

Propuesta de **Trabajo** **Inari Plays**

Bárbara Seguí Navarro
Alejandro Reyes Albillar
Joaquin Vasalo Vicedo
Raquel García Pastor
Gema Moreno Compañ

ÍNDICE

1. Introducción

2. Diseño

2.1. Diseño de la base de datos

2.2. Esquema de diseño Web

3. Descripción proyecto

1.Introducción

Hemos decidido realizar, para nuestro proyecto web, un comercio electrónico. Se trata de una tienda online de venta de videojuegos, música, merchandising y películas.

El nombre que hemos decidido para el proyecto es : ***Inari Plays***

Nuestro logo es:



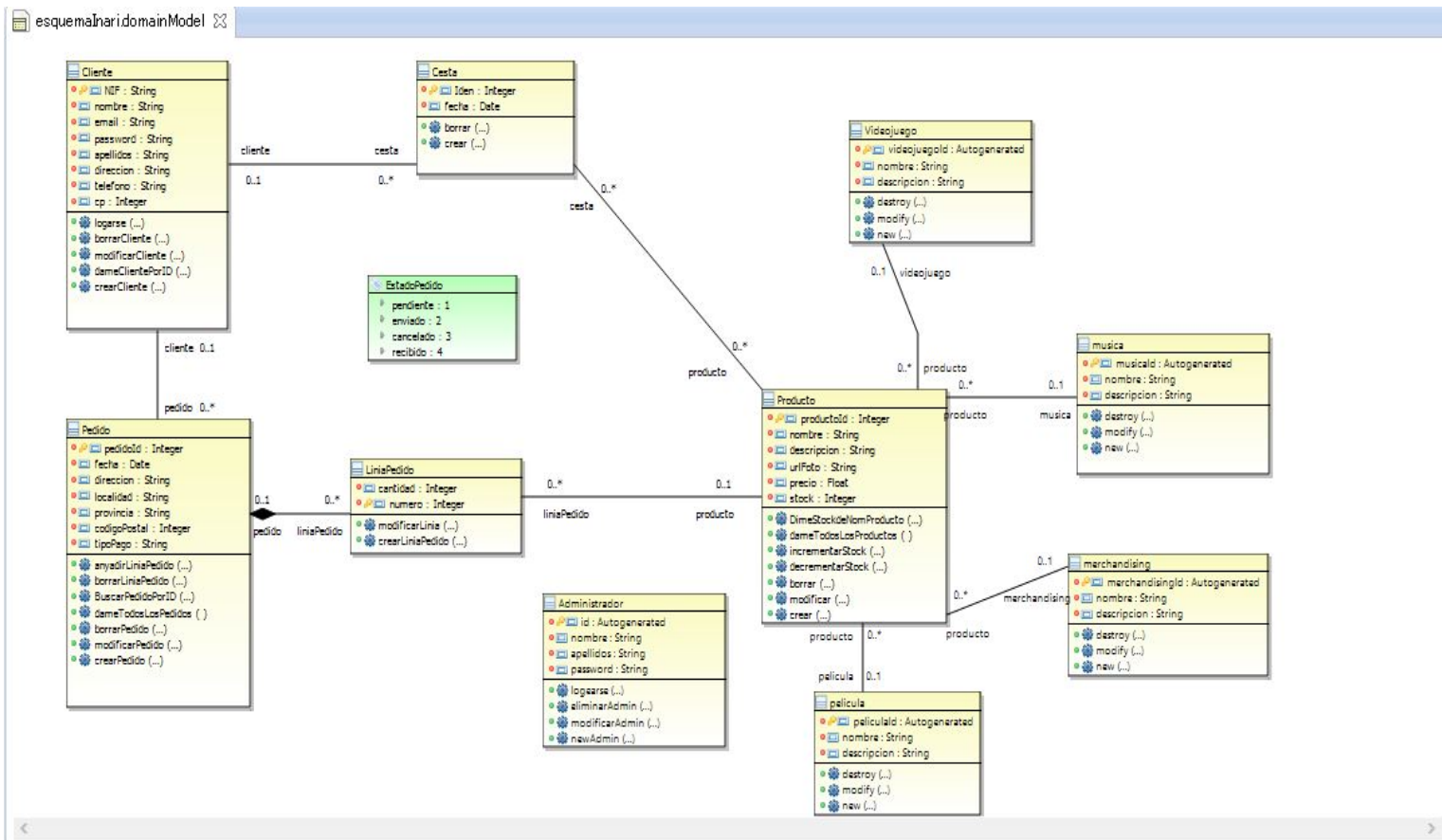
Las herramientas utilizadas para la realización del proyecto van a ser:

