su vez en un contendor para el fondo, y otro para el borde, al existir tanta cantidad de fondos posibles y colores y formatos de bordes veo al necesidad de crear esta estructura específica para ellos.

Cada borde contendrá el id de la carta en negativo, para su posterior vinculación con el thumbnail correspondiente.

```

<div class=bordecarta id=$idneg >
    <div class=fondocarta style=background-image:url({$this-> fondo})>
        <article class=articuloarticulo>

```

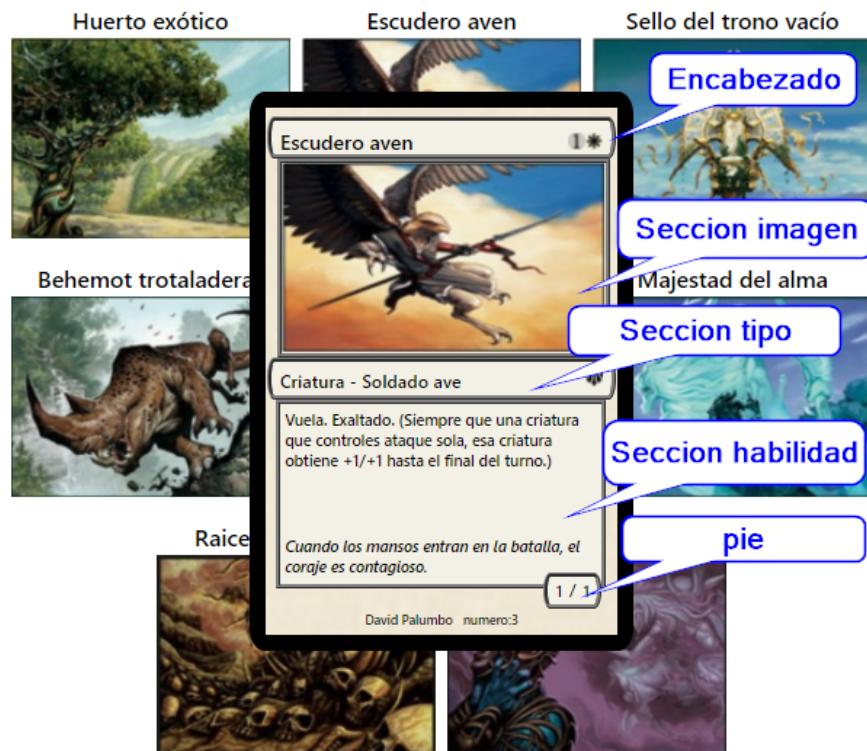
```
// estructuras del articulo  
  
    </article>  
  </div>  
</div>
```

- También habrá un método que imprima el thumbnail solicitado, su sección los atributos "idcarta", para vinculación con artículo carta, "name" nombre de la carta será usado para ordenar por nombre, "tipo" tipo de la carta para el orden por tipo y atributo "cantidad" para mostrar la cantidad de cada carta disponible en nuestra base de datos.

```
<section class=thumbnail id={$this-> idcarta} name={$this->nombre} tipo={$this->tipo} cantidad={$this-> cantidad}>  
  
  <header class=headerthumbnail>  
    <h1> {$this->nombre} </h1>  
  </header>  
  <main>  
    <img class=imgthumbnail src={$this->img} alt={$this->nombre}>  
  </main>  
  
</section>
```

Presentación y maquetado profesional.

La maquetación de la carta se realiza con las etiquetas `<article>`, `<header>`, `<section>` y `<footer>` ya que cada carta representa una composición completa, independiente de la demás y a su vez relacionada ya sea por expansión, color..



