

Virtual Vault

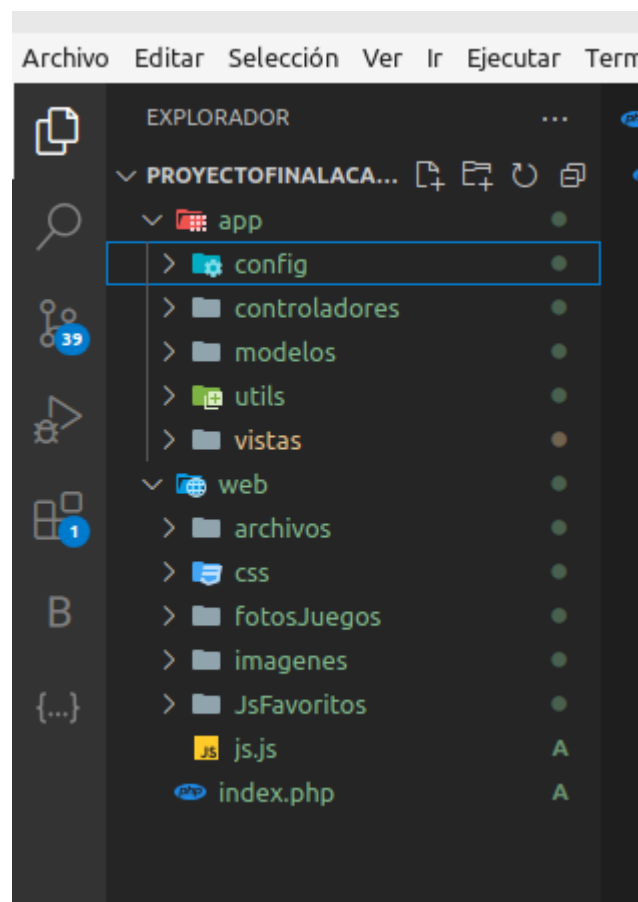
Indice

1. Introducción.....	3
2. Estructura de carpetas.....	3
3. BackEnd.....	4
3.1 Controladores.....	5
3.2 Modelos.....	7
3.3 Utils.....	9
3.4 Vistas.....	10
3.5 JS.....	11
4. FrontEnd.....	14
5. Extras.....	15

1. Introducción

En este documento se va a explicar un poco por encima las partes del código de la aplicación.