- **OOH4RIA** : Proporcionada por la UA de la asignatura DSS de tercer año. Esta herramienta sirve para diseñar bases de datos desde una interfaz visual intuitiva para luego autogenerar soluciones en Microsoft Visual Studio 2010.
- **Microsoft Visual Studio** : Herramienta que nos va a servir para programar los CEN, CAD, EN originados en la herramienta OOH4RIA , y para interactuar con la base de datos del proyecto.
- **Git**: Herramienta para almacenar los pasos del proyecto y llevar un control exhaustivo del mismo.

2.Diseño

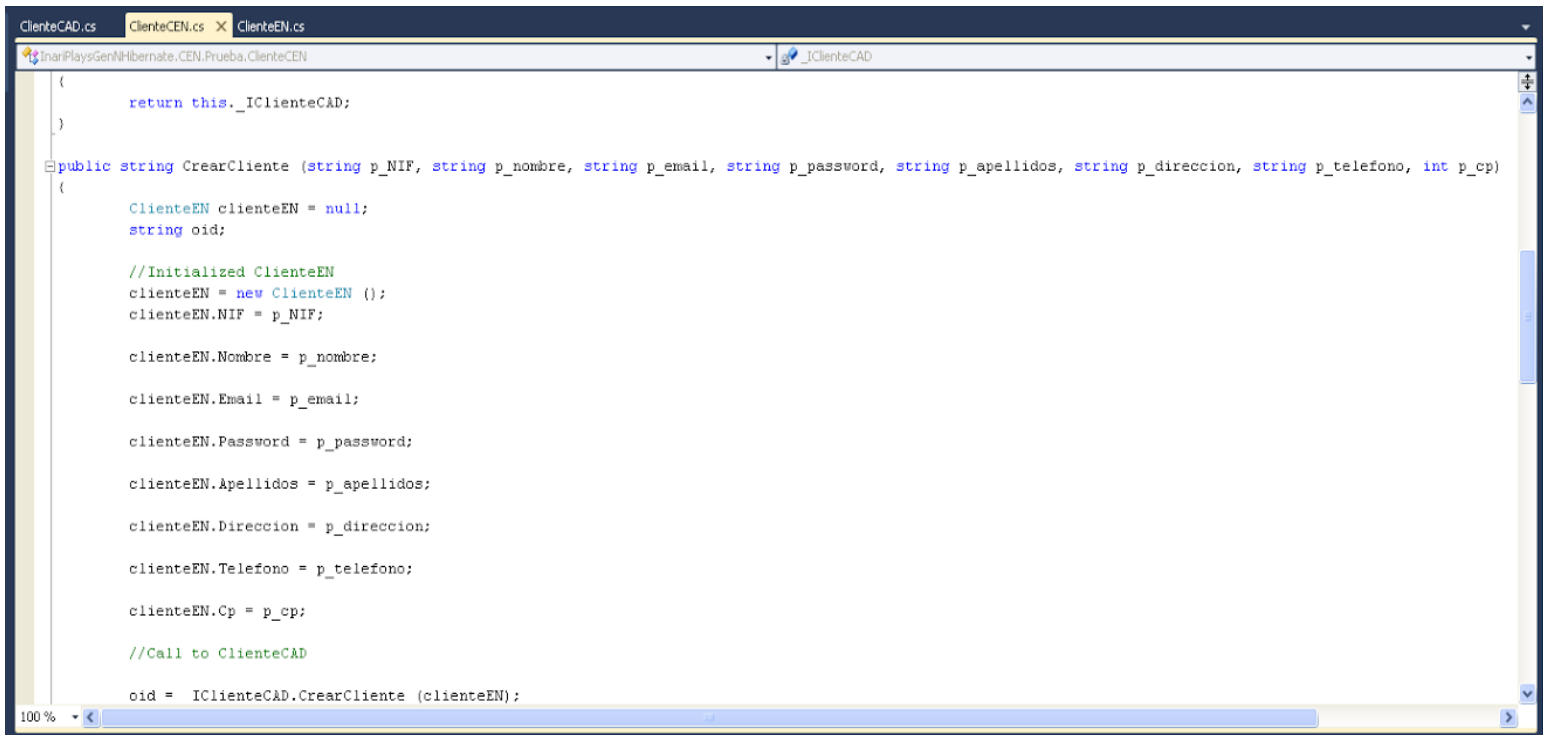
2.1.Diseño de la base de datos

Para empezar, hemos diseñado la base de datos en la herramienta OOH4RIA dando como resultado el siguiente esquema UML:



Una vez acabado el esquema, el programa autogenera una solución en Microsoft Visual Studio, con los componentes de negocio, los CEN, que actúan como interfaz entre la CAD y la EN, las CAD, también conocidas como Componentes de Acceso a Datos que estarán encargadas de interactuar con la BBDD y las EN, o Entidades de Negocio que se encargan de definir los métodos que necesitan de los datos que proporcionan las CAD.

Por ejemplo un CEN sería:



The screenshot shows a code editor with three tabs: ClienteCAD.cs, ClienteCEN.cs, and ClienteEN.cs. The active tab is ClienteEN.cs. The code defines a method CrearCliente that takes several parameters and creates a new ClienteEN object, setting its properties and then calling the CrearCliente method on the ClienteCAD interface.

```
{
    return this._IClienteCAD;
}

public string CrearCliente (string p_NIF, string p_nombre, string p_email, string p_password, string p_apellidos, string p_direccion, string p_telefono, int p_cp)
{
    ClienteEN clienteEN = null;
    string oid;

    //Initialized ClienteEN
    clienteEN = new ClienteEN ();
    clienteEN.NIF = p_NIF;

    clienteEN.Nombre = p_nombre;

    clienteEN.Email = p_email;

    clienteEN.Password = p_password;

    clienteEN.Apellidos = p_apellidos;

    clienteEN.Direccion = p_direccion;

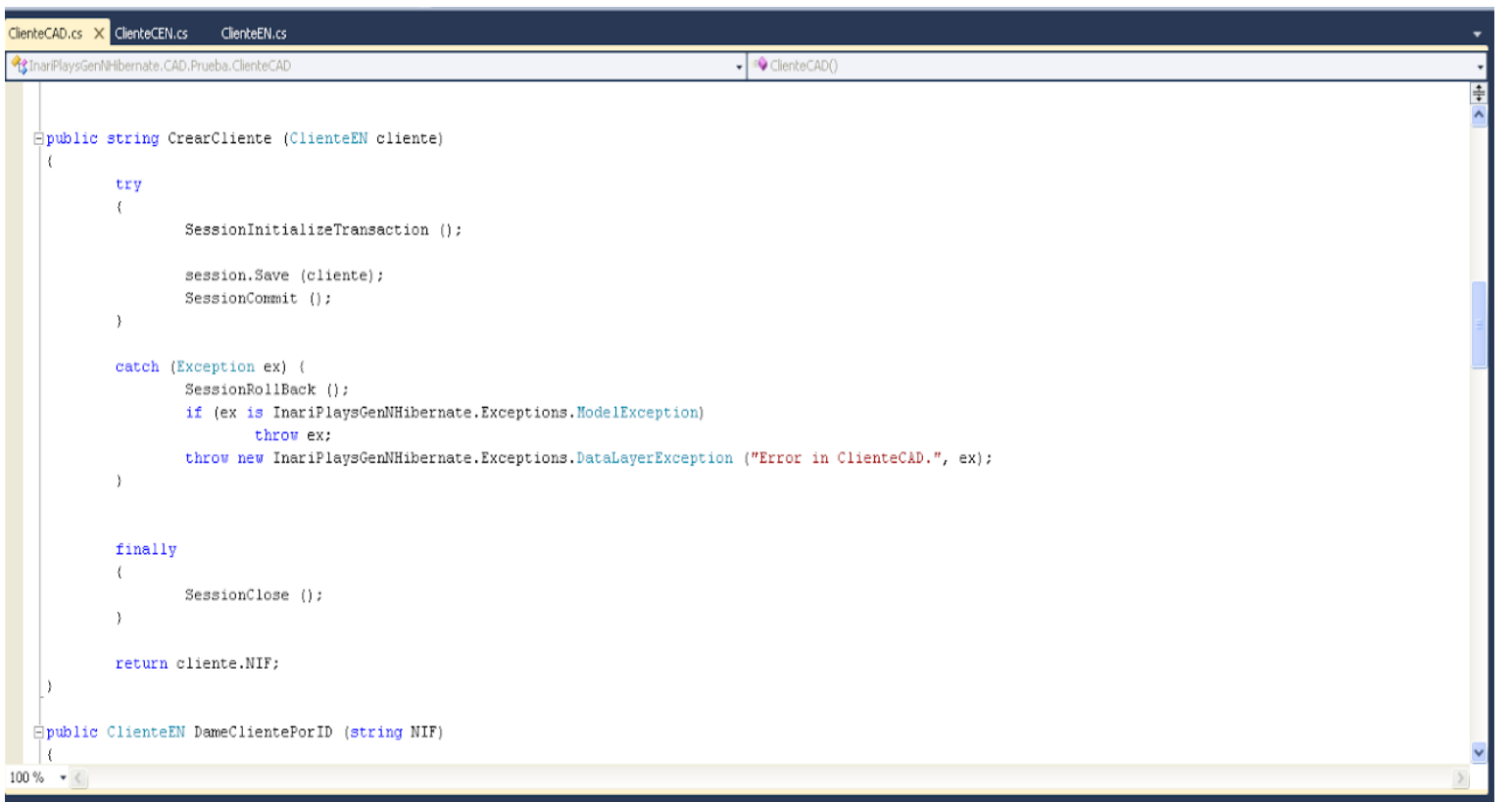
    clienteEN.Telefono = p_telefono;

    clienteEN.Cp = p_cp;

    //Call to ClienteCAD

    oid = IClienteCAD.CrearCliente (clienteEN);
}
```