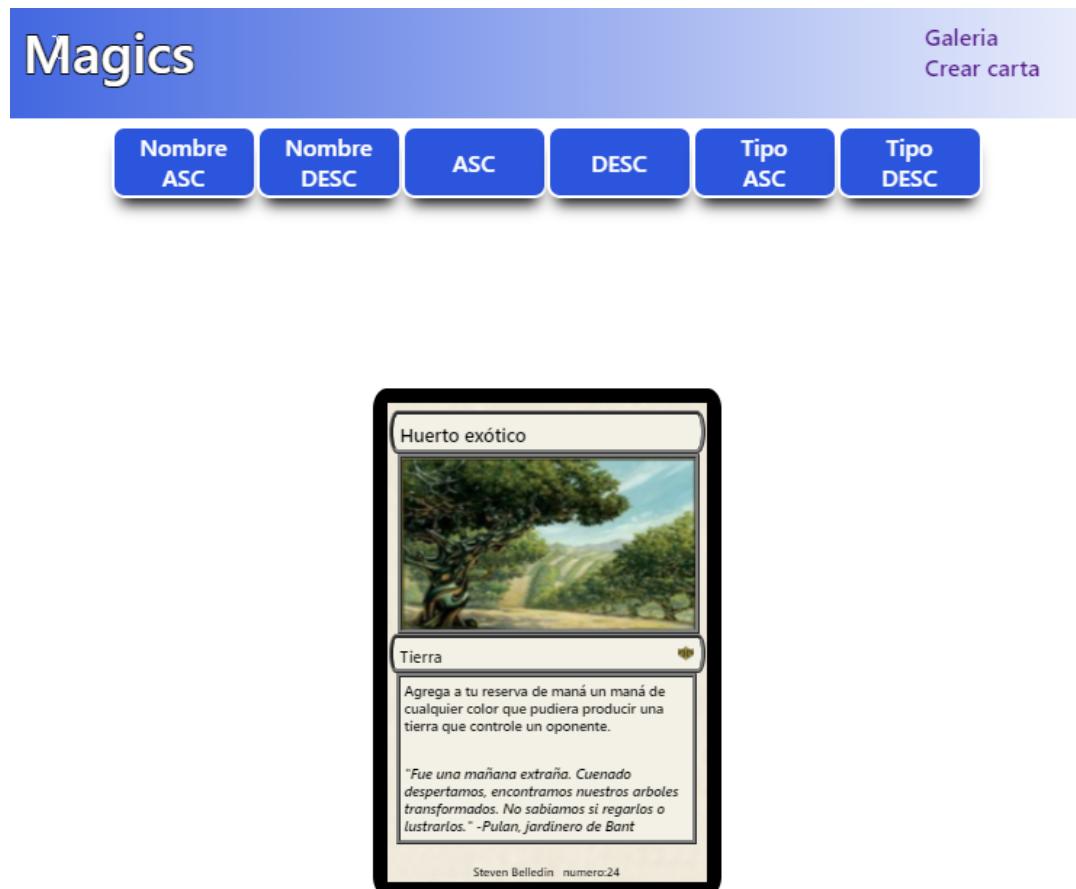
Nota: nuevos estilos aplicados: marco del thumbnails, fondo shiny, nueva visualización al click sobre thumbnails.

Magics

Galería Crear carta

Nombre ASC	Nombre DESC	ASC	DESC	Tipo ASC	Tipo DESC
Strix parasitario	Piedra obstructora	Escudero aven	Minotauro del desfiladero		
Carácter sombrío	Rabia maníaca	Nacatl indómito	Actuación de telemín		

- Al clicar thumbnail.
-

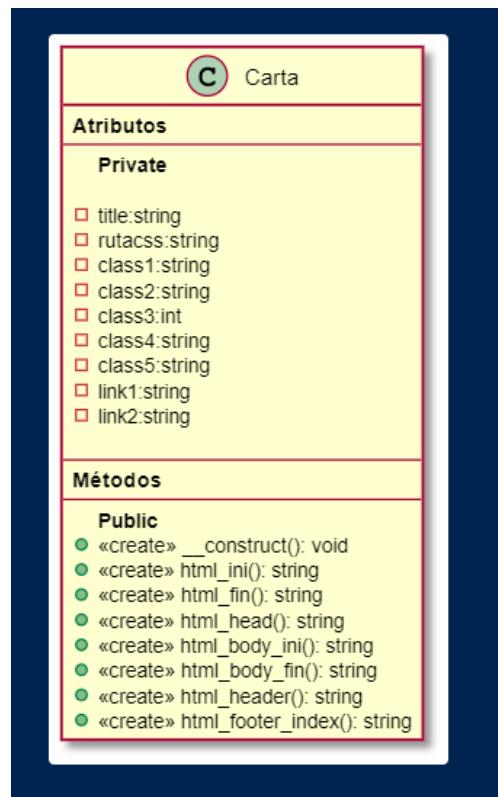


- Diagramas de clases existentes en el proyecto:

▼ Todas las cartas objeto tendrán todas los atributos disponibles en la base de datos que las describen, esta clase tendrá un constructor, un método `__toString()`. método imprime que imprimirá la estructura html que contendrá los datos de la carta, y `thumbnail`, para imprimir estructura thumbnail y los datos que corresponden a este.

Carta	Utiles
- idcarta int - nombre - fondo - shiny - mana1 - cantmana1 int - mana2 - cantmana2 - cantmanainc - img - tipo - tipoespecifico - expansion - imgtierra - textambiente - fuerza - resistencia - artista - numcolección - colorbase - cantidad	- title - rutacss - class1 - class2 - class3 - class4 - class5 - link1 - link2
+ __construct(): void + __toString(): string + imprime(): string + thumbnail(): string	+ __construct(): void + html_ini(): string +html_fin(): string +html_head(): string + html_body_ini(): string + html_body_fin(): string + html_header(): string + html_footer_index(): string

- ▼ La clase útiles contiene como atributos todos los necesarios para las partes estructurales html que genera con sus métodos, dispone de 7 métodos que imprime cada estructura html a la que refiere con su nombre.



Activar

Batería de pruebas automatizadas.

