• Nombre del proyecto: Tienda online de juegos de mesa Board this way.

• Se trata de la página web para una tienda online que dispone de:

- Un CRUD completo (Create, Read, Update, Delete).
- Filtrado de juegos por género.
- Diseño con tarjetas horizontales.
- Login/logout funcional.
- Roles (admin vs user).
- Botones y menús adaptados a los roles de los usuarios.
- Redirecciones inteligentes.
- Feedback funcional mediante alertas.
- Navegación simple e intuitiva.

• Tecnologías utilizadas:

- Editor de texto Visual Studio Code.
- Docker para lanzar el contenedor donde estará todo el proyecto.
- Base de datos en PostgreSQL.
- PHP.
- HTML y CSS.
- Librería Config para ejecutar el patrón Singleton.
- Carpeta Vendor con todas las librerías necesarias incluidala que permite vincular la base de datos al proyecto en PHP.

• Estructura del proyecto: En la raíz del proyecto están los ficheros de configuración y las carpetas principales:

- .env
- composer.lock
- composer.json
- dockerfile
- docker-compose.yml
- database (contiene el fichero con la base de datos init.sql)
- tests (vacía de momento)
- vendor (contiene todas las librerías)
- src (en detalle a continuación)

En "src" hay:

- config (config.php)
- models (contiene las clases: genero.php, producto.php y user.php)
- services (contiene los controladores: GeneroServices.php, ProductosServices.php, SessionServices.php y UserServices.php)
- uploads (contiene el favicon.png)
- create.php (página de creación de un producto)
- delete.php (función de borrado de un producto)
- details.php (página donde se muestran los detalles de un producto)
- footer.php (bloque que muestra el pie de la página web)
- header.php (bloque que muestra el encabezado de la página web)

- 📄 index.php (página principal de la web)
- 📄 login.php (página de inicio de sesión)
- 📄 logout.php (función de cerrar sesión)
- 📄 update.php (página para editar la información de un producto)

• Instrucciones de instalación:

- Primeramente se creó la carpeta del proyecto donde se guardaron los ficheros de configuración iniciales.
- Antes de lanzar Docker, se instalaron las dependencias con composer install en la máquina local para generar la carpeta vendor/.
- Una vez hecho esto, y con los ficheros de dockerfile y docker-compose.yml configurados con el código dado, se ejecutó docker-compose up -d --build y se lanzó el entorno.
- La carpeta src se genera automáticamente y dentro ya se puede crear la estructura de carpetas y ficheros del proyecto.
- En este caso la base de datos se configuró con la estructura adecuada al final, empleándose para el proceso de instalación una previamente proporcionada.

• Uso básico:

- Para acceder a la aplicación se puede hacer desde localhost o clickando en el enlace de Docker.
- Para navegar por ella están las siguientes páginas:
  - 🏠 El main menú o página principal es la primera página que se ve, desde el encabezado se puede navegar a las demás. El nombre de la web o "Lista de productos" redirigen a esta página.
  - 📝 Añadir producto redirige a la página de crear un artículo para guardarlo en la BBDD (solo disponible con inicio de sesión y privilegios de administrador).
  - 🔐 Login redirige a una página para insertar las credenciales de inicio de sesión.
  - 🔎 La barra de búsqueda de la página principal permite filtrar los productos por género. En las tarjetas de producto se encuentran las siguiente funciones:
  - ℹ️ Detalles permite ver toda la información de cada producto (función disponible para todos).
  - ✏️ Editar redirige a una página para cambiar la información de un producto (solo disponible con inicio de sesión y privilegios de administrador).
  - 🗑️ Eliminar abre un modal que hay que aceptar para borrar un producto (solo disponible con inicio de sesión y privilegios de administrador).
  - 🚪 Logout es un botón que aparece en el encabezado solo si se ha iniciado sesión para cerrarla.

• Lógica del proyecto:

1. Gestión de la base de datos:
   - Estructura de tablas:
     1. usuarios (id, apellidos, email, nombre, password, username).
     2. productos (nombre, precio, stock, id, genero_id, uuid, imagen, num_jugadores, tipo).
     3. generos (id, nombre).
     4. user_roles (user_id, roles).
   - Características técnicas:
     - IDs autoincrementales para productos y usuarios.
     - UUIDs para géneros, productos y relaciones.
     - Claves únicas en username y email.

        - Integridad referencial con foreign keys.
2. CRUD de productos:
    - Create:
        1. Formulario para validar los datos obligatorios (nombre, precio, stock, etc.).
        2. Generación de UUID único para cada producto.
        3. PostgreSQL asigna un ID autoincremental automáticamente.
        4. Establece relación con género mediante UUID.
    - Read:
        1. findAll(): Obtiene todos los productos.
        2. findById(): Busca un producto específico por ID.
        3. findAllWithGenreName(): Filtra por género con JOIN.
        4. Los productos se muestran en tarjetas horizontales.
    - Update:
        1. Recibe ID del producto a editar por parámetro GET.
        2. Rellena el formulario con datos actuales.
        3. Reordena géneros para mostrar el actual primero.
        4. Actualiza solo los campos modificados.
    - Delete:
        1. Confirmación JavaScript antes de eliminar.
        2. Eliminación física directa en base de datos.
        3. Alertas de éxito/error con redirección.
3. Gestión de usuarios y roles:
    - Roles:
        - Administrador: Acceso completo (CRUD).
        - Usuario: Solo lectura (ver productos).
        - Visitante: Solo login y ver productos.
    - Autenticación:
        1. Verifica credenciales (username + password).
        2. Consulta roles del usuario en tabla user_roles.
        3. Inicia sesión con datos de usuario y roles.
        4. Redirige automáticamente al index.
    - Control de accesos:
        - Botones "Editar/Eliminar" ocultos para no-administradores.
        - Páginas CRUD protegidas con requireAdmin().
        - El encabezado muestra el rol actual (Admin/User).
4. Gestión de categorías y productos:
    - Relación: Productos → genero_id (UUID) → Géneros (id).
    - Menú dropdown de géneros que se reordena dinámicamente en modo edición.
    - Filtrado que mantiene los valores del formulario.
5. Seguridad del sistema:
    - Autenticación:
        - Sesiones con expiración (1 hora).
        - Verificación de credenciales segura.
        - Logout limpio de variables de sesión.
    - Autorización:
        - Verificación en dos niveles: isLoggedIn() + isAdmin().

        - Redirecciones automáticas si no tiene permisos.
    - Validaciones:
        - Filtrado de inputs con filter_input().
        - Sanitización de datos antes de guardar.
        - Validación de tipos de datos (float, int).
        - Manejo de errores con try-catch.

• Requisitos previos:

- PHP 7.4+ con extensiones: pdo_pgsql, session
- Docker 4.48.0
- PostgreSQL 12+
- Apache 2.0
- Ramsey/UUID: Generación de identificadores únicos.
- vlucas/phpdotenv: Gestión de variables de entorno.
- Configuración adicional:
    - Conexión a PostgreSQL con variables de entorno.
    - Patrón Singleton para una única instancia de conexión.
    - Configuración de rutas y uploads.

• Autores y créditos:

- Ángel José García Ibáñez
- GitHub: https://github.com/Angelgarib/Proyecto_TiendaOnline.git

• Licencia para uso educativo.