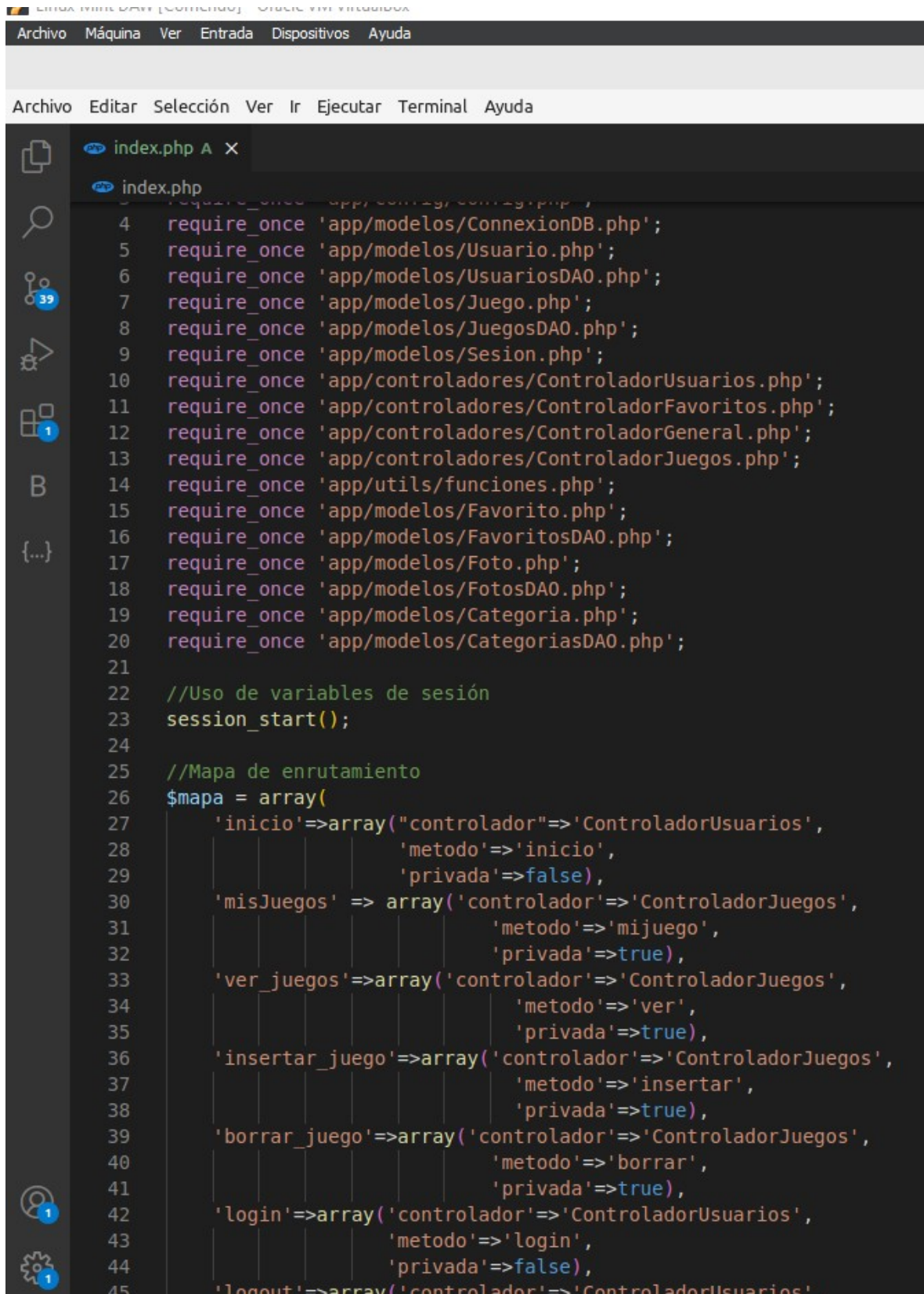
2. Estructura de carpetas



Esta sería mi estructura de carpetas del proyecto. Esta dividida en dos, la parte de web que es donde se guardan las imagenes, los estilos, el código de javaScript y las fotos de los juegos.

En la otra parte se encuentra toda la chicha de la aplicación.

