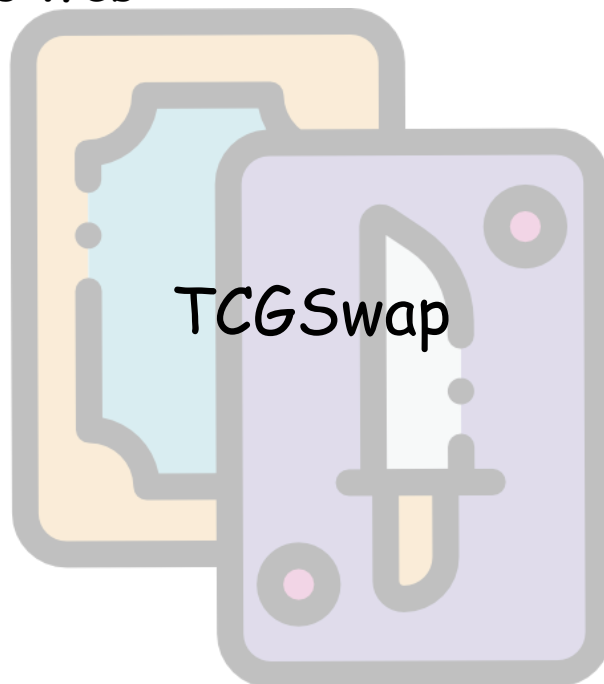




I.E.S. "ÁGORA"

Dpto.: Informática

Ciclo Formativo de G. Superior: Desarrollo de
Aplicaciones Web



Autor: Darío Estévez Bueso

Tutor: Jesús María Labrador López

Cáceres, a 21 de abril de 2022

Tabla de contenido

Definición del Proyecto.....	4
Introducción.....	4
Objetivo.....	4
Tecnologías, lenguajes y metodologías.....	6
Visual Studio Code.....	6
HTML/HTML5.....	6
CSS/CSS3.....	7
PHP.....	7
MySQL.....	8
JavaScript.....	9
jQuery.....	10
Sweet Alerts 2.....	11
BootStrap 5.....	11
MVC.....	12
Análisis de requisitos.....	13
Usuarios.....	13
Requisitos funcionales y no funcionales.....	13
Casos de uso.....	15
Diseño.....	17
Diseño de la base de datos.....	17
Diseño de la interfaz.....	18
Implementación.....	20
Codificación.....	20
Documentación.....	23
Manual de usuario.....	24
1. Landing page.....	24
2. Registro.....	25
3. Inicio de sesión.....	26
4. Portal.....	27
5. Búsqueda de cartas.....	27
6. Ver una carta, añadirla al carrito o añadirla a tu colección.....	28
7. Perfil personal.....	30
8. Mi colección.....	31

9. Mis mensajes.....	31
10. Realizar un intercambio.....	32
11. Gestionar intercambios.....	35
12. Otros usuarios: perfil, colecciones y enviar mensajes.....	37
Manual de la aplicación.....	39
1. Instalación.....	39

Definición del Proyecto

Introducción

Este proyecto surge a partir de una de mis aficiones, que es el juego **Magic: The Gathering**. Para ponerlo en contexto se trata de un **juego de cartas coleccionables** en el que dos o más jugadores juegan entre sí, cada uno de ellos con su mazo individual. Por una diversa cantidad de razones su comunidad es enorme y alrededor del juego se han creado organizaciones, competiciones profesionales, coleccionismo e incluso mercados secundarios de compraventa. Hay una cantidad ingente de cartas debido a que lleva produciéndose desde 1993 y anualmente se añaden otros cientos diferentes a las ya existentes.

Mi proyecto funciona sobre la idea de intercambiar cartas entre personas, algo que ya se realiza dentro de estas comunidades, pero solo de manera **presencial**. Yo planteo una manera de hacerlo **online** a través de un portal, permitiendo poder realizar ofertas de intercambio a cualquier persona dentro de la plataforma que tenga artículos para hacerlo.

Objetivo

El objetivo sería una aplicación que se compone de un único tipo de usuario con acceso a todas las funcionalidades detalladas a continuación:

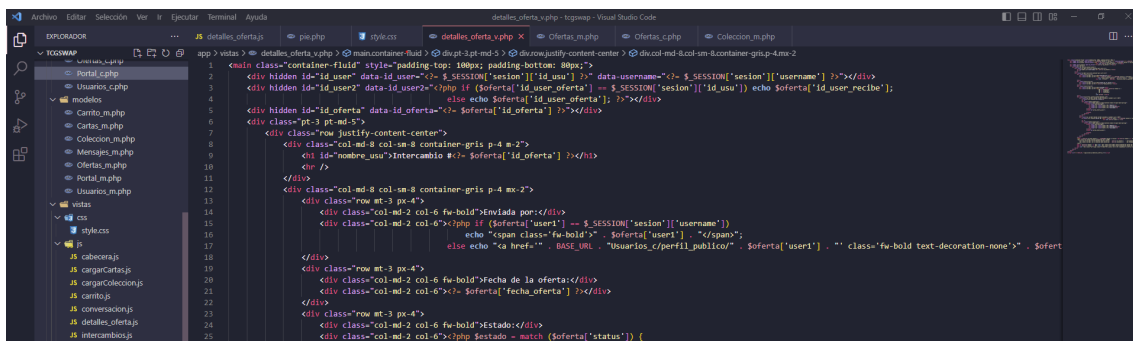
1. Ver tu perfil y tus datos.
2. Ver tu colección de cartas en un listado y filtrar los resultados.
3. Ver usuarios registrados, con su información personal y su colección de cartas.
4. Poder buscar cartas a través de una barra de búsqueda.
5. Ver cartas, listados con toda su información asociada si algún usuario dispone de ellas, filtrarlas en caso de que se busquen unas características concretas y poder añadirlas a tu colección en caso de que las tengas.
6. Poder añadir cartas de otros usuarios a tu carrito de intercambios.
7. Enviar ofertas de intercambio donde tú ofreces cartas de tu colección a cambio de las cartas de un usuario que tengas en el carrito.

8. Poder contactar con cualquier usuario a través de un sistema de mensajería donde cada conversación se registra.
9. Poder visualizar todas las ofertas, tanto recibidas como enviadas.
10. Poder visualizar una oferta y, en caso de ser una oferta recibida, aceptarla o rechazarla según tu interés.

Tecnologías, lenguajes y metodologías

Visual Studio Code

Esta aplicación se ha desarrollado sobre el editor de código **Visual Studio Code** debido a su soporte nativo para la mayoría de los lenguajes, la cantidad de extensiones que agregan funcionalidades adicionales y su depuración. Esta herramienta fue desarrollada por Microsoft y a día de hoy es una de las más utilizadas para desarrollar cualquier tipo de aplicación.



1

HTML/HTML5

Lenguaje de marcado para la creación de páginas web y estructura básica para definir el contenido de estas. En este caso utilizo la quinta revisión del lenguaje HTML, el cual añade una variedad de elementos que, aunque en esencia son similares a los ya conocidos **<div>** o ****, tienen un significado semántico. Ejemplo de esto son los elementos **<nav>**, **<footer>**, **<article>**, etc. También se añaden los atributos personalizados **'data-'**, los cuales cobran importancia a la hora de aplicar funcionalidades e interacción con JavaScript y que en este proyecto han sido utilizados muy asiduamente.

```
<div hidden id="id_user2" data-id_user2="<?php if ($oferta['id_user_oferta'] == $SESSION['sesion']['id_usu']) echo $oferta['id_user_recibe'];
else echo $oferta['id_user_oferta']; ?>"></div>
<div hidden id="id_oferta" data-id_oferta="<?php if ($oferta['id_oferta'] == $SESSION['sesion']['id_usu']) echo $oferta['id_user_recibe'];
else echo $oferta['id_user_oferta']; ?>"></div>
```

2

¹ Interfaz de usuario de Visual Studio Code.

² Ejemplo de uso de los atributos personalizados 'data-'.