- Seguimos con la clase Carta crearé un método que retorne cada parámetro, con el fin de testear el tipo de dato que contiene la Clase es correcto.

```
public function idcarta(){
    return $this->idcarta;
}
public function nombre(){
    return $this->nombre;
}
public function fondo(){
    return $this->fondo;
}
public function shiny(){
    return $this->shiny;
}
public function mana1(){
    return $this->mana1;
}
public function cantmanaa1(){
    return $this->cantmanaa1;
}
....
```

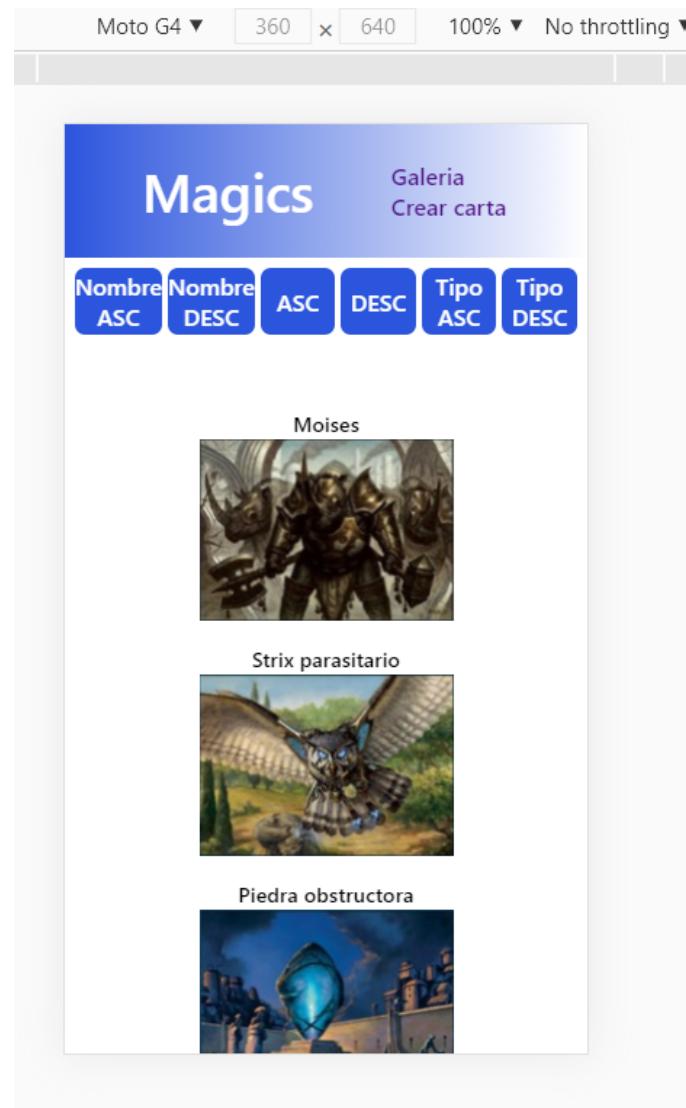
- En la carpeta tests crearé un fichero llamado CartaTest.php para testear la clase Carta.php podemos contemplar un ejemplo de prueba unitaria, será de la misma manera para cada una de la propiedades del objeto.

```
Class CartaTest extends TestCase{
    /**
     * @covers Carta::idcarta
     */
    public function testDevuelveTrueSiIdcartaEsNumeroEntero(){
        $sut = new Carta(100,'Dragón Prueba','6','1','15',2,'1',0,'4',
        'img/dibujo/Dragónvoraz.png','2','Dragón','3',
        'Vuela. Devorar 1.(En cuanto esto entra en juego puedes sacrificar cualquiera cantidad de criaturas.
        Esta criatura entra en juego con esa cantidad de contadores +1/+1 sobre ella.)'
        Cuando el Dragón voraz entre en juego, hace daño a la criatura o jugador objetivo igual
        al doble del número de Trasgos que devoró.'1','',
        4,4,'Dominick Domingo',75,'#f4e1d2',1);
        $ret= $sut->idcarta();
        $this->assertIsInt($ret);
    }
}
```

RWD

la aplicación es totalmente responsive, de adapta a las mas comunes resoluciones del mercado, para este fin se utiliza en la hoja de estilos flexbox, unidades de medida rem y em, posiciones absolutas y relativas, la propiedad box-sizing: border-box;