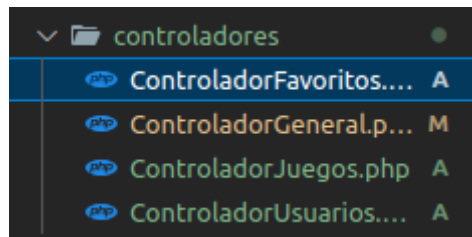
3. BackEnd



```
index.php A X
index.php
1 require_once 'app/controladores/ControladorUsuarios.php';
2 require_once 'app/controladores/ControladorFavoritos.php';
3 require_once 'app/controladores/ControladorGeneral.php';
4 require_once 'app/controladores/ControladorJuegos.php';
5 require_once 'app/modelos/ConexionDB.php';
6 require_once 'app/modelos/Usuario.php';
7 require_once 'app/modelos/UsuariosDAO.php';
8 require_once 'app/modelos/Juego.php';
9 require_once 'app/modelos/JuegosDAO.php';
10 require_once 'app/modelos/Sesion.php';
11 require_once 'app/controladores/ControladorUsuarios.php';
12 require_once 'app/controladores/ControladorFavoritos.php';
13 require_once 'app/controladores/ControladorGeneral.php';
14 require_once 'app/controladores/ControladorJuegos.php';
15 require_once 'app/utils/funciones.php';
16 require_once 'app/modelos/Favorito.php';
17 require_once 'app/modelos/FavoritosDAO.php';
18 require_once 'app/modelos/Foto.php';
19 require_once 'app/modelos/FotosDAO.php';
20 require_once 'app/modelos/Categoria.php';
21 require_once 'app/modelos/CategoriasDAO.php';
22
23 //Uso de variables de sesión
24 session_start();
25
26 //Mapa de enrutamiento
27 $mapa = array(
28     'inicio'=>array('controlador'=>'ControladorUsuarios',
29                     'metodo'=>'inicio',
30                     'privada'=>false),
31     'misJuegos' => array('controlador'=>'ControladorJuegos',
32                          'metodo'=>'mijuego',
33                          'privada'=>true),
34     'ver_juegos'=>array('controlador'=>'ControladorJuegos',
35                        'metodo'=>'ver',
36                        'privada'=>true),
37     'insertar_juego'=>array('controlador'=>'ControladorJuegos',
38                            'metodo'=>'insertar',
39                            'privada'=>true),
40     'borrar_juego'=>array('controlador'=>'ControladorJuegos',
41                          'metodo'=>'borrar',
42                          'privada'=>true),
43     'login'=>array('controlador'=>'ControladorUsuarios',
44                   'metodo'=>'login',
45                   'privada'=>false),
46     'logout'=>array('controlador'=>'ControladorUsuarios',
```

Primero se encuentra el index que es la página que se encarga de realizar el mapeo del modelo vista controlador.

3.1 Controladores



Aquí se encuentran las direcciones para que las vistas puedan ser vistas sin ningún problema y también es donde se encuentra el código de la funcionalidad de la página. Como por ejemplo:

```
Codeium: Refactor | Explain | Generate Function Comment | x
public function insertar(){

    $error = '';

    $connexionDB = new ConnexionDB(MYSQL_USER,MYSQL_PASS,MYSQL_HOST,MYSQL_DB);
    $conn = $connexionDB->getConnexion();

    $juegosDAO = new JuegosDAO($conn);
    $categoriasDAO = new CategoriasDAO($conn);
    $juegos = $juegosDAO->obtenerJuegoPorIdUsuario(Sesion::getUsuario()->getId());

    if($_SERVER['REQUEST_METHOD']=='POST'){
        $titulo = htmlspecialchars($_POST['titulo']);
        $descripcion = htmlspecialchars($_POST['descripcion']);
        $precio = htmlspecialchars($_POST['precio']);
        $idCategoria = htmlspecialchars($_POST['idCategoria']);

        if(empty($titulo)||empty($descripcion)||empty($precio)||empty($idCategoria)){
            $error = "Los campos son obligatorios";
        }
        else{
            $array_fotos = array();
            $array_fotosTMP = array();
            $array_fotosINS = array();

            if ($_FILES['fotos']['error'][0] == UPLOAD_ERR_NO_FILE) {
                $error = "Debes añadir al menos una foto al juego";
            } elseif (count($_FILES['fotos']['name']) > 7) {
                $error = "No puedes subir más de 7 fotos a un juego";
            } else {
                $num_files = count($_FILES['fotos']['name']);

                for ($i = 0; $i < $num_files; $i++) {
                    $array_fotos[] = $_FILES['fotos']['name'][$i];
                    $array_fotosTMP[] = $_FILES['fotos']['tmp_name'][$i];
                }
            }
        }
    }
}
```

```

foreach ($array_fotos as $i => $foto) {
    // Comprobamos que la extensión de los archivos introducidos son válidas
    $extension = pathinfo($foto, PATHINFO_EXTENSION);
    if ($extension != 'jpg' && $extension != 'jpeg' && $extension != 'png') {
        $error = "Alguna de las fotos no tiene un formato admitido, deben de ser jpg, jpeg o png";
    } else {
        // Copiamos la foto al disco
        // Calculamos un hash para el nombre del archivo
        $foto = uniqid(true) . '.' . $extension;

        // Si existe un archivo con ese nombre volvemos a calcular el hash
        while (file_exists("web/fotosJuegos/$foto")) {
            $foto = uniqid(true) . '.' . $extension;
        }

        foreach ($array_fotosTMP as $j => $fotoTMP) {
            if ($i == $j && $error == '') {
                if (!move_uploaded_file($fotoTMP, "web/fotosJuegos/$foto")) {
                    die("Error al copiar la foto a la carpeta fotosJuegos");
                }
            }
        }

        $array_fotosINS[] = $foto;
    }
}

if($error == ''){
    $juego = new Juego();
    $juego->setTitulo($titulo);
    $juego->setDescripcion($descripcion);
    $juego->setIdCategoria($idCategoria);
    $juego->setPrecio($precio);
    $juego->setIdUsuario($sesion->getUsuario()->getId());
}