CSS/CSS3

El lenguaje de hojas de estilo en cascada es el utilizado para dar forma a cualquier documento HTML. En el caso de este proyecto el CSS propio se ha utilizado puntualmente y cuando ha sido estrictamente necesario, posteriormente en esta lista mencionaré el motivo.

```
#header {
  padding: 0rem 1rem;
  background-color: #705021;
}
.busqueda {
  font-size: 16px;
  font-weight: 600;
  text-indent: 8px;
}
#login-logo,
#fondo-color {
  background-color: #ffcb7d;
}
```

3

PHP

En lo referido al Backend o lado del servidor (aunque también lo he usado en algunas situaciones en Frontend o lado del cliente) se ha utilizado PHP para acceder a la base de datos y gestionar todo lo necesario. Está planteado para generar cualquier tipo de consulta a la base de datos (insertar, leer, actualizar o borrar), además usando vinculación de parámetros de **PHP Data Objects (PDO)**, protegiéndolo así de ataques externos de inyección de SQL.

```
public function enColeccion($usu, $car, $est, $idi)
{
  $cadSQL = "SELECT * FROM coleccion_usu where id_usuario= :id_usu and id_carta= :id_car and estado= :est and idioma= :idi";
  $this->consultar($cadSQL);
  $this->enlazar(":id_usu", $usu);
  $this->enlazar(":id_car", $car);
  $this->enlazar(":est", $est);
  $this->enlazar(":idi", $idi);
  return $this->fila();
}
```

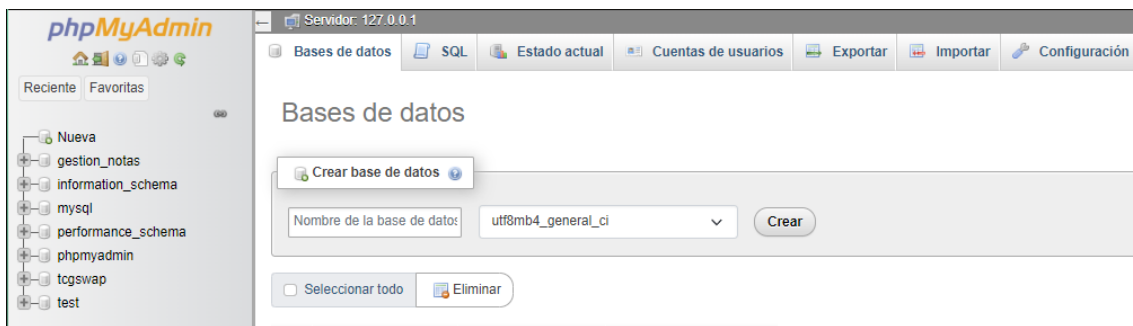
4

³ Fragmento de código CSS proveniente de mi proyecto.

⁴ Ejemplo de un método en PHP en un modelo, mostrando tanto una consulta SQL como el uso de PDO.

MySQL

A la hora de gestionar la base de datos se ha utilizado MySQL y se ha gestionado a través de la herramienta **phpMyAdmin**, donde he realizado todo lo necesario para diseñar la base de datos, sus tablas, sus relaciones, sus disparadores, etc.



5

JavaScript

En la parte de Frontend se ha utilizado principalmente JavaScript ya que mi proyecto precisa de la capacidad que ofrece para crear webs dinámicas y ha provocado una mejora evidente en la funcionalidad del mismo.

```
function cerrarListas(elemento) {
    // cierra todas las listas que existan menos la introducida en el argumento
    // de momento no esta implementado
    var x = document.getElementsByClassName("items-autocompletar");
    for (var i = 0; i < x.length; i++) {
        if (elemento != x[i]) x[i].parentNode.removeChild(x[i]);
    }
}

$("#buscador").on("submit", function (evento) {
    evento.preventDefault();
});

$("#buscar").on("keyup", function (evento) {
    if ($("#buscar").val().length > 2) autocompletar();
    else cerrarListas(evento);
});
```

6

⁵ Fragmento de la interfaz de phpMyAdmin.

⁶ Algunos ejemplos de uso de JavaScript y manejo de eventos (DOM).

Su uso junto con sus librerías **jQuery** y **SweetAlert2** ha sido lo que me ha permitido tener un funcionamiento más dinámico y estético tanto a la hora de mostrar contenido como de actualizarlo y manejar todos los eventos necesarios.

jQuery

Esta librería de JavaScript simplifica la forma en la que se interactúa con la manipulación del árbol **DOM (Document Object Model)**, el manejo de eventos y el uso de **AJAX**, que gracias a su funcionamiento asíncrono ha facilitado con creces mi necesidad de enviar y recuperar datos del servidor sin necesidad de recargar la página donde se encuentre el usuario.

```
// añadir al carrito
$(".agregarCarrito").on("click", function (evento) {
    // Introducirá el producto en el carrito
    // Hacer llamada AJAX al metodo
    $.post(
        base_url + "Carrito_c/insertarCarta_ajax",
        {
            id_usuario: $("#id_user").data("id_user"),
            id_usu_carta: $(this).parent().data("id_usu"),
            id_carta: $(this).parent().data("id_carta"),
            estado: $(this).parent().data("estado"),
            idioma: $(this).parent().data("idioma"),
        },
        function () {
            filtrarCartas();
        }
    );
});
```

7

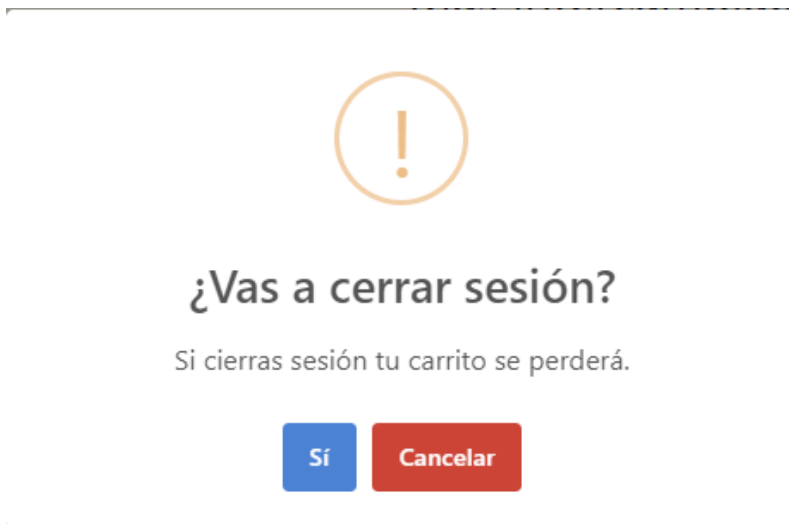
⁷ Fragmento de código utilizando jQuery para enviar datos al servidor vía POST.

Sweet Alerts 2

Esta es otra librería de JavaScript que permite mostrar alertas personalizadas donde puedes decidir el contenido, incluyendo animaciones, diálogos de confirmación y saber las respuestas del usuario incluso recogiendo la información.

```
$(".borrarUsuCarrito").on("click", function (evento) {
  let id_usu = $(this).data("id_usu");
  Swal.fire({
    title: "¿Seguro?",
    text: "Todas las cartas de este usuario se borrarán del carrito.",
    icon: "info",
    showCancelButton: true,
    confirmButtonColor: "#3085d6",
    cancelButtonColor: "#d33",
    confirmButtonText: "Sí",
    cancelButtonText: "Cancelar",
  }).then((result) => {
    if (result.isConfirmed) {
      // Hacer la llamada ajax
      $.post(
        base_url + "Carrito_c/eliminarUsuCarrito",
        { id_usu_carta: id_usu },
        function (dev) {
          location.reload();
        }
      );
    }
  });
});
```

8



9

⁸ Fragmento de código donde se muestra un evento que provoca la aparición de una alerta de esta librería.

⁹ Un ejemplo de una alerta creada con esta librería.

BootStrap 5

Este framework CSS es el responsable de casi toda mi carga de estilos y es el que me ha permitido no solo mantener una estética uniforme a lo largo de todo el proyecto, sino que además posibilita crear una página web responsive de una forma muy simple y eficiente. Además, incluye una librería de iconos amplia y de muy buena calidad.

```
<div class="justify-content-end d-flex col-md-6 col-10">
  <a class="btn btn-warning me-2" href="{%= BASE_URL %}Usuarios_c/login">
    <i class="bi bi-box-arrow-in-right"></i><span class="d-none d-md-inline"> Iniciar sesión</span>
  </a>
  <a class="btn btn-warning me-2" href="{%= BASE_URL %}Usuarios_c/registro">
    <i class="bi bi-person-plus-fill"></i><span class="d-none d-md-inline"> Registrarse</span>
  </a>
</div>
```

10

MVC

El Modelo-vista-controlador ha sido el patrón utilizado para organizar la estructura de este proyecto. A grandes rasgos se trata de construir tres componentes: los modelos que representan y gestionan los accesos a la información, los controladores que responden a los eventos y lanzan peticiones al modelo cuando es necesario y las vistas que muestran la información al usuario y permite la interacción (una interfaz de usuario).

