| | | |
|---|---|---|
| **Ciclo: DAM**<br>**Modulo: DESARROLLO DE INTERFACES**<br>**EXAMEN: 2º EVALUACIÓN** | | **Curso: 2024 / 2025**<br>**Grupo: DAM2A** |
| **NOMBRE:** | **José Francisco Vico López** | **Fecha: 04/03/2025** |

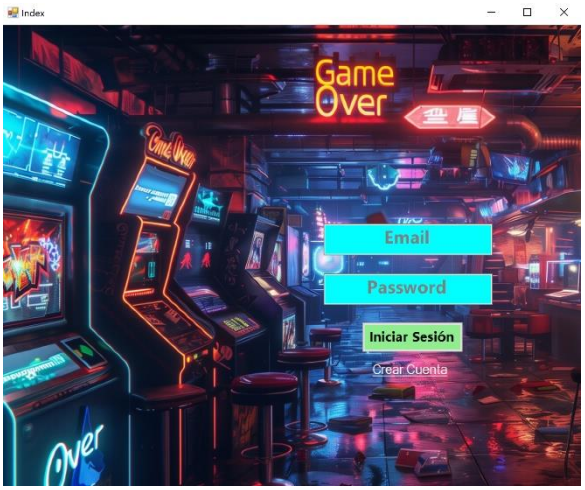
**Segunda evaluación. Catálogo de videojuegos. 10 puntos**

Desarrollo de un sistema de catálogo de videojuegos en **Windows Forms** con **.NET y MySQL**.

**Preguntas**

**Pregunta 1. (1,5 puntos)**. Confección de interfaces de usuario. Muestra una captura de pantalla de la interfaz de la HOME implementada y explica la importancia de la usabilidad en el diseño de interfaces de usuario.

La primera pantalla de la aplicación es un login. En él se muestra un formulario y una disposición muy intuitiva de los elementos lo que facilita su uso.




Login


Home de usuario

Por otro lado, una vez el usuario accede a su cuenta se abre la Home.

Para facilitar la usabilidad se ha optado por un diseño clásico en la disposición de componentes. Encontramos:

- El **navbar superior** en el que se recogen los datos del usuario junto al botón de cerrar sesión a la izquierda, un título de la página centrado y un carrito a la derecha.
- El **menú de navegación** a la izquierda de la pantalla.
- El **contenido** de la página, que ocupa la mayor parte del espacio en pantalla y en el cual se cargarán los distintos contenidos necesarios para interactuar con la aplicación. Es uso de componentes reutilizables para esta parte es crucial.

**Pregunta 2. (1,5 puntos)**. Acceso a datos en interfaces de usuario. En la aplicación con interfaz gráfica que utiliza MySQL como base de datos

a. ¿Cuáles son los pasos esenciales para establecer la conexión y recuperar los datos?

   En mi caso particular, en el desarrollo de la práctica final, he optado por separar completamente la lógica de acceso a la base de datos de la lógica de la interfaz gráfica. Para ello he implementado una API (que hará las veces de backend) como intermediaria entre la interfaz gráfica y la base de datos. La API ha sido construida siguiendo una lógica de negocio en la que se diferencian los modelos y controladores que darán acceso a la API y devolverán la información de la base de datos.

   Por ello, los pasos esenciales serían:

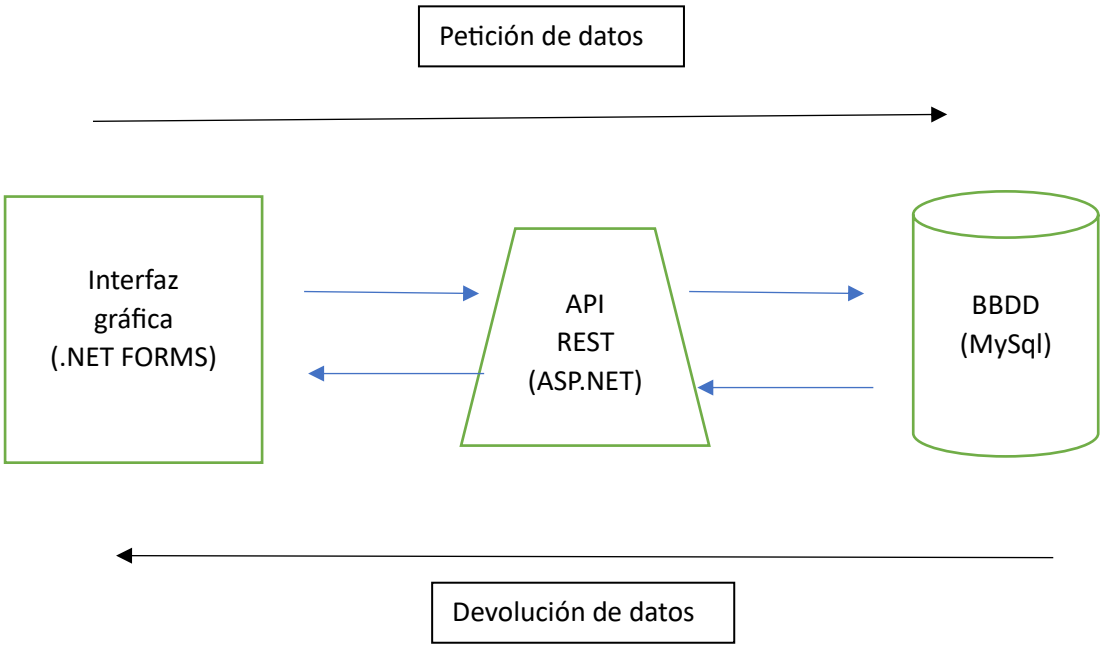
   1. Estudio de las necesidades de persistencia de datos de la aplicación previa a la creación de la BBDD.