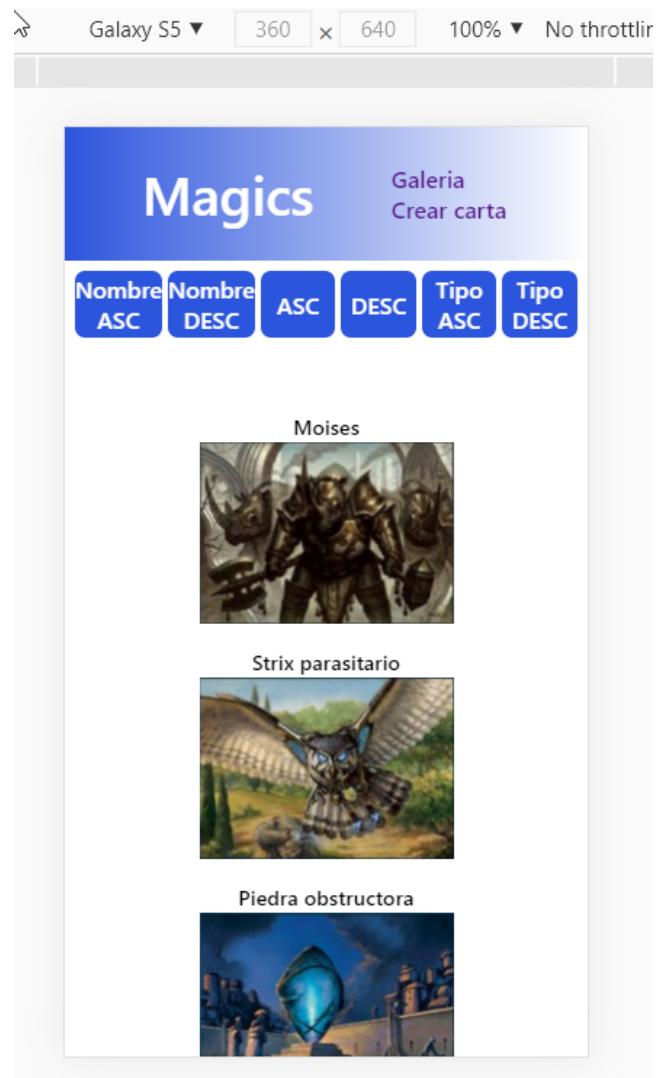
Podemos ver ejemplos de la visualización en diferentes resoluciones:



Moto G4 ▾ | 360 | x | 640 | 100% ▾ No throttling ▾

Conteo de cartas	Total de cartas	Total de cartas diferentes
	cartas: NaN	20





iPhone 5/SE ▾ 320 x 568 100% ▾ No throttling

Magics

Galeria
Crear carta

Nombre ASC Nombre DESC ASC DESC Tipo ASC Tipo DESC

Moises

Strix parasitario

Piedra obstructora

Magics

Galería
Crear carta

Nombre ASC	Nombre DESC	ASC	DESC	Tipo ASC	Tipo DESC
Moises	Strix parasitario	Piedra obstructora			
Huerto exótico	Escudero aven	Sello del trono vacío			
Minotauro del desfiladero	Carácter sombrío	Rabia maníaca			
Nacatl indómito	Actuación de telemín	Dragón voraz			
Behemot trolaladeras	Cantante solar cyliana	Majestad del alma			

iPad Pro ▾ 1024 x 1366 50% ▾ No throttling ▾

Magics

Galería
Crear carta

Nombre ASC Nombre DESC ASC DESC Tipo ASC Tipo DESC

Moises	Strix parasitario	Piedra obstrutora	Huerto exótico	Escudero aven
Sello del trono vacío	Minotauro del desfladero	Carácter sombrío	Rabla maníaca	Nacatl indómito
Actuación de telemín	Dragón voraz	Beheimot trotaladeras	Cantante solar cylana	Majestad del alma
Camino al exilio	Frontera inestable	Ziggurat antiguo	Raíces corrompidas	Voces del vacío

Conteo de cartas Total de cartas: NaN Total de cartas diferentes: 20

iPad Pro ▾ 1024 x 1366 50% ▾ No throttling ▾

Magics

Galería
Crear carta

Los campos no requeridos son color azul

Nueva carta
Nombre:

Fondo
 Sin fondo
 azul
 rojo
 negro
 blanco
 verde

Color base
 azul claro
 blanco claro
 rojo claro
 verde claro
 negro claro

Shiny
 No
 Si

Mana 1
 Sin mana
 azul
 rojo
 verde
 negro
 blanco
 Cantidad:

Mana 2
 Sin mana
 azul
 rojo
 verde
 negro
 blanco
 Cantidad:

Mana incoloro
 Cantidad:

Imagen
 Seleccionar archivo | Ningún...ionado

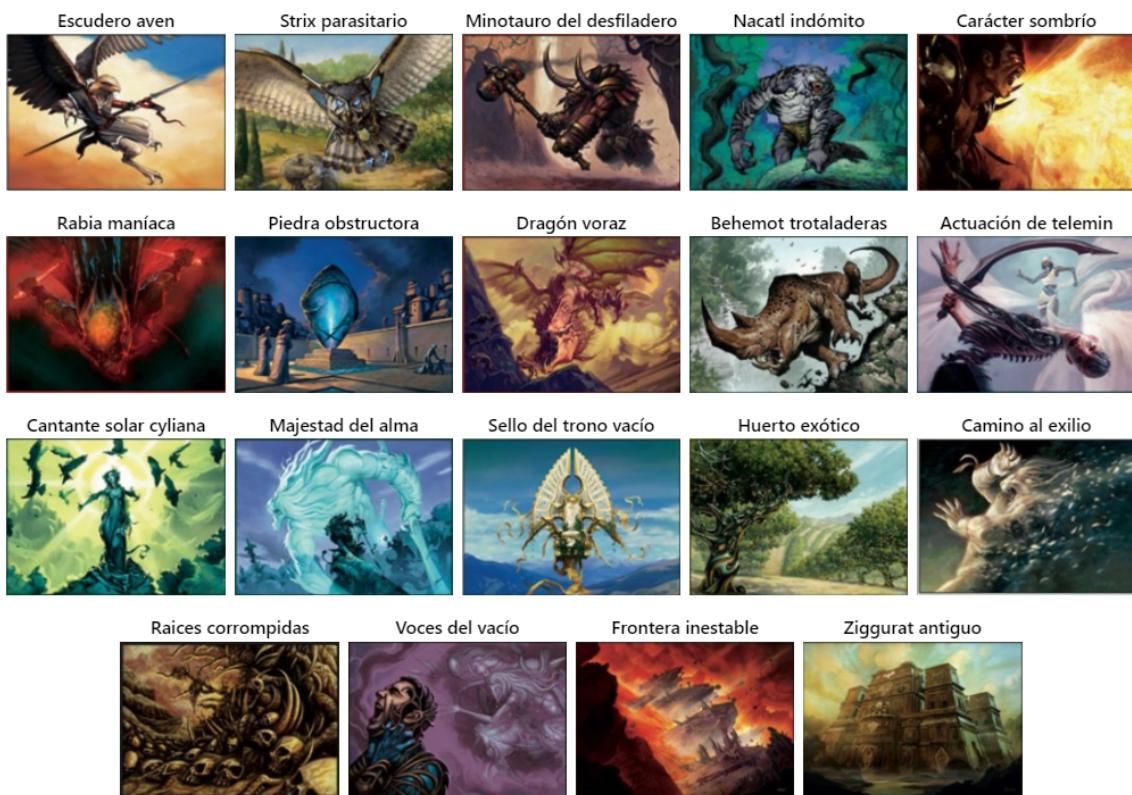
Estará debidamente documentado con bloques de documentación entendibles por phpDocumentor.

```
$ cd /var/www/html/pr102
$ phpDocumentor -d ./src/ -t ./src/02-docphp/
```

```
moimu@ip-172-31-85-184:/var/www/html/pr102/src/02-docphp$ ll
total 52
drwxrwxr-x 11 moimu www-data 4096 may 23 01:26 .
drwxrwxr-x  9 moimu www-data 4096 may 22 23:47 ../
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 classes/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:25 css/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 files/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 graphs/
-rw-rw-r--  1 moimu moimu  5287 may 23 01:14 index.html
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 indices/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 js/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 namespaces/
drwxr-xr-x  2 moimu moimu  4096 may 23 01:14 packages/
drwxr-xr-x  2 moimu moimu 4096 may 23 01:14 reports/
```

Debe tener un pull de base de 10 cartas diferentes.

La aplicación posee de base un pull de 19 cartas diferentes, entre ellas hay criaturas, encantamientos, hechizos, artefactos, instantáneos, tierras.



Conteo de cartas Total de cartas: 19 Total de cartas diferentes: 19

Saber cuantas cartas en total y cuantas cartas diferentes tenemos.

- Calculo del total de cartas existentes: recorremos todos los thumbnails y sumamos todas las cantidades de cartas existentes (atributo creado en etiqueta <section> del thumbnail) para obtener el total.
- Calculo del total de cartas diferentes, recorremos todos los thumbnails y sumamos 1 por cada uno recorrido.

```
let totalcartas = 0;
let totalcartasdif = 0;
thumbnail.forEach( function(thumb){
    totalcartas = totalcartas + parseInt( thumb.getAttribute("cantidad") , 10);
    totalcartasdif++;
});
```

- Mostramos por pantalla principal a través del pie de nuestra página en una sección dedicada, devolveremos al documento mediante función JavaScript writeln() el contenido de las variables anteriormente obtenidas.

```
<section>
    <header> <h1> Conteo de cartas </h1> </header>
    <p>Total de cartas: <script> document.writeln(totalcartas); </script> </p>
    <p>Total de cartas diferentes: <script> document.writeln(totalcartasdif); </script> </p>
</section>
```

- Resultado tras aplicar estilo

Conteo de cartas Total de cartas: 19 Total de cartas diferentes: 19

Poder dar de alta nuevas cartas a nuestro mazo

- El alta a la base de datos de nuevas cartas se realizará mediante un formulario crearcarta.php podremos acceder a él a través de menú de navegación de la web, este fichero mediante un formulario devolverá