¹⁰ Fragmento de código donde se puede apreciar el uso de la librería Bootstrap.

Análisis de requisitos

Usuarios

El tipo de usuario para el que está destinada esta aplicación es muy específico: personas dentro de la misma afición que buscan intercambiar artículos entre sí. Es por esto que he pensado en un único rol para cualquier persona que se registre ya que van a tener las mismas funcionalidades: todos van a poder visualizar cartas, añadirlas a sus colecciones en su cuenta y enviar ofertas a otros usuarios.

Requisitos funcionales y no funcionales

Para cumplir con todas las partes del proceso de intercambio de cartas la aplicación necesita los siguientes requisitos funcionales:

- Registro de usuarios: los usuarios deben poder crear una cuenta en la aplicación con sus datos personales para tener acceso a la mayoría de las funcionalidades.
- Ver perfiles: los usuarios deben poder visualizar tanto los perfiles de otros usuarios como el suyo propio y ver la información que se encuentre dentro de los perfiles.
- Buscar y ver las cartas: los usuarios deben poder buscar artículos utilizando el nombre de la carta y deben poder ver información asociada a esa carta además de cuantas se encuentran disponibles para intercambiar.
- Ver colecciones: los usuarios deben poder ver tanto la colección propia como las de otros usuarios.
- Gestión de la colección: los usuarios deben poder añadir cartas que posean a su colección personal con el fin tanto de recibir como enviar ofertas. También podrán borrarlas en caso de que ya no las tengan o no deseen intercambiarlas.
- Gestión del carrito: los usuarios deben poder añadir cartas de otros usuarios a su carrito con el fin de realizar un intercambio. En caso de cambiar de idea o de cerrar sesión también deberán poder borrar esas cartas de su carrito.
- Realizar intercambios con otros usuarios: los usuarios deben poder enviar ofertas de intercambio a esos usuarios donde se incluirán tanto las cartas del

usuario destino que se encuentren en el carrito como las cartas que el usuario decida utilizar para el intercambio.

- Gestión de intercambios: los usuarios deben poder visualizar los intercambios enviados y recibidos, ver las características y las cartas asociadas a ese envío y en el caso de ser una oferta recibida deben poder aceptar o rechazar esa oferta.
- Enviar mensajes con otros usuarios: los usuarios deben poder enviar mensajes a otros usuarios con el fin de informar o negociar.

Como requisitos no funcionales considero que lo más determinante va a ser una interfaz limpia, clara e intuitiva para que los usuarios disfruten de una experiencia amigable.

Casos de uso



11

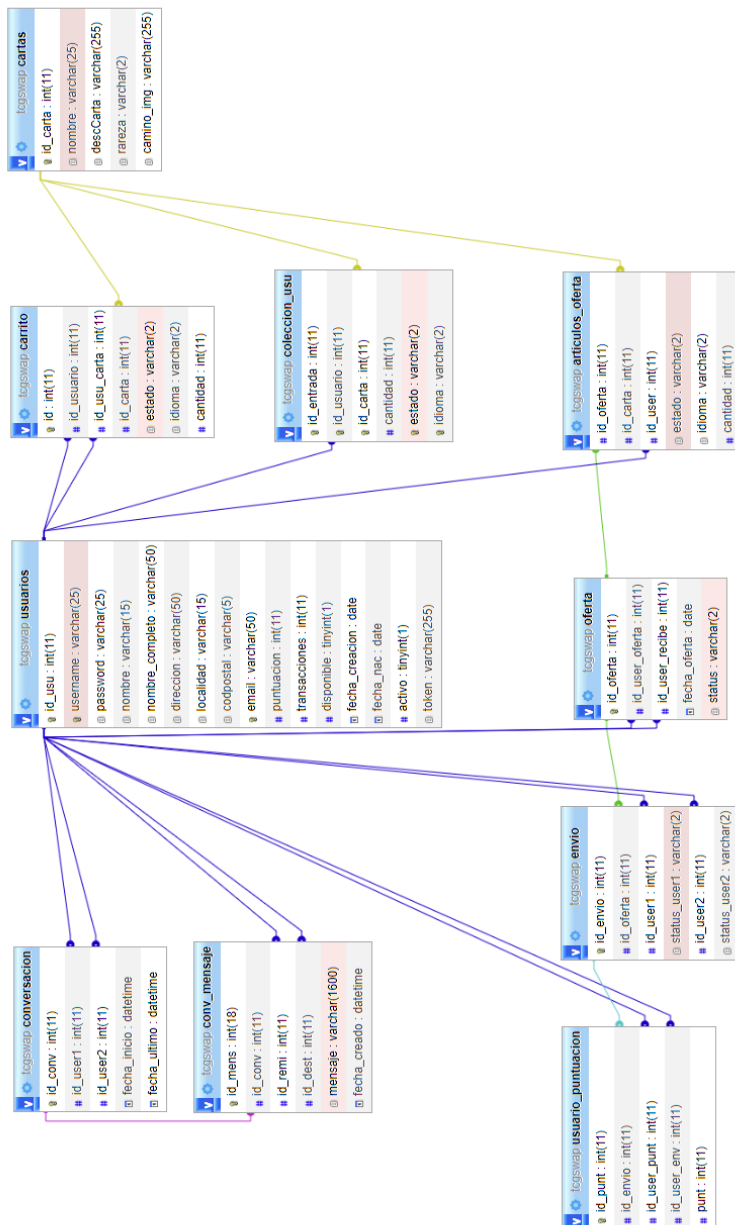
¹¹ Diagrama de casos de uso de la aplicación TCGSwap.

Aquí el desarrollo de un caso de uso:

Flujo de eventos para el caso de uso Buscar carta	
Caso de uso	Buscar carta
Autor	Usuario (registrado o no registrado)
Fecha	12 de mayo de 2023
Descripción Breve	El usuario pone el foco en la barra de búsqueda e introduce una cadena de texto para buscar resultados que coincidan con ella. El sistema busca entre las cartas existentes para encontrar una carta con nombre coincidente y lo muestra debajo.
Precondiciones	Que haya cartas añadidas a la BBDD.
Flujo de eventos normal	<ol style="list-style-type: none"> 1. El usuario pone el foco en la barra de búsqueda para permitir la escritura. 2. El usuario escribe una cadena de caracteres perteneciente a la carta que quiera encontrar. 3. El sistema compara la cadena escrita con las cartas existentes en la BBDD. 4. El sistema muestra una lista de cartas cuyo nombre coincide con la cadena escrita por el usuario.
Flujo alternativo (no hay resultados coincidentes)	<ol style="list-style-type: none"> 1. El usuario pone el foco en la barra de búsqueda para permitir la escritura. 2. El usuario escribe una cadena de caracteres perteneciente a la carta que quiera encontrar. 3. El sistema compara la cadena escrita con las cartas existentes en la BBDD. 4. El sistema muestra un mensaje describiendo que no se han encontrado cartas que coincidan con la cadena escrita.
Postcondiciones	Se muestran todas las cartas que resulten de la búsqueda.

Diseño

Diseño de la base de datos



La imagen anterior muestra el diseño final de la BBDD de mi aplicación, con todas las tablas y relaciones pensadas desde un enfoque de añadir todas las posibles funcionalidades, incluidas aquellas que por razones de tiempo y planificación no se han incluido en la primera iteración del proyecto (usuario_puntuacion y envio serían los ejemplos de esto). Aún así, faltarían tablas y relaciones por crear.

Las entidades principales son usuarios y cartas y a partir de estas se desarrollan el resto.

Diseño de la interfaz

A partir de lo mencionado en los requisitos no funcionales pensé en una interfaz limpia, clara e intuitiva que permitiera que todas las funcionalidades fueran accesibles desde cualquier punto. Acabé llegando a la conclusión de que lo mejor podría ser un encabezado que incluyera todo lo necesario para el usuario no registrado o registrado. Después pensé en la estética de la página y la forma en la que quería que se mostrasen los datos de cada vista y lo desarrollé a partir del uso de mockups. Estos son algunos ejemplos:



13

¹³ Mockups del encabezado/barra de navegación para usuarios no registrados y registrados.