   2. Conforme al estudio previo, elaboración de la BBDD creando las entidades pertinentes.

   3. Creación de la API de comunicación. Modelos y controladores. Pruebas de llamadas a la API y verificación de la respuesta de esta con registros de la BBDD. (Recomendado uso de Postman)

   4. Desarrollo de la interfaz gráfica e implementación de las llamadas desde esta a la API.

   Este sería el esquema de comunicación de mi apliación:



b. Muestra a continuación un ejemplo con el código de la consulta de acceso a BBDD para recuperar los datos de los usuarios que se muestran por pantalla desde la ventana de administrador

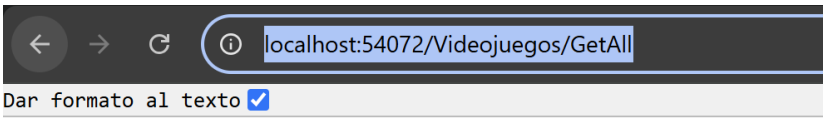
Un ejemplo de código para pedir todos los videojuegos de catálogo sería:

```
namespace CatalogoVideojuegos.API.Controllers
{
    0 referencias
    public class VideojuegosController : Controller
    {
        private string connectionString = ConfigurationManager.ConnectionStrings["catalogo_videojuegos_DB"].ConnectionString;

        // GET: Videojuegos/GetAll
        0 referencias
        public ActionResult GetAll()
        {
            using (var connection = new MySqlConnection(connectionString))
            {
                connection.Open();
                string query = "SELECT * FROM videojuegos";
                var videojuegos = connection.Query<Videojuego>(query).ToList();
                return Json(videojuegos, JsonRequestBehavior.AllowGet);
            }
        }
```

En este caso, el controlador VideojuegosController hace una petición a la BBDD cuando se hace una llamada a su método GetAll(). Si nuestra API está corriendo en localhost por el puerto 54072 (por ejemplo) la llamada a este método se realizaría de la siguiente forma:

http://localhost:54072/Videojuegos/GetAll

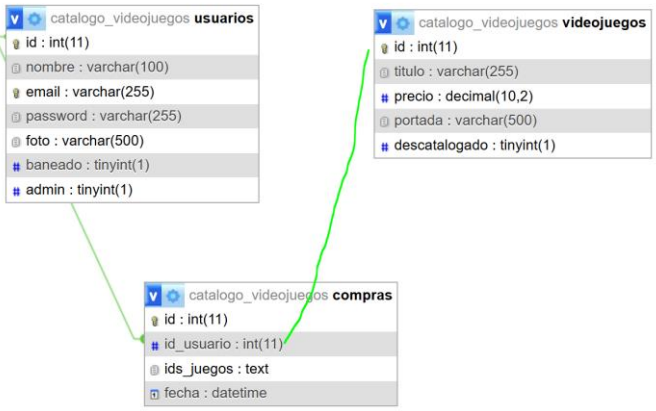




```
[
  {
    "Id": 2,
    "Titulo": "GTA6",
    "Precio": null,
    "Portada": "1343294.jpeg",
    "Descatalogado": false
  },
  {
    "Id": 3,
    "Titulo": "GT",
    "Precio": null,
    "Portada": "1370435.png",
    "Descatalogado": true
  },
  {
    "Id": 4,
    "Titulo": "HOLA",
    "Precio": null,
    "Portada": "default.png",
    "Descatalogado": false
  },
  {
    "Id": 5,
    "Titulo": "HOLA",
    "Precio": null,
    "Portada": "1343294.jpeg",
    "Descatalogado": false
```

c. Representa el modelo relacional utilizado en la práctica

Este sería mi modelo en el que se relacionan las entidades “Usuario”, “Videojuego” y “Compra” (que recoge la relación entre “Usuario” y “Videojuego”




**Pregunta 3. (1,5 puntos)**. Explica los componentes visuales utilizados en el desarrollo de la práctica y menciona tres ventajas de crear componentes reutilizables en una aplicación.