```

```

        $idJ = $juegosDAO->insert($juego);

        $juego->setIdCategoria($categoriasDAO->getNombreById($idCategoria));

        if($idJ != null){
            $fotosDAO = new FotosDAO($conn);
            $fotosDAO->insertarVariasFotos($array_fotosINS, $idJ);

            // Asegúrate de que $array_fotosINS[0] contiene el nombre de la primera foto
            $fotoUrl = "web/fotosJuegos/" . $array_fotosINS[0];
            print json_encode(['respuesta' => 'ok', 'juego' => $juego->toJSON($idJ), 'primerafoto' => $fotoUrl]);
        } else {
            print json_encode(['respuesta' => 'error']);
        }
    }
}
}

```

Este sería el código para poder insertar un juego, incluyendo una o varias fotos y las categorías.

3.2 Modelos

En los modelos se divide en dos que son las clases y los daos ahora a continuación un ejemplo tanto de clase como de dao.

```
<?php

Codeium: Refactor | Explain
class Juego {
    private $id;
    private $precio;
    private $titulo;
    private $descripcion;
    private $fecha_creacion;
    private $idUserario;
    private $idCategoria;
    private $foto;

    /**
     * Get the value of id
     */
    Codeium: Refactor | Explain | ×
    public function getId() {
        return $this->id;
    }

    /**
     * Set the value of id
     */
    Codeium: Refactor | Explain | ×
    public function setId($id): self {
        $this->id = $id;
        return $this;
    }

    /**
     * Get the value of precio
     */
    Codeium: Refactor | Explain | ×
    public function getPrecio() {
        return $this->precio;
    }

    /**
     * Set the value of precio
     */
    Codeium: Refactor | Explain | ×
    public function setPrecio($precio): self {

```

[File Link](#) [Generate Commit Message](#) [Explain Code](#) [Comment Code](#) [Find Bugs](#) [Co](#)

Así se ve el contenido de una clase con bastante contenido. A continuación su DAO

```
modelos / JuegosDAO.php
class JuegosDAO {
}

Codeium: Refactor | Explain | Generate Function Comment | x
public function getAll():array{
    //$this->conn->prepare() devuelve un objeto de la clase mysqli_stmt
    if (!$stmt = $this->conn->prepare("SELECT * FROM juegos ORDER BY fecha_creacion DESC")) {
        echo "Error en la SQL: " . $this->conn->error;
    }
    //Ejecutamos la SQL
    $stmt->execute();
    //Obtener el objeto mysql_result
    $result = $stmt->get_result();

    $array_juegos = array();

    while($juegos = $result->fetch_object(Juego::class)){
        $array_juegos[] = $juegos;
    }
    return $array_juegos;
}

Codeium: Refactor | Explain | Generate Function Comment | x
public function obtenerJuegoPorIdUsuario($idUsuario) {
    if (!$stmt = $this->conn->prepare("SELECT * FROM juegos WHERE idUsuario = ?")) {
        echo "Error en la SQL: " . $this->conn->error;
    }
    //Asociar las variables a las interrogaciones (parámetros)
    $stmt->bind_param('i',$idUsuario);
    //Ejecutamos la SQL
    $stmt->execute();
    //Obtener el objeto mysql_result
    $result = $stmt->get_result();

    $array_misjuegos = array();

    while($juego = $result->fetch_object(Juego::class)){
        $array_misjuegos[] = $juego;
    }
    return $array_misjuegos;
}
```