los datos recogidos tales como nombre de carta, mana, cantidades, la imagen de la carta, puntos de fuerza y resistencia entre otros a insertarcarta.php.

```
<main class="maincrearcarta">
    <form class="formcrearcarta" name="crear-carta" method="post" enctype="multipart/form-data" action="insertarcarta.php">
        <p class="notrequired nota"> Los campos no requeridos son color azul </p>

        <fieldset>
            <legend>Nueva carta</legend>
            <div><label> Nombre: <input type="text" name="nombre" maxlength="100" required></label></div>
        </fieldset>
        <fieldset class="typeradio">
            <legend>Fondo</legend>

            <div><label> Sin fondo <input type="radio" name="idfondo" value="1" required></label></div>
            <div> <label> azul <input type="radio" name="idfondo" value="3" required></label></div>
            <div> <label> rojo <input type="radio" name="idfondo" value="6" required></label></div>
            <div> <label> negro <input type="radio" name="idfondo" value="5" required></label></div>
            <div> <label> blanco <input type="radio" name="idfondo" value="4" required></label></div>
            <div> <label> verde <input type="radio" name="idfondo" value="7" required></label></div>

        </fieldset>
        <fieldset class="typeradio">
            <legend>Color base</legend>

            <div><label> azul claro <input type="radio" name="colorbase" value="#d5dfe9" required></label></div>
            <div><label> blanco claro <input type="radio" name="colorbase" value="#f3f1e6" required></label></div>
            <div> <label> rojo claro <input type="radio" name="colorbase" value="#f4e1d2" required></label></div>
            <div> <label> verde claro <input type="radio" name="colorbase" value="#d6e7d4" required></label></div>
            <div> <label> negro claro <input type="radio" name="colorbase" value="#eae7e0" required></label></div>

        </fieldset>
    ...

```

- Visualización con estilos.

Los campos no requeridos son color azul

Nueva carta

Nombre:

Fondo

- Sin fondo
- azul
- rojo
- negro
- blanco
- verde

Color base

- azul claro
- blanco claro
- rojo claro
- verde claro
- negro claro

fuerza y resistencia

fuerza 0 resistencia 0

Artista y colección

Nombre artista

Nº colección 1

Cantidad de cartas a crear

Cantidad 1

Crear carta

- Desde el fichero insertarcarta.php recogemos datos del formulario...

```
<?php
$nombre = $_POST["nombre"];
$idfondo = intval($_POST["idfondo"]);
$idshiny = intval($_POST["idshiny"]);
$idmana1 = intval($_POST["idmana1"]);
$cantmana1 = intval($_POST["cantmana1"]);
$idmana2 = intval($_POST["idmana2"]);
$cantmana2 = intval($_POST["cantmana2"]);
$idmanaincoloro = intval($_POST["idmanaincoloro"] +1);
$idtipo = intval($_POST["idtipo"]);
```

- Recogemos nombre imagen, imagen y la enviamos a la carpeta img/ desde donde nuestra aplicación la leerá.

```
// nombre de la imagen
$nombreimg = $_FILES["imagen"]["name"];
// fichero imagen
$archivo = $_FILES["imagen"]["tmp_name"];
// ruta relativa destino server lamp
$destino = './img/dibujo/'.$nombreimg;
// para insertar ruta destino en base de datos
$img = "img/dibujo/".$nombreimg;
// mover fichero imagen a ruta en servidor
move_uploaded_file($archivo, $destino);
```

- Inserción en la base de datos y redirección al index.php para visualizar la nueva inserción.

```
$sentencia = $db->prepare("
INSERT INTO `cartas`
(`nombre`, `idfondo`, `idshiny`, `idmana1`, `cantmana1`, `idmana2`,
`cantmana2`, `idmanaincoloro`, `img`, `idtipo`, `tipoespecifico`,
`idexpansion`, `habilidad`, `idimgtierra`, `textambiente`, `fuerza`,
`resistencia`, `artista`, `numcoleccion`, `colorbase`, `cantidad`)
VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
);
$sentencia ->bind_param('siiiiisisisisisi', $param1,
.....);
$sentencia -> execute();
$db->close();
header('location:index.php');
```

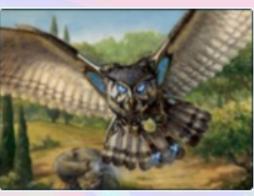
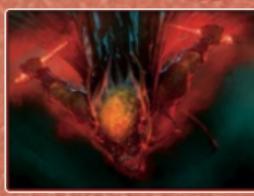
Visualización con un thumbnail y nombre, al hacer clic se mostrará la carta.

NOTA: A partir de este punto las capturas de pantalla de la aplicación mostrarán cambios de estilo, la parte estructural y funcional no es alterada.

- Todas las cartas están en la pantalla solo que ocultas con la propiedad css opacity y cada carta esta vinculado su thumbnail correspondiente con su mismo nombre de carta y misma imagen, esto se hace asignado a la estructura thumbnail un id="" , el idcarta devuelto por la base de datos, y a la estructura carta el mismo id pero negativo, de esta manera para thumbnail id=9 su carta tendrá id =-9.
- Visualización estandar.

Magics

Galeria
Crear carta

Nombre ASC	Nombre DESC	ASC	DESC	Tipo ASC	Tipo DESC
Strix parasitario	Piedra obstructora	Escudero aven	Minotauro del desfiladero		
					
Carácter sombrío	Rabia maníaca	Nacatl indómito	Actuación de telemín		
					

- Visualización al clicar sobre thumbnail.