Mi coleccion

Busqueda avanzada

Nombre

Q search

Idioma

Idioma1

Idioma2

Idioma3

Estado

Cal nueva

Excelente

Estado3

Observaciones

Q search

Filtrar

Nombre	Idioma	Estado	Extra	Observaciones	Cantidad	Editor
Nombre1	Idioma1	Estado1			1	  
Nombre2	Idioma1	Estado1	Foil		2	  
Nombre3	Idioma1	Estado1	Foil		1	  
Nombre4	Idioma1	Estado1			1	  
Nombre5	Idioma1	Estado1			2	  
Nombre6	Idioma1	Estado1			1	  
Nombre7	Idioma1	Estado1			1	  
Nombre8	Idioma1	Estado1			1	  
Nombre9	Idioma1	Estado1			1	  
Nombre10	Idioma1	Estado1			1	  
Nombre11	Idioma1	Estado1	Foil		3	  
Nombre12	Idioma1	Estado1			2	  
Nombre13	Idioma1	Estado1			1	  
Nombre14	Idioma1	Estado1			3	  
Nombre15	Idioma1	Estado1			1	  

14

Salvo por algunas modificaciones los mockups resultan bastante orientativos con respecto al resultado final.

¹⁴ Mockup sobre la vista de la colección de un usuario.

Implementación

Codificación

Teniendo en consideración la carga del proyecto y el uso de la arquitectura MVC, he decidido incluir solo capturas significativas de todas las partes para que haya una visión general y quede claro el uso de todas las tecnologías mencionadas anteriormente.

```

Usuarios_m.php x
app > modelos > Usuarios_m.php > Usuarios_m
1  <?php
2  // modelo para la gestion de usuarios
3  class Usuarios_m extends Model
4  {
5      public function __construct()
6      {
7          parent::__construct();
8      }
9      // leerUsu realiza la busqueda del usuario a partir del username
10     public function leerUsu($usu){
11         $cadSQL = "SELECT * FROM usuarios WHERE (username=:usu)";
12         $this->consultar($cadSQL);
13         $this->enlazar(":usu", $usu);
14         $fila = $this->fila();
15         return $fila;
16     }
17     // autenticar es usado a la hora de hacer login para confirmar que tanto el user como la contraseña son correctas
18     public function autenticar($usu, $pass)
19     {
20         $cadSQL = "SELECT * FROM usuarios WHERE (username=:usu or email=:usu) and activo=1";
21         $this->consultar($cadSQL);
22         $this->enlazar(":usu", $usu);
23         $fila = $this->fila();
24         if ($fila) {
25             // Comprobar el password
26             if ($fila['password']!=$pass) {
27                 return null;
28             }
29         }
30         return $fila;
31     }
32     // insertar realiza el insert con todos los datos del usuario que se hayan rellenado en el formulario de registro
33     public function insertar($datos)
34     {
35         // Recibimos los datos del formulario en un array
36         // Obtenemos cadena con las columnas desde las claves del array asociativo
37         $columnas = implode(",", array_keys($datos));
38         // Campos de columnas
39         $campos = array_map(
40             function ($col) {
41                 return ":" . $col;
42             },
43             array_keys($datos)
44         );
45         $parametros = implode(",", $campos); // Parametros para enlazar
46         $cadSQL = "INSERT INTO usuarios ($columnas) VALUES ($parametros)";
47         $this->consultar($cadSQL); // Preparar sentencia
48         for ($ind = 0; $ind < count($campos); $ind++) { // Enlace de parametros
49             $this->enlazar($campos[$ind], $datos[array_keys($datos)[$ind]]);
50         }
51         return $this->ejecutar();

```

15

¹⁵ Fragmento de código del modelo “Usuarios_m.php”. Se pueden ver las sentencias SQL para llamar a la BBDD

```

Usuarios_c.php X
app > controladores > Usuarios_c.php > Usuarios_c > miCuenta
12 class Usuarios_c extends Controller
13 {
14     private $usuarios_m; // Propiedad para instanciar el modelo
15
16     public function __construct()
17     {
18         $this->usuarios_m = $this->load_model("Usuarios_m");
19     }
20     public function index()
21     {
22     }
23     // login presenta la vista del Login
24     public function login()
25     {
26         $titulo['nombre'] = "Inicia sesión";
27         $contenido = "login_v";
28         $this->load_view("plantilla/cabecerasinheader", $titulo);
29         $this->load_view($contenido);
30     }
31     // logout es el metodo que permite a un usuario logeado desconectarse
32     public function logout()
33     {
34         $carrito_m = $this->load_model("Carrito_m");
35         // Destruimos el carrito
36         $carrito_m->vaciarCarritoUsu($_SESSION['sesion']['id_usu']);
37         // Destruimos la sesion
38         unset($_SESSION['sesion']);
39     }
40     // miCuenta carga la vista que muestra la informacion personal del usuario que se encuentra logeado
41     public function miCuenta()
42     {
43         // Recibo el usuario de session y lo envio al modelo para sacar sus datos
44         $datos['usuario'] = $this->usuarios_m->leerUsu($_SESSION['sesion']['username']);
45         // Visualizar la pagina de perfil
46         $titulo['nombre'] = "Mi cuenta";
47         $contenido = "cuenta_v";
48         $this->load_view("plantilla/cabecera", $titulo);
49         $this->load_view($contenido, $datos);
50         $this->load_view("plantilla/pie");
51     }
52     //perfil_publico carga la vista que muestra el perfil publico de un usuario que se encuentre registrado en la web
53     public function perfil_publico($datos)
54     {
55         if (!$datos[0]) {
56             $this->load_view("errorPagina");
57         } else {
58             $usu = $datos[0];
59             // Recibo el usuario y lo envio al modelo para sacar sus datos
60             $datos['usuario'] = $this->usuarios_m->leerUsu($usu);
61             // Visualizar la pagina de perfil
62             if ($datos['usuario']) {

```

16

¹⁶ Fragmento de código del controlador “Usuarios_c.php”. Se puede apreciar la instancia del modelo para poder interactuar con él.

```

carrito_v.php
app > vistas > carrito_v.php > main.container-fluid > div.pt-3.pt-md-5 > div.row.justify-content-center > div.col-md-8.col-sm-8.container-gris.p-4.mx-2
19
20 <div class="col-md-8 col-sm-8 container-gris p-4 mx-2">
21 <!-- Generación del carrito usando php -->
22 <?php if (empty($items)) : ?>
23 <div class="text-center fs-3 fw-bold">¡Tu carrito está vacío! Añade algunas cartas de otros usuarios.</div>
24 <?php endif; ?>
25 <?php $userPrevio = "";
26 ?>
27 <?php foreach ($items as $clave => $item) : ?>
28 <?php if (($item['username'] == $userPrevio && $clave == array_key_last($items)) : ?>
29 <tr class="align-middle" data-id="<?php $item['id'] ?>">
30 <td><a href="<?php BASE_URL ?>Cartas_c/datosCarta/<?php $item['nombre_carta'] ?>" class="fw-bold text-decoration-none"><?php str_replace("-", " ", $item['nombre_carta']) ?></a></td>
31 <td><?php $item['estado'] ?>, <?php $item['idioma'] ?></td>
32 <td class="d-flex justify-content-between align-items-center"><span><?php $item['cantidad'] ?></span>
33 <div class="id_usu"><button class="btn btn-sm btn-danger m-1 borrarFilaCarrito"><i class="bi bi-trash"></i></button></div>
34 </td>
35 </tr>
36 </tbody>
37 </table>
38
39 <div class="d-flex justify-content-end">
40 <a href="<?php BASE_URL ?>Ofertas_c/enviarOferta/<?php $item['id_usu_carta'] ?>" class="btn btn-sm btn-warning hacerOferta">Hacer oferta</a>
41 </div>
42 </div>
43 <?php endif; ?>
44 <?php if (($item['username'] != $userPrevio && $clave != array_key_first($items)) : ?>
45 </tbody>
46 </table>
47 </div>
48 <div class="d-flex justify-content-end">
49 <a href="<?php BASE_URL ?>Ofertas_c/enviarOferta/<?php $item['id_usu_carta'] ?>" class="btn btn-sm btn-warning hacerOferta">Hacer oferta</a>
50 </div>
51 </div>
52 <?php endif; ?>
53 <?php if ($userPrevio == $item['username'] && $clave != array_key_last($items)) : ?>
54 <tr class="align-middle" data-id="<?php $item['id'] ?>">
55 <td><a href="<?php BASE_URL ?>Cartas_c/datosCarta/<?php $item['nombre_carta'] ?>" class="fw-bold text-decoration-none"><?php str_replace("-", " ", $item['nombre_carta']) ?></a></td>
56 <td><?php $item['estado'] ?>, <?php $item['idioma'] ?></td>
57 <td class="d-flex justify-content-between align-items-center"><span><?php $item['cantidad'] ?></span>
58 <div class="id_usu"><button class="btn btn-sm btn-danger m-1 borrarFilaCarrito"><i class="bi bi-trash"></i></button></div>
59 </td>
60 </tr>
61 </div>
62 <?php if ($clave == array_key_first($items) || $item['username'] != $userPrevio) : ?>
63 <div class="row border border-secondary rounded-1 p-2 mb-3">
64 <div class="d-flex justify-content-between my-2">
65 <a href="<?php BASE_URL ?>Usuarios_c/perfil_publico/<?php $item['username'] ?>" class="fw-bold fs-5 text-decoration-none"><?php $item['username'] ?></a>
66 <button class="btn btn-sm btn-danger borrarUsuCarrito" data-id-usu="<?php $item['id_usu_carta'] ?>">Eliminar todo</button>
67 </div>
68 </div>
69 <div class="table-responsive tabla_carrito">
70 <table class="table table-sm table-bordered table-striped table-warning">
71 <thead class="table bg-light">
72 <th scope="col" style="width: 40%;>Nombre</th>
73 <th scope="col" style="width: 20%;>Información</th>
74 <th scope="col" style="width: 20%;>Cantidad</th>
75 </thead>
76 <tbody>
77 <tr class="align-middle" data-id="<?php $item['id'] ?>">
78 <td><a href="<?php BASE_URL ?>Cartas_c/datosCarta/<?php $item['nombre_carta'] ?>" class="fw-bold text-decoration-none"><?php str_replace("-", " ", $item['nombre_carta']) ?></a></td>
79 <td><?php $item['estado'] ?>, <?php $item['idioma'] ?></td>
80 <td class="d-flex justify-content-between align-items-center"><span><?php $item['cantidad'] ?></span>
81 <div><button class="btn btn-sm btn-danger m-1 borrarFilaCarrito"><i class="bi bi-trash"></i></button></div>
82 </td>
83 </tr>
84 <?php if ($clave == array_key_last($items)) : ?>
85 </tbody>
86 </table>
87 </div>
88 <div class="d-flex justify-content-end">
89 <a href="<?php BASE_URL ?>Ofertas_c/enviarOferta/<?php $item['id_usu_carta'] ?>" class="btn btn-sm btn-warning hacerOferta">Hacer oferta</a>
90 </div>
91 </div>
92 <?php
93
94