Como se puede ver en este DAO son consultas SQL como por ejemplo GETALL que es para coger todos los juegos y ObtenerJuegoPorIdUsuario que como su nombre indica es para seleccionar los juegos de un usuario en concreto.

3.3 Utils

```
<?php
/**
 * Genera un hash aleatorio para un nombre de archivo manteniendo la extensión original
 */
Codeium: Refactor | Explain | ×
function generarNombreArchivo(string $nombreOriginal):string {
    $nuevoNombre = md5(time()+rand());
    $partes = explode('.', $nombreOriginal);
    $extension = $partes[count($partes)-1];
    return $nuevoNombre.'.'.$extension;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
function guardarMensaje($mensaje){
    $_SESSION['error']=$mensaje;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
function imprimirMensaje(){
    if(isset($_SESSION['error'])){
        echo '<div class="error" id="mensajeError">'.$_SESSION['error'].'</div>';
        unset($_SESSION['error']);
    }
}
```

Aquí está el imprimirMensajes que sirve para el control de errores.

3.4 Vistas

```
VerJuegos.php M X
app > vistas > VerJuegos.php
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Virtual Vault</title>
7   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" integrity="sha384-QWTKZyjpPEjISv5WaRU9
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9
9   <style>
10     .card {
11       width: 250px;
12       margin: 15px;
13     }
14     .card-img-top {
15       height: 200px;
16       object-fit: cover;
17     }
18     .card-title, .card-text {
19       white-space: nowrap;
20       overflow: hidden;
21       text-overflow: ellipsis;
22     }
23
24     body{
25       background-image: url(web/imagenes/fondoTodosLosJuegos.png);
26       background-attachment: fixed;
27       background-repeat: no-repeat;
28     }
29   </style>
30 </head>
31 <body>
32   <header>
33     <nav class="navbar navbar-expand-lg bg-body-tertiary">
34       <div class="container-fluid">
35         <a class="navbar-brand" href="index.php?accion=ver_juegos">Virtual Vault</a>
36         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
37         <span class="navbar-toggler-icon"></span>
38       </button>
39       <div class="collapse navbar-collapse" id="navbarNav">
40         <ul class="navbar-nav">
41           <li class="nav-item">
42             <a class="nav-link active" aria-current="page" href="index.php?accion=misJuegos">Mis Juegos</a>
```

```

43           <li class="nav-item">
44             <a class="nav-link" href="index.php?accion=sobreMi">Sobre Mi</a>
45           </li>
46           <li class="nav-item">
47             <a class="nav-link" href="index.php?accion=logout">Cerrar Sesión</a>
48           </li>
49         </ul>
50       </div>
51     </nav>
52   </header>
53
54   <div class="container">
55     <div class="d-flex flex-wrap">
56       <?php foreach ($juegosConFoto as $i => $juego): ?>
57         <div class="card flex">
58           <a href="index.php?accion=ver_dentro&id=?= $juego->getId()" ?> class="w-100" style="text-decoration: none; color: black;">
59             <div>
60               getTitulo() ?>">
61               <div class="card-body">
62                 <h5 class="card-title"><?= $juego->getTitulo() ?></h5>
63                 <p class="card-text"><?= $juego->getDescripcion() ?></p>
64                 <p class="card-text"><?= $juego->getIdCategoria() ?></p>
65                 <p class="card-text"><small class="text-muted"><?= $juego->getPrecio() ?> $</small></p>
66               </div>
67             </div>
68           </a>
69           <input class="idJuego" type="hidden" value="<?= $juego->getId()" ?>">
70           <input class="idUserario" type="hidden" value="<?= $juego->getIdUsuario()" ?>">
71           <?php
72             $isFavorite = false;
73             if ($favoritosDeUsuario != null) {
74               foreach ($favoritosDeUsuario as $j => $favorito) {
75                 if ($favorito->getId() == $juego->getId()) {
76                   $isFavorite = true;
77                 }
78               }
79             }
80           </?php>
81         </div>
82       </?php>
83     </div>
84   </div>
85 </body>
86 </html>
```

```

        foreach ($favoritosDeUsuario as $j => $favorito) {
            if ($favorito == $juego->getId()) {
                $isFavorite = true;
                break;
            }
        }

        ?>
        <?php if ($isFavorite): ?>
            <i class="fa-solid fa-star m-2 p-2 w-100" id="favorito" style="color: #FFD43B;"></i>
        <?php else: ?>
            <i class="fa-regular fa-star m-2 p-2 w-100" id="nofavorito"></i>
        <?php endif; ?>
    </div>
<?php endforeach; ?>
</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNKmXc5s9fDVZLEsAA55NDz0xhy9GkcIdslK1e"
<script src="web/JsFavoritos/favoritos.js"></script>
</body>
</html>