Una CAD:



The screenshot shows a code editor with three tabs: ClienteCAD.cs, ClienteCEN.cs, and ClienteEN.cs. The active tab is ClienteCAD.cs. The code defines a method CrearCliente that takes a ClienteEN object and uses Hibernate to save it to the database. It includes try-catch-finally blocks for transaction management and error handling.

```
public string CrearCliente (ClienteEN cliente)
{
    try
    {
        SessionInitializeTransaction ();

        session.Save (cliente);
        SessionCommit ();
    }

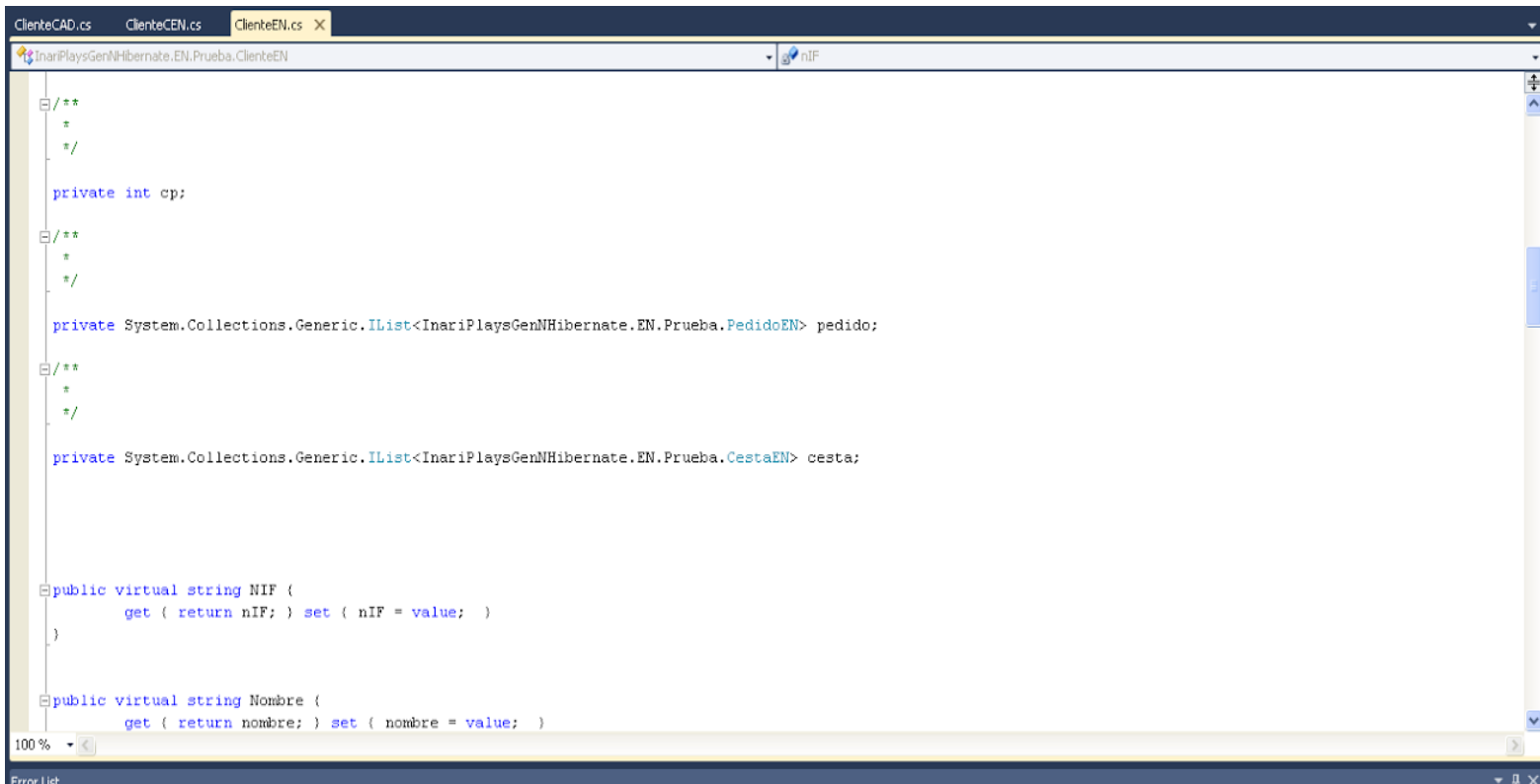
    catch (Exception ex) {
        SessionRollBack ();
        if (ex is InariPlaysGenNHibernate.Exceptions.ModelException)
            throw ex;
        throw new InariPlaysGenNHibernate.Exceptions.DataLayerException ("Error in ClienteCAD.", ex);
    }

    finally
    {
        SessionClose ();
    }

    return cliente.NIF;
}

public ClienteEN DameClientePorID (string NIF)
{
}
```