```

17

¹⁷ Fragmento de la vista “carrito_v.php”. Aquí se puede visualizar la carga de datos vía PHP a través del controlador y el uso de la librería de Bootstrap.

```

JS cabecera.js X
app > vistas > js > JS cabecera.js > ...
1 //region BOTON DE CIERRE DE SESION
2 // Si el usuario intenta cerrar sesion, vamos a avisarle de que el carrito va a ser eliminado.
3 $("#cerrarSesion").on("click", function (evento) {
4     Swal.fire({
5         title: "¿Vas a cerrar sesión?",
6         text: "Si cierras sesión tu carrito se perderá.",
7         icon: "warning",
8         showCancelButton: true,
9         confirmButtonColor: "#3085d6",
10        cancelButtonColor: "#d33",
11        confirmButtonText: "Sí",
12        cancelButtonText: "Cancelar",
13    }).then((result) => {
14        if (result.isConfirmed) {
15            // Hacer la llamada ajax
16            $.post(base_url + "Usuarios_c/logout", "", function (dev) {
17                location.href=base_url+"Portal_c";
18            });
19        }
20    });
21 });
22 //endregion
23

```

```

JS cabecera.js X
app > vistas > js > JS cabecera.js > ...
25 // variable con los parametros del buscador
26 let opcion = { buscar: "" };
27
28 // metodo que a traves de ajax llama al controlador
29 function buscarCartas(opcion) {
30     $.post(base_url + "Cartas_c/buscarCartas", opcion, function (datos) {
31         mostrarResultados(JSON.parse(datos));
32     });
33 }
34 // metodo que muestra las cartas que resulten de la busqueda
35 function mostrarResultados(cartas) {
36     let lista = document.createElement("DIV");
37     lista.setAttribute("id", "lista-autocompletar");
38     lista.setAttribute("class", "items-autocompletar");
39     $("#resultados").html(lista);
40     let item;
41     if (cartas.length == 0) {
42         item = document.createElement("DIV");
43         item.setAttribute(
44             "class",
45             "resultados fw-bold rounded-1 text-warning d-flex d-md-flex"
46         );
47         item.innerHTML = "Lo sentimos, no existe la carta introducida.";
48         lista.appendChild(item);
49     } else {
50         for (carta of cartas) {
51             item = document.createElement("A");
52             item.setAttribute(
53                 "class",
54                 "resultados fw-bold rounded-1 text-decoration-none d-flex d-md-flex"
55             );
56             item.setAttribute(
57                 "href",
58                 `${base_url}Cartas_c/datosCarta/${carta.nombre}`
59             );
60             item.innerHTML = `${carta.nombre.replaceAll("-", " ")}`;
61             lista.appendChild(item);
62         }
63     }
64 }
65

```

18

¹⁸ Fragmentos del archivo “cabecera.js”. En ellos se puede apreciar tanto el uso de la librería de Sweet Alerts 2 como el uso de la librería de jQuery.

Documentación

Manual de usuario

1. Landing page

Cuando un usuario accede a la página web lo primero que se le presenta es una página de inicio con una breve descripción y tres opciones: registrarse, iniciar sesión o explorar la web para ver su contenido.



Un usuario no registrado podrá explorar algunas partes de la web, pero para poder usarla al completo necesitará estar registrado.

2. Registro



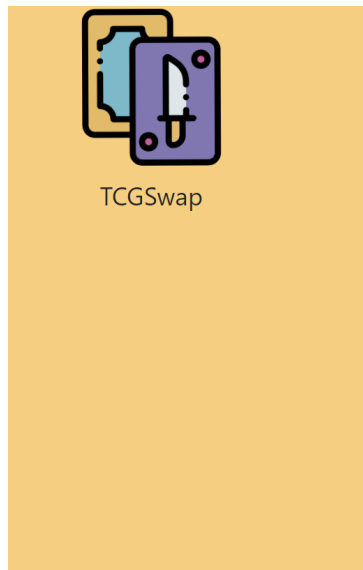
The screenshot shows the TCGSwap registration form. At the top, the TCGSwap logo is displayed. The form is titled "Formulario de registro" with the subtitle "Crea tu cuenta de usuario en TCGSwap". A note states "Los campos con * son obligatorios". The form is divided into three main sections: "Datos personales*", "Dirección*", and "Datos de la cuenta*". The "Datos personales*" section includes fields for "Nombre*", "Apellido/s*", and "Fecha de nacimiento*" (with a date picker). The "Dirección*" section includes fields for "Calle*", "Número/Piso*", "Código postal*", and "Ciudad*" (with a "Localidad" sub-field). The "Datos de la cuenta*" section includes fields for "Email*", "Nombre de usuario*", "Contraseña*", and "Confirmar contraseña*". A "Registrarse" button is at the bottom.

A esta vista se puede acceder de varias formas: a través del botón en la landing page, a través de un enlace en el login o a través de un botón de la cabecera en caso de que no haya un usuario registrado. En el proceso de registro se valida tanto el mail como el usuario y la contraseña, esta última asegurando que esté bien escrita.



This close-up shows the "Datos de la cuenta*" section. The "Email*" field contains "dario.estevez@educarex.es" with a red border and the message "Este email ya ha sido usado." below it. The "Nombre de usuario*" field contains "dariusmtg" with a red border and the message "Ese nombre de usuario ya existe." below it. The "Contraseña*" field contains masked characters "*****" with a red border. The "Confirmar contraseña*" field also contains masked characters "*****" with a red border and a red circular icon with an exclamation mark. A message at the bottom states "Las contraseñas no son iguales."

3. Inicio de sesión



Inicio de sesión

Usuario/Email

Contraseña

Recuérdame ☐

Login

[¿Aún no tienes cuenta? Regístrate aquí](#)

El inicio de sesión consta de los campos donde introducir el usuario o email y la contraseña asociada a estos. Una vez validadas las credenciales puede haber dos escenarios: que los datos sean correctos y nos lleve al portal asociado al usuario que se ha introducido o que los datos sean incorrectos y nos devuelva a la pantalla de login con un mensaje de error.