```

Estas es por ejemplo la vista de ver todos los juegos y para ello se necesita hacer un foreach, también se puede ver el código para ver si se ha marcado el favorito.

3.5 JS

```

let botonInsertar = document.getElementById('botonNuevoJuego');

botonInsertar.addEventListener('click', function () {
    let fotos = document.getElementById('fotos').files;

    // Enviamos los datos mediante el POST construyendo el FormData
    const datos = new FormData();
    for (var i = 0; i < fotos.length; i++) {
        datos.append('fotos[]', fotos[i]);
    }
    datos.append('titulo', document.getElementById('titulo').value);
    datos.append('descripcion', document.getElementById('descripcion').value);
    datos.append('precio', document.getElementById('precio').value);
    datos.append('idCategoria', document.getElementById('categoria').value);

    const options = {
        method: "POST",
        body: datos
    };

    fetch("index.php?accion=insertar_juego", options)
    .then(respuesta => {
        return respuesta.json();
    })
    .then(data => {
        if (data.respuesta === 'ok') {
            const juego = JSON.parse(data.juego);

            // Crear el contenedor para el nuevo juego
            let juegoDiv = document.createElement('div');
            juegoDiv.className = 'juego';

            // Título del juego
            let titulo = document.createElement('div');
            titulo.innerText = 'Título: ' + juego.titulo;
            juegoDiv.appendChild(titulo);

            // Descripción del juego
            let descripcion = document.createElement('div');
            descripcion.innerText = 'Descripción: ' + juego.descripcion;
            juegoDiv.appendChild(descripcion);

```

```

// Añadir la primera foto del juego
if (data.primerafoto) {
    let fotosDiv = document.createElement('div');
    fotosDiv.className = 'fotos';
    let img = document.createElement('img');
    img.src = data.primerafoto;
    fotosDiv.appendChild(img);
    juegoDiv.appendChild(fotosDiv);
}

// Categoria del juego
let categoria = document.createElement('div');
categoria.innerText = 'Categoria: ' + juego.idCategoria;
juegoDiv.appendChild(categoria);

// Precio del juego
let precio = document.createElement('div');
precio.innerText = 'Precio: ' + juego.precio + ' $';
juegoDiv.appendChild(precio);

// Añadir el icono de la papelera
var papelera = document.createElement('i');
papelera.className = 'fa-solid fa-trash papelera';
papelera.setAttribute('data-idJuego', juego.id);
juegoDiv.appendChild(papelera);

// Añadir el nuevo juego al contenedor de juegos
document.getElementById('contenedorJuegos').appendChild(juegoDiv);

// Cerrar el modal
var myModalEl = document.getElementById('modalCrear');
var modal = bootstrap.Modal.getInstance(myModalEl);
modal.hide();

document.getElementById('titulo').value = '';
document.getElementById('descripcion').value = '';
document.getElementById('precio').value = '';
document.getElementById('categoria').value = '';
document.getElementById('fotos').value = '';