Magics

Galeria
Crear carta

Nombre
ASC

Nombre
DESC

ASC

DESC

Tipo
ASC

Tipo
DESC



- Al clicar sobre la carta, vuelve a estado oculto y los thumbnails se mostrarán.
- Esto se realiza mediante JavaScript de lado del cliente, primero obtenemos todos los thumbnails y todas las cartas en una constante:

```
const thumbnail = document.querySelectorAll(".thumbnail");
const cartas = document.querySelectorAll(".bordecarta");
```

- Primero crearé un función para ocultar todas las cartas, y posicionarlas debajo de los thumbnail con la propiedad z-index

```
function ocultarTodo(){
    cartas.forEach( function( carta ){
        carta.style.opacity = 0;
        carta.style.zIndex = -1;

    });
}
```

- También existirán dos funciones para ocultar y mostrar los thumbnails.

```
function ocultarThumbnails(){
    thumbnail.forEach( function( thumb ){
        thumb.style.opacity = 0;
    });
}
function mostrarThumbnails(){
    thumbnail.forEach( function( thumb ){
        thumb.style.opacity = 1;
    });
}
```

- A cada thumbnail asociaremos un evento de click que, llame a la función ocultarTodo(), y levante por encima de los thumbnails (z-index) e muestre la estructura de la carta asociada (opacity), los thumbnails serán ocultados para mejor visibilidad de la carta y para inducir al usuario a clicar sobre ella para ocultarla.

```
thumbnail.forEach(function( thumb ){
    thumb.addEventListener( 'click' , function(){
        ocultarTodo();
        cartas.forEach( function( carta ){
            if( thumb.id == carta.id*-1 ){
                carta.style.opacity = 1;
                carta.style.zIndex = 1;
            }
        });
        ocultarThumbnails();
    });
});
```

- Existe también otra asociación de evento click a las cartas, para que al clicar sobre la carta que se esté mostrando esta se oculte, y todos los thumbnails sean mostrados.

```
cartas.forEach( function(carta){
    carta.addEventListener( 'click' , function ( carta ){
        ocultarTodo();
        mostrarThumbnails();
    });
});
```

Ordenadas según esté establecido en la base de datos, pudiendo ordenar por nombre (asc/desc), orden establecido (asc/desc) y tipo (asc/desc), sin recargar la página.

- Obtención de constantes con los valores de carta y ordenaremos según requerimientos: extraemos del documento el valor de los atributos nombre y tipo de carta, se guardaran con el orden establecido devuelto por la base de datos, tras esto asignamos estos valores a las constantes correspondientes posteriormente las usaremos para asignar el orden de visualización de todos los thumbnails.

```
const names = [];
const tipos= [];
let cont = 0;
thumbnail.forEach( function(thumb){

    names[cont] = thumb.getAttribute("name");
    tipos[cont] = thumb.getAttribute("tipo");
    cont++;
});
cont=0;

const ordenEstablecidoAsc = [...names];
const ordenEstablecidoDesc = [...names].reverse();
const ordenAlfabeticoAsc = names.sort();
const ordenAlfabeticoDesc = [...ordenAlfabeticoAsc].reverse();
const ordenTipoAsc = tipos.sort();
const ordenTipoDesc = [...tipos].reverse();
```

- Creación de botones de control del orden de visualización

```
<section class="seccionbotones">
<nav>
    <button class="button" id="nomAsc"><p>Nombre</p><p> ASC</p></button>
    <button class="button" id="nomDesc"><p>Nombre</p><p> DESC</p></button>
    <button class="button" id="estAsc"> ASC</button>
    <button class="button" id="estDesc"> DESC</button>
    <button class="button" id="tipoAsc"><p>Tipo</p><p> ASC</p></button>
    <button class="button" id="tipoDesc"><p>Tipo</p><p> DESC</p></button>
</nav>
</section>
```

Para cada estructura botón generamos una constante del mismo nombre al id del botón, este id además representa semánticamente la acción que realizará el botón.

Para cada botón se asociará un evento de click, esta acción provoca recorrer la constante con los ordenes establecidos creados anteriormente, en este caso ordenEstablecidoAsc contiene los nombres en el orden ha establecer, así a cada lectura recorremos todos los thumbnails y para el que tenga el mismo nombre asignaremos la propiedad de flexbox order será igual a valor de la clave del array ordenEstablecidoAsc que corresponda en ese momento.