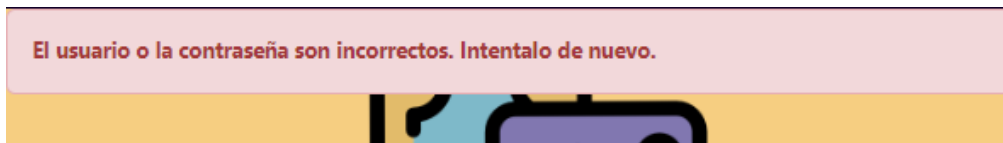
En caso de que el usuario no introduzca uno de los campos:

Inicio de sesión

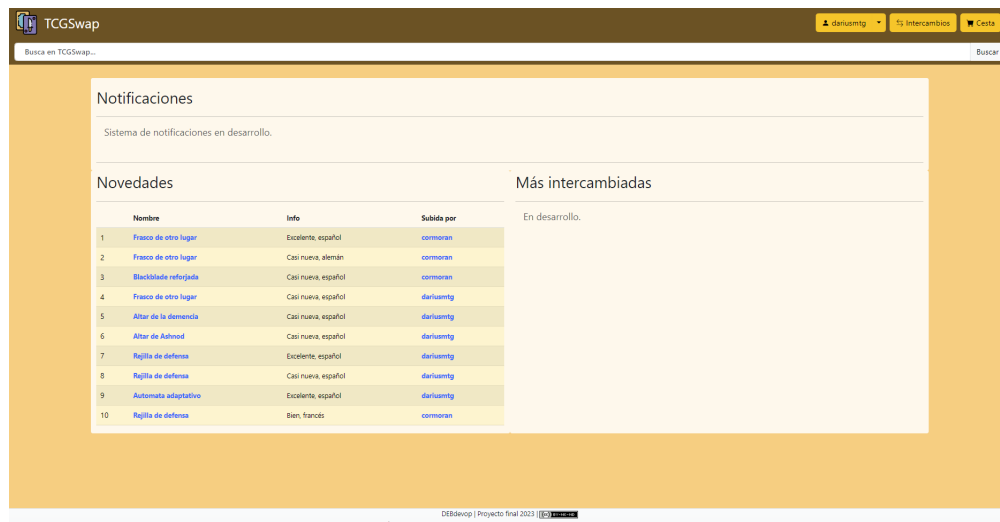
Usuario/Email

Contraseña

En caso de que las credenciales sean incorrectas:



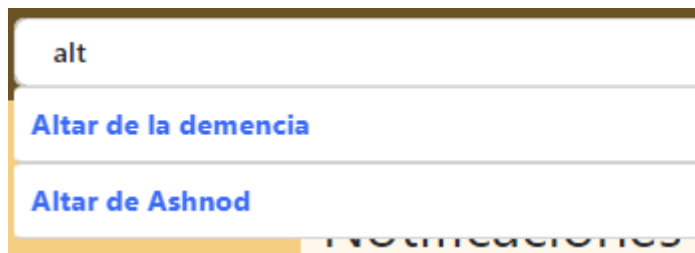
4. Portal



Una vez la sesión está iniciada se nos presenta el portal de inicio, donde nos encontramos por primera vez con la cabecera (el elemento común que comparten la mayoría de las vistas y donde se encuentra todo lo relacionado con el usuario y la búsqueda de cartas) y el portal principal del usuario. Como su contenido aún se encuentra en desarrollo, el único contenido disponible es las novedades (últimas cartas insertadas por cualquier usuario), ya que las otras dos partes aún no se encuentran implementadas.

5. Búsqueda de cartas

Para realizar la búsqueda de una carta que se encuentre dentro de la base de datos, el usuario (tanto registrado como no registrado) puede escribir dentro de la barra de búsqueda.



El sistema según vaya encontrando resultados los mostrará en una lista desde la cual el usuario puede clicar sobre cualquier carta para ver sus detalles.

6. Ver una carta, añadirla al carrito o añadirla a tu colección

Una vez hemos seleccionado una carta, la vista nos mostrará los detalles únicos de esta, la opción de añadirla a nuestra colección con las características que elijamos y un listado de personas que tienen esa carta en su colección actualmente, el cual puede ser filtrado según las características deseadas. Añadirla a nuestra colección e interactuar con las cartas de otros usuarios solo es posible si nuestra sesión esta iniciada.

Sin una sesión iniciada:

Altar de la demencia

Información de la carta

Rareza: Mítica

Descripción: Sacrificar una criatura: El jugador objetivo muele una cantidad de cartas igual a la fuerza de la criatura sacrificada.

Unidades disponibles: 3

Añadir a mi colección

Para poder añadir esta carta a tu colección debes iniciar sesión.

Opciones de filtrado


Estado Idioma

[Añadir filtros](#) [Limpiar filtros](#)

Nombre	Información	Cantidad
dariusmtg	Casi nueva, español	3

Con una sesión iniciada:

Altar de Ashnod



Información de la carta

Rareza: Rara

Descripción: Sacrificar una criatura: Agrega dos manás incoloros.

Unidades disponibles 3

Añadir a mi colección

Cantidad

Estado

Idioma



Añadir a tu colección

Opciones de filtrado

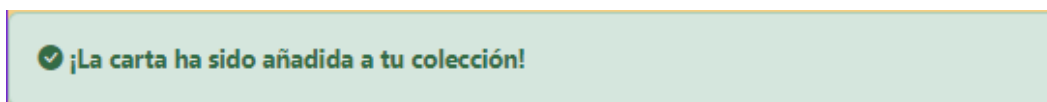
Estado

Idioma

Añadir filtros **Limpiar filtros**

Nombre	Información	Cantidad
cormoran	Casi nueva, español	2 
dariusmtg	Casi nueva, español	1 

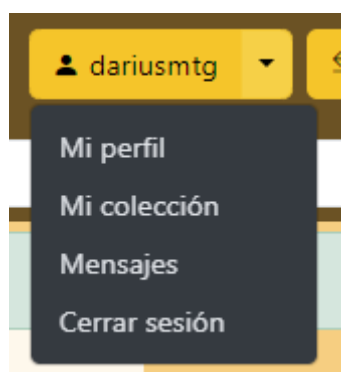
En este último ejemplo podemos ver que el botón de añadir el carrito se encuentra activo y que no podemos añadir cartas propias (botón desactivado y con un aviso si pasamos el cursor sobre él). Para añadir esa carta a tu carrito solo tienes que pulsar sobre el botón azul y automáticamente se añadirá a este. También aparece la opción de añadir esa carta a tu colección. En el caso de añadir una nueva carta se actualizará el listado inferior y se enviará un mensaje de aviso:



También podemos ver la carta añadida en la colección propia.

7. Perfil personal

Para esta acción deberemos hacer clic sobre el botón donde se encuentra el nombre de usuario con el que se ha accedido a la aplicación y se desplegarán una serie de acciones:



Una vez nos encontremos ahí, seleccionaremos la opción de **“Mi perfil”**. Nos llevará a la página del perfil personal del usuario.

Mi perfil

dariusmtg

Fecha de ingreso:	2023-02-23
Nombre completo:	Dario Fribidis
Fecha de nacimiento:	2005-02-01
Estado de la cuenta:	Activo

Correo electrónico:	dario.estevez@educarex.es
Contraseña:	*****

Dirección:	Gil Cordero 12
Localidad:	Caceres
Código postal:	10001

[Ver mi perfil público](#)

En esta página podemos revisar todo lo referente al usuario que se haya introducido a la hora de registrarlo y un enlace al final para ver su perfil público, que es el que cualquier otro usuario puede visualizar.