Y una EN:



```
ClienteCAD.cs  ClienteCEN.cs  ClienteEN.cs x
InariPlaysGenNHibernate.EN.Prueba.ClienteEN  nIF

/**
 *
 */
private int cp;

/**
 *
 */
private System.Collections.Generic.IList<InariPlaysGenNHibernate.EN.Prueba.PedidoEN> pedido;

/**
 *
 */
private System.Collections.Generic.IList<InariPlaysGenNHibernate.EN.Prueba.CestaEN> cesta;

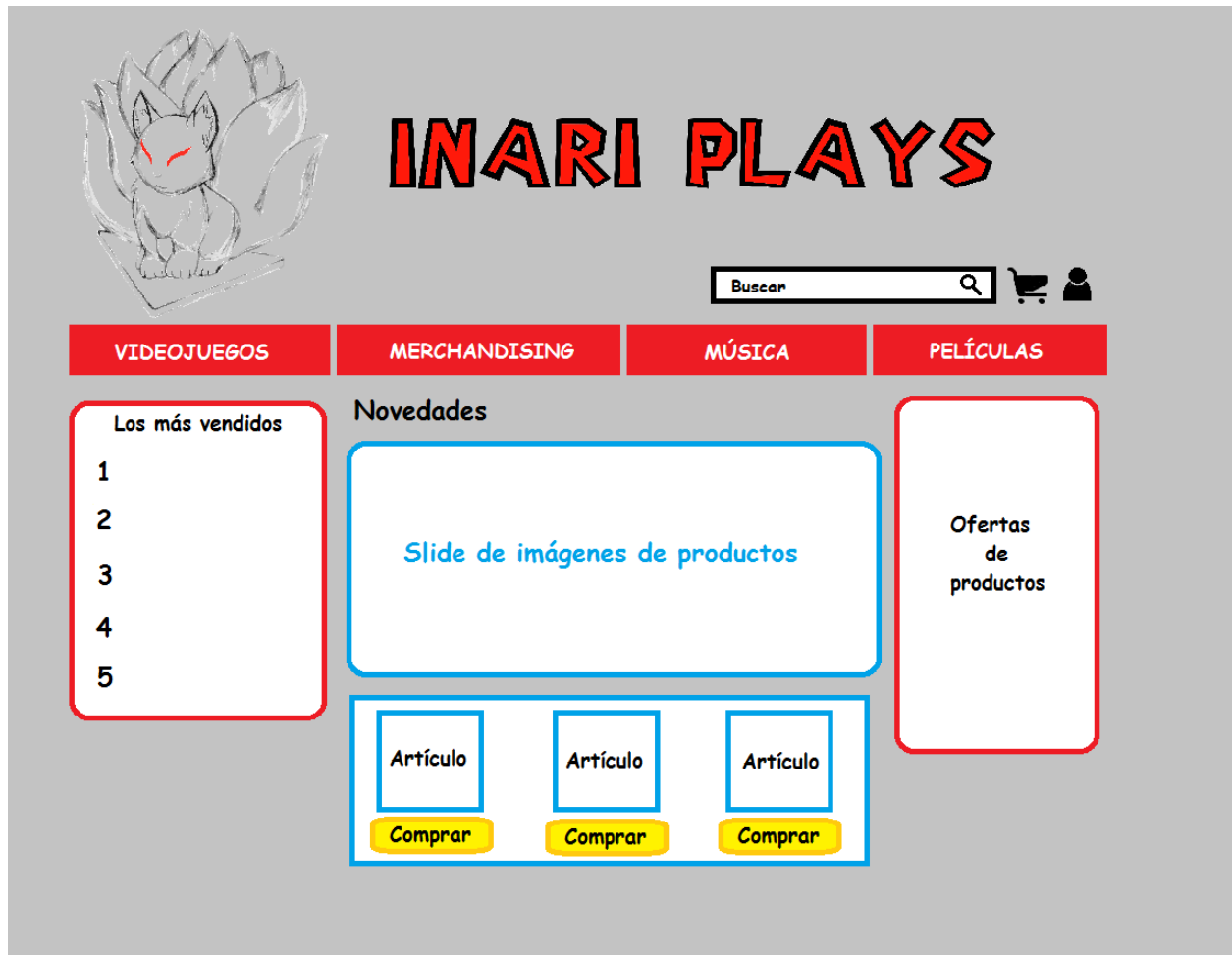
public virtual string NIF {
    get { return nIF; } set { nIF = value; }
}

public virtual string Nombre {
    get { return nombre; } set { nombre = value; }
}
```