De igual manera se asigna este evento de click a cada botón y se asigna la propiedad order: num_correspondiente a cada thumbnail según del orden establecido en la constante array correspondiente.

```
const estAsc = document.getElementById("estAsc");
const estDesc = document.getElementById("estDesc");
const nomAsc = document.getElementById("nomAsc");
const nomDesc = document.getElementById("nomDesc");
const tipoAsc = document.getElementById("tipoAsc");
const tipoDesc = document.getElementById("tipoDesc");
```

```

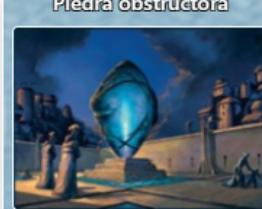
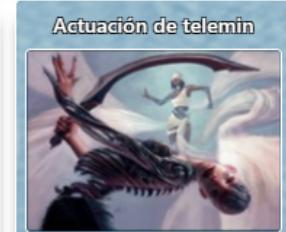
estAsc.addEventListener( 'click' , function (){
    ordenEstablecidoAsc.forEach( function(){
        thumbnail.forEach( function(thumb){
            if( thumb.getAttribute("name") == ordenEstablecidoAsc[cont]){
                thumb.style.order = cont;
                cont++;
            }
        });
    });
    cont=0;
});

```

- Orden por defecto devuelto de la base de datos:

Magics

Galeria
Crear carta

Nombre ASC	Nombre DESC	ASC	DESC	Tipo ASC	Tipo DESC		
							

- Orden por defecto devuelto de la base de datos descendente:

Magics

Galería
Crear carta

Nombre
ASC

Nombre
DESC

ASC

DESC

Tipo
ASC

Tipo
DESC



- Orden Nombre ascendente:

Magics

Galería
Crear carta

Nombre
ASC

Nombre
DESC

ASC

DESC

Tipo
ASC

Tipo
DESC



- Orden Nombre descendente:

Magics

Galeria
Crear carta

Nombre
ASC

Nombre
DESC

ASC

DESC

Tipo
ASC

Tipo
DESC



- Orden por Tipo ascendente:

Magics

Galeria
Crear carta

Nombre
ASC

Nombre
DESC

ASC

DESC

Tipo
ASC

Tipo
DESC



- Orden por Tipo descendente:

Magics

[Galeria](#)
[Crear carta](#)

Nombre ASC
Nombre DESC
ASC
DESC
Tipo ASC
Tipo DESC



Huerto exótico



Frontera inestable



Ziggurat antiguo



Carácter sombrío



Camino al exilio



Raíces corrompidas



Rabia maníaca



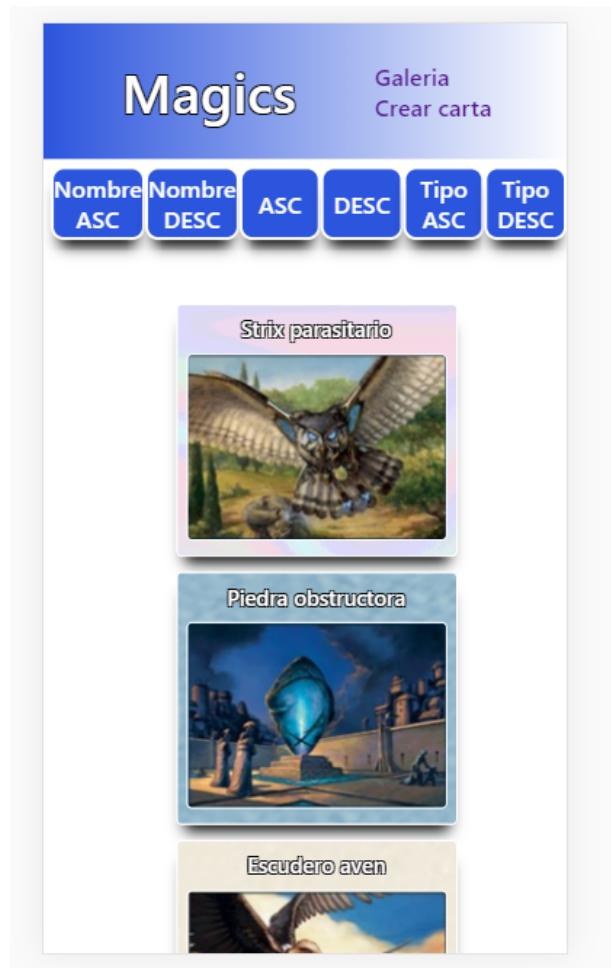
Sello del trono vacío

Optimización de carga de las imágenes.

Cuando el pull de cartas se mayor no es deseable bajar todas las imágenes desde el servidor al cliente así que html nos proporciona el atributo `loading="lazy"` este atributo especifica al navegador que vaya solicitando al servidor las imágenes según el cliente se baja acercando a ellas en el flujo de contenido de la web de esta forma se harán varias peticiones más pequeñas y mejorará la primera carga en cliente de la aplicación, pues no descargaremos 500 imágenes de cartas del tirón sino que el navegador las pedirá al servidor, a demanda de la navegación mejorando sustancialmente la fluidez percibida por el usuario.

- Dotaré a las imágenes impresas mediante la Clase Carta de este atributo

```
<img class=imgcarta src={$this-> img} alt={$this-> nombre} loading=lazy>
...
<img class=imgthumbnail src={$this->img} alt={$this->nombre} loading=lazy>
```



- En la barra inferior de Network en nuestro navegador veremos 11,5MB que fueron transferidos, en la primera solicitud a nuestro servidor, el atributo loading será utilísimo cuando la cantidad de cartas sea muy grande.
- Algunos navegadores no implementan esta nueva etiqueta, no hay problema la aplicación cargará de manera normal.