8. Mi colección

Para esta opción debemos situarnos de nuevo sobre el desplegable de opciones del usuario y seleccionar la opción de **“Mi colección”**. La página que se carga es la misma que cargarías al visualizar la colección de otro usuario, con la salvedad de que se pueden borrar cartas que pertenezcan a tu colección.

dariusmtg

Colección

Opciones de filtrado

Nombre

Estado

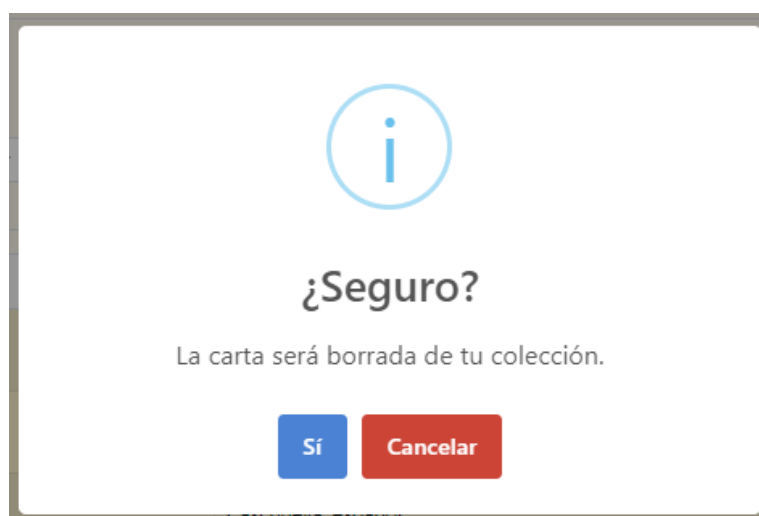
Idioma

Añadir filtros

Limpiar filtros

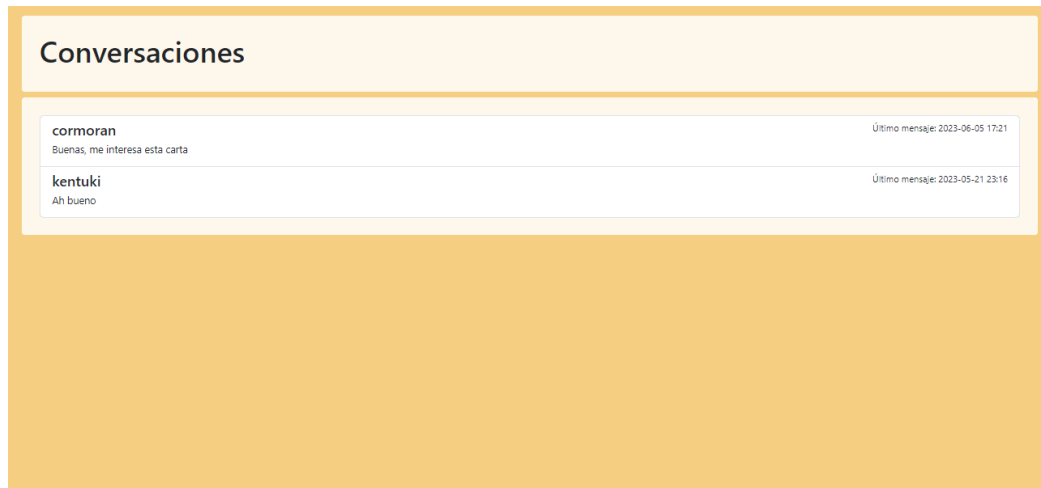
Nombre	Información	Cantidad
Altar de la demencia	Casi nueva, español	3
Altar de Ashnod	Casi nueva, español	2
Rejilla de defensa	Casi nueva, español	2
Rejilla de defensa	Excelente, español	1
Frasco de otro lugar	Casi nueva, español	1

En el momento en el que pulsas el botón rojo, se muestra una ventana modal advirtiéndote de que la carta va a ser borrada.



9. Mis mensajes

De nuevo, para usar esta opción nos situamos en el menú desplegable del usuario y esta vez seleccionamos la opción de **“Mensajes”**. Se carga una página que muestra todas las conversaciones que hayas tenido con otros usuarios.

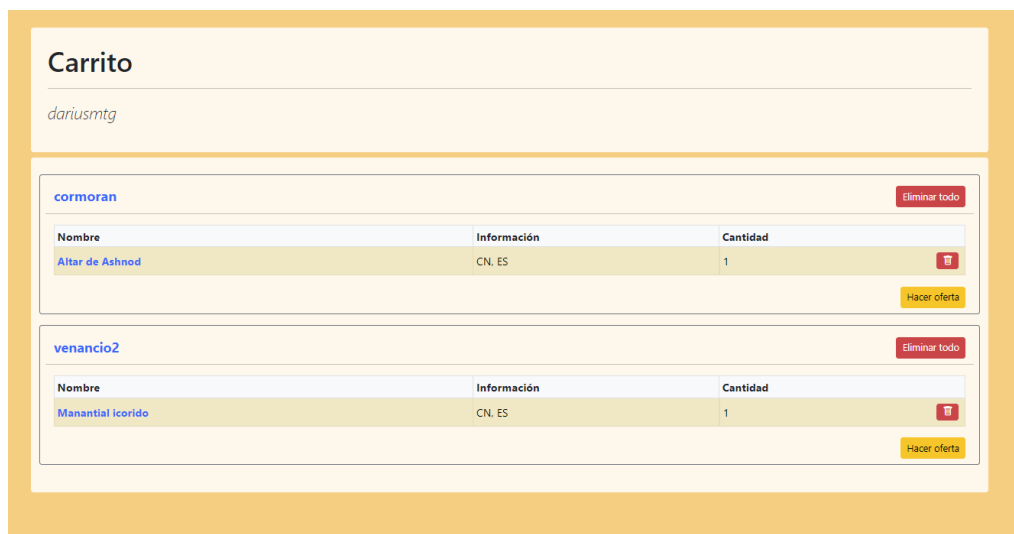
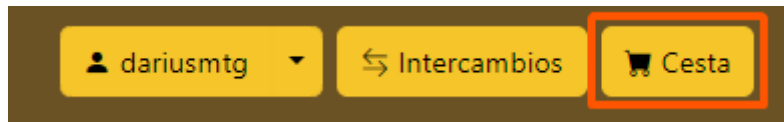


Si se pulsa sobre una de las conversaciones, se cargarán todos los mensajes que se hayan mandado en ella y una pequeña barra para escribir y mandar tus mensajes al usuario con el que estás teniendo la conversación.



10. Realizar un intercambio

Una vez se han añadido cartas al carrito puedes pasar al proceso de intercambio. Para esto, deberás situarte en la parte derecha de la cabecera y hacer clic en el botón llamado “Cesta”.



La página muestra de manera ordenada los usuarios a los que pertenecen las cartas que hemos añadido al carrito con una lista de las cartas que hemos seleccionado de sus colecciones, pudiendo borrar tanto una línea de una carta que ya no te interese en la oferta como borrar todas aquellas relacionadas con el usuario. Para iniciar un intercambio solo hay que pulsar en el botón “Hacer oferta”.

(ADVERTENCIA: si no has añadido cartas a tu colección no podrás realizar intercambios y el botón de “Hacer oferta” estará desactivado).

La página que aparecerá a continuación será la de realizar la oferta, que tiene este aspecto:

Hacer oferta

venancio2

Tus cartas

Añade cartas para continuar.

Sus cartas

Nombre	Información	Cantidad
Manantial icorido	CN, ES	1

Confirmar intercambio

Opciones de filtrado

Nombre Estado Idioma

Nombre	Información	Cantidad
Altar de la demencia	Casi nueva, español	3
Altar de Ashnod	Casi nueva, español	2
Rejilla de defensa	Casi nueva, español	2
Rejilla de defensa	Excelente, español	1
Frasco de otro lugar	Casi nueva, español	1

Hay que diferenciar tres partes: “Tus cartas” contiene los artículos que tú ofreces en la oferta, “Sus cartas” contiene los artículos que has escogido del usuario previamente y la parte de abajo donde aparece el listado de cartas que tienes para ofrecer. En cada una de estas filas, al pulsar sobre el botón verde que se ve a la derecha las añadirás a la sección de “Tus cartas” para ofrecerlas en el intercambio (**no puedes continuar si no añades tu parte de la oferta**). Además, para hacerlo más fácil se incluye un pequeño filtro de búsqueda.

Tus cartas

Nombre	Información	Cantidad
Altar de la demencia	CN, ES	1
Altar de Ashnod	CN, ES	1

Sus cartas

Nombre	Información	Cantidad
Manantial icorido	CN, ES	1

Confirmar intercambio

Para finalizar el proceso enviar una oferta solo faltaría pulsar sobre el botón de **“Confirmar intercambio”**, que muestra un mensaje de confirmación y devuelve a la página del carrito.

11. Gestionar intercambios

Para poder visualizar y gestionar tus intercambios debes volver a situarte en la parte superior derecha de tu cabecera y hacer clic sobre el botón “Intercambios”.

Intercambios			
dariusmtg			
Todos			
Ofertador	Ofertado	Fecha	Estado
Tú	venancio2	2023-06-06	Enviado
Tú	venancio2	2023-03-11	Enviado
Tú	kentuki	2023-03-11	Enviado
Tú	kentuki	2023-03-09	Rechazado

La página contiene un listado de los pedidos realizados, donde se puede observar quien lo envía, quien lo recibe, la fecha de inicio del trámite y el estado de la oferta, además de un botón que lleva a los detalles de la misma. Además, se incluye un filtro que separa las ofertas entre las enviadas, las recibidas y todas.