En el desarrollo de la aplicación fue crucial la utilización de componentes reutilizables debido a que en varias partes de la aplicación iban a ser necesarios.

Componente como la Home con un panel de contenido donde se sirve el Catálogo de Videojuegos general o el Catálogo de Videojuegos propio del usuario. El componente Videojuego se pinta tantas veces como registros de videojuegos haya en la base de datos, por ello es necesario que sea común.

Otra ventaja del uso de componentes comunes es la **mejor organización** del código, pudiendo recogerlo en una carpeta común. Esto se define como refactorización del código, crucial para que nuestro **desarrollo sea escalable** y se pueda mantener.

Gracias a esto último, también se mejora la **fluidez de la aplicación**, ya que no necesita compilar y renderizar un gran cantidad de componentes.

**Pregunta 4. (1,5 puntos)**. Confección de informes

a. ¿Cuál es la importancia de la confección de informes en el desarrollo de software?
   Es esencial obtener informes para analizar y mantener la información de las entidades que interactúan con la aplicación. Por ejemplo: usuarios que se han dado de alta en la aplicación, ingresos obtenidos en una determinada campaña, usuarios baneados por mal uso de la aplicación, videojuegos más o menos vendidos, momentos en los que se realizan más compras, etc.
b. Si tuvieras que elaborar un informe sobre el catálogo de videojuegos. ¿Qué datos mostrarías?
   Título del juego
   Género
   Fecha de lanzamiento
   Precio
   Ventas
c. Muestra un ejemplo a continuación de un posible informe o cuadro de mando

| Titulo | Género | Fecha de lanzamiento | Precio (€) | Ventas |
|---|---|---|---|---|
| The Last of Us | Acción | 2013 | 39,99 | 20 |
| FIFA 10 | Deportes | 2009 | 49,99 | 30 |
| Minecraft | Sandbox | 2012 | 19,99 | 50 |

**Pregunta 5. (1,5 puntos)**. Documentación y pruebas de aplicaciones

a. ¿Por qué es importante la documentación en el desarrollo de software?

   Porque ayudan a planear el desarrollo desde etapas tempranas hasta su despliegue, pasando por desarrollo de base de datos, elaboración de pruebas, etc. Además de que son necesarios cuando varias personas trabajan en el mismo desarrollo, para organizar equipos y competencias.

b. ¿Por qué son fundamentales las pruebas en el desarrollo de aplicaciones? Explica tres beneficios que aportan a la calidad del software.

   **Detección de errores antes de que ocurran**

   **Organización de equipos** para las distintas competencias

   **Garantizar el correcto funcionamiento** de las distintas partes de la aplicación

   Permite **ajustar los tiempo** de entrega del proyecto.

c. Diseña un plan de pruebas para tu aplicación

Plan de pruebas.xlsx

d. Adjunta las evidencias de la ejecución del plan de pruebas


Evidencias del plan de pruebas.pdf

(da un error al abrir pero se puede visualizar el contenido. Por algún motivo no se visualiza la portada del documento)

**Pregunta 6. (1,5 puntos)**. Distribución de aplicaciones

a. Explica brevemente cuales son los factores claves para una distribución exitosa de la aplicación

   Los factores clave para una distribución exitosa de una aplicación son:

- **Compatibilidad:** Asegurarse de que la aplicación funcione correctamente en los sistemas operativos, dispositivos y versiones para los que fue diseñada.
- **Pruebas previas:** Realizar pruebas exhaustivas para detectar y corregir errores antes del lanzamiento, garantizando estabilidad y buen rendimiento.
- **Documentación:** Contar con manuales de usuario, guías de instalación y soporte técnico para facilitar el uso y mantenimiento.
- **Canales de distribución adecuados:** Elegir plataformas seguras y accesibles para que los usuarios puedan descargar o acceder fácilmente a la aplicación (por ejemplo, App Store, Google Play, página web oficial).
- **Actualizaciones y soporte:** Ofrecer mantenimiento continuo, actualizaciones y atención a los usuarios después del lanzamiento para corregir errores y mejorar funciones.
- **Seguridad:** Proteger los datos del usuario y evitar vulnerabilidades durante y después de la instalación.

b. Adjunta a continuación el enlace de github con el despliegue de la aplicación implementada.

   https://github.com/franvico/catalogo_videojuegos

**Pregunta 7 (1 punto)**. ¿Vas a presentar el proyecto?

[X] Sí  No [ ]