2.2. Esquema de diseño Web


Hemos hecho, a modo de guía visual, unas imágenes en las cuales mostramos cómo queremos que se vean las principales partes de la web, de este modo será más sencillo programar el aspecto visual una vez terminado lo básico.

La página principal con la que se encontrarían los clientes al entrar sería así:






El logo no se aprecia de manera correcta debido a que esta hecho con escala de grises, pero lo que intentamos mostrar es una visión general de lo que será el diseño final.

La vista de la página de un producto se vería así:



INARI PLAYS

VIDEOJUEGOS **MERCHANDISING** **MÚSICA** **PELÍCULAS**

Imagen del producto


Nombre del producto

Descripción del producto




Número de artículos disponibles de ese producto

Cantidad :

El formulario de registro sería así:



INARI PLAYS

VIDEOJUEGOS **MERCHANDISING** **MÚSICA** **PELÍCULAS**

REGISTRASE

NOMBRE

DNI

E-MAIL

PASSWORD

DIRECCIÓN

TELÉFONO

APELLIDOS

CÓDIGO POSTAL

Debe tenerse en cuenta de que en el producto final existirán campos de tipo obligatorio, por lo que se señalarán a su debido tiempo.

Por último la cesta de productos se vería de este modo:

The mockup shows a web interface for 'INARI PLAYS'. At the top left is a sketch of a fox-like creature with red eyes. The title 'INARI PLAYS' is in large, bold, red-outlined letters. To the right is a search bar with the placeholder 'Buscar' and a magnifying glass icon, followed by a shopping cart icon and a user profile icon. Below these are four red navigation buttons: 'VIDEOJUEGOS', 'MERCHANDISING', 'MÚSICA', and 'PELÍCULAS'. The main section is titled 'CESTA' and contains three columns: 'PRODUCTOS', 'PRECIO', and 'CANTIDAD'. Each column has a blue-bordered box with descriptive text: 'La lista con las imágenes de cada producto comprado.' for products, 'El precio correspondiente a cada producto y dependiendo de la cantidad.' for price, and 'La cantidad que se quiere de cada uno de ellos.' for quantity. At the bottom, there is a 'TOTAL' label, a blue-bordered input field, and a red 'COMPRAR' button.

PRODUCTOS	PRECIO	CANTIDAD
La lista con las imágenes de cada producto comprado.	El precio correspondiente a cada producto y dependiendo de la cantidad.	La cantidad que se quiere de cada uno de ellos.

TOTAL **COMPRAR**

Puede ser que a la hora de realizar el diseño final se realicen algunos pequeños cambios ya que, como hemos dicho, este es un diseño parcial, es una idea general de lo que queremos mostrar.

Hay varias vistas que no hemos dibujado, como la página del perfil de usuario y la de administración de productos, pero que están presentes dentro del código y que se mostrarán cuando la página esté más avanzada.

3.Descripción de proyecto

Los métodos que hemos implementado para desarrollar el proyecto son:

Cliente:

- logarse: Método que nos devuelve true si coincide la contraseña pasada al método con la almacenada en la tabla cliente.
- borrarCliente: Método destructor de cliente
- modificarCliente: Método que modifica los campos de la tabla cliente
- dameClientePorID: busca un cliente por el nif (CP) → Vista administrador
- crearCliente: Constructor de cliente

Cesta:

- borrar: borra una cesta
- crear: crea una cesta

Pedido:

- anyadirLineaPedido: Relaciona el pedido con una linea de pedido
- borrarLineaPedido: borra una linea de pedido
- buscaPedidoPorID: busca un pedido por ID → Vista administrador
- dameTodosLosPedidos: devuelve todos los pedidos realizados
- borrarPedido: borra un pedido
- modificarPedido: modifica los campos de un pedido
- crearPedido: crea un pedido

EstadoPedido (Enumeracion): Pendiente, Enviado, Cancelado y Recibido

LineaPedido:

- modificaLinea: modifica los campos de una linea de pedido
- creaLineaPedido: crea una linea de pedido

Producto:

- dimeStockdeNomProducto: dado un nombre de producto devuelve el stock
- dameTodosLosProductos: devuelve todos los productos
- incrementarStock: incrementa el stock de un producto
- decrementarStock: decrementa el stock de un producto
- borrar: borra un producto
- modificar: modifica un producto

- nuevo: Introduce un nuevo producto a la base de datos

Administrador:

- logearse: Método que nos devuelve true si coincide la contraseña pasada al método con la almacenada en la tabla administrador.
- eliminarAdmin: Elimina al administrador
- modificarAdmin: Modificar al administrador.
- newAdmin: Crea un nuevo administrador

Videojuego:

- destroy: borra un videojuego
- modify: modifica un videojuego
- new: Introduce un nuevo videojuego a la base de datos

Musica

- destroy: borra un disco de musica
- modify: modifica un disco de musica
- new: Introduce un nuevo disco de musica a la base de datos

Merchandising:

- destroy: borra un objeto merchandising
- modify: modifica un objeto merchandising
- new: Introduce un nuevo objeto merchandising a la base de datos

Pelicula:

- destroy: borra una pelicula
- modify: modifica una pelicula
- new: Introduce una nueva pelicula a la base de datos

Los métodos en la base de datos se relacionan de la siguiente manera:

1 Cliente - N pedidos

1 Pedido - 1 Cliente

1 Cliente - N Cestas

1 Cesta - 1 Cliente

1 LineaPedido - 1 Pedidos

1 Pedido - N LineaPedido

1 Cesta - N Productos

1 Producto - N Cestas

1 LineaPedido - 1 Producto

1 Producto - N LineaPedido

1 Producto - 1 Videojuego

1 Videojuego - N Producto

1 Producto - 1 Musica

1 Musica - N Producto

1 Producto - 1 Merchandising

1 Merchandising - N Productos

1 Producto - 1 Pelicula

1 Pelicula - N Productos

Administrador sin relación (puede visualizar todo)