```



```

        //Añadir manejador de evento Borrar a la nueva papelera
        papelera.addEventListener('click',manejadorBorrar);

    } else {
        // Manejo de errores (opcional)
        alert('Error al insertar el juego: ' + data.error);
    }
}
})
.catch(error => {
    console.error('Error:', error);
});
});

let papeleras = document.querySelectorAll('.papelera');
papeleras.forEach(papelera => {
    papelera.addEventListener('click',manejadorBorrar);
});

Codeium: Refactor | Explain | Generate JSDoc | x
function manejadorBorrar(){
    //this referencia al elementos del DOM sobre el que hemos hecho click
    var idJuego= this.getAttribute('data-idJuego');
    //Llamamos al script del servidor que borra la tarea pasándole el idTarea como parámetro
    fetch('index.php?accion=borrar_juego&id='+idJuego)
    .then(datos => datos.json())
    .then(respuesta =>{
        if(respuesta.respuesta=='ok'){
            this.parentElement.remove();
        }
        else{
            alert("No se ha encontrado el juego en el servidor");
            this.style.visibility='visible';
        }
    })
}
}

```

Este es el código de JS en el cual viene tanto el insertar dinamicamente con fetch como borrar el juego. En todo momento hay que pasarle identificadores como por ejemplo el id del juego.

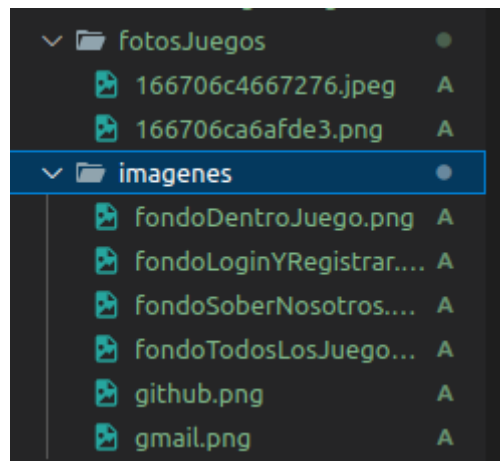
4. FrontEnd

Aquí es donde van a ir los estilos del código. Se ha de tener en cuenta que algunos han sido puestos con Bootstrap por lo cual ya están incluidos en el código de las vistas.

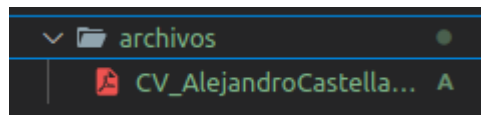
```
web > css > estilosdentroJuego.css > .photos
1  body {
2      font-family: Arial, sans-serif;
3      background-color: #f8f9fa;
4      padding: 20px;
5
6      background-image: url(web/imagenes/fondoSobreMi.png);
7      background-attachment: fixed;
8      background-repeat: no-repeat;
9  }
10 h1 {
11     font-size: 2.5rem;
12     margin-bottom: 20px;
13 }
14 p {
15     font-size: 1.1rem;
16     margin-bottom: 10px;
17 }
18 .game-details strong {
19     font-weight: bold;
20 }
21 .photos {
22     display: flex;
23     flex-wrap: wrap;
24     gap: 10px;
25     margin-top: 20px;
26 }
27 .photos img {
28     width: 200px;
29     height: auto;
30     border: 2px solid #dee2e6;
31     border-radius: 5px;
32     box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
33 }
34
```

Este sería por ejemplo el código de los estilos de la vista de dentro de los juegos.

5. Extras



En la carpeta de FotosJuegos es donde se guardan las fotos de los juegos que se van a ir creando y en imágenes las imágenes de los estilos



En la carpeta archivos es donde se guardar el currículum que en la vista sobre mi es descargado.