Si hacemos clic sobre el botón de detalles se muestra lo siguiente:

Intercambio #1

Enviada por:
Fecha de la oferta:
Estado:

dariusmtg
2023-03-09
Rechazada

Tus cartas

Nombre	Información	Cantidad
Automata adaptativo	EX, ES	1
Rejilla de defensa	CN, ES	1

Cartas de kentuki

Nombre	Información	Cantidad
Deposito de flujo etereo	CN, ES	1
Lanzaesquivas trasgo	US, ES	1

La oferta ha sido rechazada.

Se muestra de nuevo los datos de la oferta y además se incluyen cuales han sido las cartas que se han incluido en esta oferta. En este ejemplo podemos ver

que la oferta ha sido rechazada. El siguiente ejemplo muestra una oferta que aún no ha sido terminada.

Intercambio #5

Enviada por: [dariusmtg](#)

Fecha de la oferta: 2023-06-06

Estado: Pendiente

Tus cartas


Nombre	Información	Cantidad
Manantial icorido	CN, ES	1

Cartas de dariusmtg

Nombre	Información	Cantidad
Altar de la demencia	CN, ES	1
Altar de Ashnod	CN, ES	1

AceptarRechazar

Como se puede ver, al final aparecen dos botones que solo son accesibles para el usuario que ha recibido la oferta y que sirven para la decisión de si aceptas o rechazas la oferta. Al pulsar cualquiera de las dos opciones aparecerá un modal para confirmar la decisión tomada. En el caso de aceptar el estado de la oferta cambia a aceptada. En el caso de rechazarla su estado cambiará a rechazada y las cartas que se incluyeron en la oferta volverán a las colecciones de sus respectivos usuarios.



¿Aceptas la oferta?

Si aceptas dará comienzo el proceso de intercambio.

ConfirmarCancelar



¿Rechazas la oferta?

Todas las cartas de esta oferta volverán a sus respectivas colecciones.

ConfirmarCancelar

Cualquiera de estas opciones da por finalizado el proceso de oferta.

12. Otros usuarios: perfil, colecciones y enviar mensajes

Para poder acceder al perfil de otro usuario solo necesitas pulsar sobre uno cuando lo veas en cualquiera de las partes del proceso: ya sea en el listado de una carta como en el carrito o en una oferta ya enviada.

venancio2

Usuario desde:	2023-03-05
Nombre:	Venancio de Pueblo
Dirección:	Cumingo 2
Código postal:	53245
Localidad:	El Bierzo
Puntuación:	Sin puntuación
Intercambios:	Sin transacciones

ContactarVer colección

Ultimas puntuaciones

En desarrollo.

Al acceder al perfil del usuario nos encontramos con sus datos personales: su fecha de registro, nombre, apellidos, dirección, código postal y localidad. Además, hay dos botones que nos permiten tanto contactar por mensajería con el usuario como ver su colección completa.

Mensajes con venancio2

No hay mensajes.

Enviar

venancio2

Colección

Opciones de filtrado

Nombre

Estado

Idioma

Añadir filtros

Limpiar filtros

Nombre	Información	Cantidad	
Venado bruido	Casi nueva, español	3	
Sol enjaulado	Excelente, francés	2	
Mina aullante	Casi nueva, español	1	
Mina aullante	Excelente, ruso	4	
Volumen de Jalum	Excelente, alemán	4	
Cometa del viajante	Excelente, inglés	2	

Manual del programador

1. Instalación

Para poder desplegar esta aplicación necesitaremos instalar XAMPP, un paquete de software que incluye tres herramientas necesarias: el servidor web Apache, gestión de bases de datos SQL y el intérprete de lenguaje PHP. En la [página oficial](#) podemos encontrar el enlace de descarga.



Una vez instalado, debemos dirigirnos a la ruta donde haya sido instalado (C:\xampp por defecto) y situarnos en la carpeta **htdocs**. Una vez ahí podremos extraer la carpeta **tcgswap** en ella.

Lo siguiente sería importar la base de datos. Para ello solo necesitas acceder a phpMyAdmin y, seleccionando la pestaña “Importar” situada en la barra superior, subir el fichero .sql que se encuentra en la carpeta comprimida del proyecto.

Cuando esto esté realizado la aplicación estará lista para ser utilizada. Dentro de la tabla “usuarios” podrás encontrar varios usuarios con distintos datos. La contraseña es 12345 para todos.


2. Funcionamiento


En lo que se refiere al código, una vez extraída la carpeta podrás acceder a todas las partes que utiliza. Dentro de ese código encontrarás la información necesaria para comprender el funcionamiento del mismo en formato de comentarios de implementación.






















Dentro de la base de datos vas a encontrar además de la estructura mencionada en el diseño una serie de disparadores:

Disparadores

☐ Seleccionar todo

 Exportar

 Eliminar

	Nombre	Tabla	Tiempo	Evento			
<input type="checkbox"/>	delcarr_AD	carrito	AFTER	DELETE	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	insartof_AI	articulos_oferta	AFTER	INSERT	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	inscarr_AI	carrito	AFTER	INSERT	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	insmens_AI	conv_mensaje	AFTER	INSERT	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	insmens_BI	conv_mensaje	BEFORE	INSERT	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	upcarr_AU	carrito	AFTER	UPDATE	 Editar	 Exportar	 Eliminar
<input type="checkbox"/>	upofer_AU	oferta	AFTER	UPDATE	 Editar	 Exportar	 Eliminar

Nombre	Descripción
delcarr_AD	Valida si las cartas salen del carrito por eliminación o por realizar un intercambio. En caso de salir por eliminación comprueba si ya existe registro de esa carta con las características específicas en la colección del usuario al que pertenece. Si no existe la inserta y si existe actualiza la cantidad que tenga.
insartof_AI	Activa la variable utilizada en el disparador anterior para validar que las cartas salen por la realización de un intercambio y borra las cartas del carrito que tengan las mismas características que la insertada.
inscarr_AI	Actualiza las cartas en colecciones con las características de la insertada y borra las filas cuya columna cantidad se encuentre a cero.

insmens_AI	Valida si la conversación donde se introduce el mensaje existe o no. En el caso de existir se actualiza la fecha de la conversación con la fecha de creación del mensaje. En caso contrario se crea la conversación.
insmens_BI	Valida si la conversación a la que pertenece el mensaje existe o no. En el caso de existir asocia el mensaje a la conversación usando su ID. En caso contrario se crea la conversación y se asocia el ID al mensaje.
upcarr_AU	Al actualizar el carrito valida si la carta con esas características existe dentro de una colección y de ser así actualiza su cantidad. En caso contrario inserta la carta de nuevo.
upofer_AU	Valida si la oferta ha sido rechazada, en cuyo caso devuelve todas las cartas que pertenecían a los dos usuarios involucrados a sus respectivas colecciones, comprobando si las claves están duplicadas para poder realizar inserción o actualización según proceda.

Conclusiones

La idea de este TFG era desarrollar una aplicación web de intercambio de cartas entre usuarios que permitiera realizar todas las operaciones relacionadas con la gestión de sus colecciones y la realización de dichos intercambios y creo que eso se ha realizado de manera satisfactoria.

Me he encontrado con algunas dificultades en el desarrollo, como podría ser el desarrollo de los triggers dentro de la base de datos o desplegar la vista del carrito validando cada una de las filas que contiene, pero al final he encontrado la manera de resolverlas.

A nivel de aprendizaje me ha servido para terminar de afianzar los conocimientos adquiridos en todos los lenguajes utilizados, aprender a gestionar bases de datos de una manera más eficiente y limpia y como toma de contacto para comprender mejor el funcionamiento de algunas librerías y sus características.

Además cabe destacar que no se trata de una versión final. Hay funcionalidades y validaciones que no han sido implementadas por razones de tiempo, algunas de ellas se encuentran dentro del proyecto pero no se llegan a usar como podría ser la librería de phpMailer o las tablas de puntuación y envío dentro de la base de datos. Si el desarrollo de la aplicación continúa serán implementadas.

En conclusión, me siento conforme con el resultado ya que creo que cumple las expectativas. He conseguido finalizar los objetivos propuestos y la aplicación cumple con las funcionalidades esperadas, además de tener una interfaz que es agradable tanto para usarla como para verla.