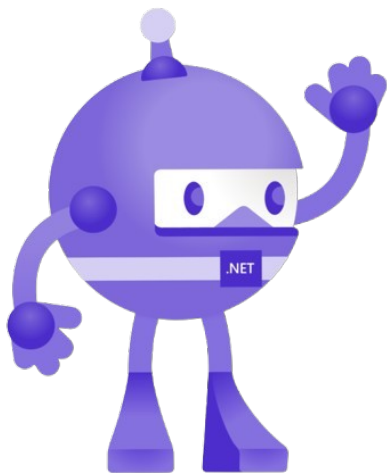


# DESARROLLO DE INTERFACES

Proyecto final  
PowerStudioCode



Izan López y Marcos Z.

## Índice

<b>Introducción.....</b>	<b>3</b>
<b>Tiempo utilizado.....</b>	<b>4</b>
<b>Material utilizado.....</b>	<b>5</b>
Ordenador de casa de Marcos Z.....	5
Ordenador de casa de Izan López.....	5
Portátiles de clase.....	5
<b>Objetivos de la práctica.....</b>	<b>6</b>
<b>Pre-desarrollo del Proyecto.....</b>	<b>7</b>
Teoría del color.....	7
<b>Desarrollo del Proyecto.....</b>	<b>9</b>
Idea inicial.....	9
Bocetos.....	9
Boceto Perfil.....	9
Boceto página principal provisional.....	10
Boceto página principal definitivo.....	10
Base de datos utilizada.....	11
Aspecto gráfico de la base de datos.....	11
Contenido utilizado del curso.....	12
Pantallas de la aplicación.....	14
RegisterPage.....	15
LoginPage.....	16
Página principal.....	17
Perfil.....	18
Gestión de usuarios.....	18
Menú lateral.....	19
Contenido a futuro.....	20
Tecnologías utilizadas.....	21
GitHub – Control de versiones y gestión de tareas.....	22
Multimedia utilizado.....	23
<b>Conclusión.....</b>	<b>24</b>
<b>Problemas y/o sugerencias en la actividad.....</b>	<b>25</b>
Problema 1# - ¿Qué base de datos usar?.....	25
Problema 2# - Muchas ideas poco tiempo.....	25
<b>Webgrafía.....</b>	<b>26</b>

## Introducción

En esta actividad veremos una gestión de tareas como la reciente actividad pero con la particularidad que en este caso utilizaremos ViewModels para todo el tema de gestión de información y todo lo que tenga que ver con las páginas, también haremos uso de los ICommands para gestionar varios botones y darles un uso funcional.

También hacemos uso de conocimientos anteriores para volverlos a poner uso como por ejemplo el Shell, para navegar por páginas gracias a él y también el uso de Triggers, para cambiar su aspecto al estar completada la tarea o no, esto nos da más margen de decoración y varias posibles opciones a realizar.

## Tiempo utilizado

Hemos utilizado muchas horas y mucho esfuerzo a este proyecto, hasta tal punto que sería muy difícil decir unas horas específicas pero tal vez entre los dos habremos usado 15-20 horas de nuestro tiempo y para realizar el documento y presentación del proyecto habrán sido 3 horas, también hay que contar los días planificando todo el proyecto, bocetos, etc. que fueron 2-3 clases que igual se traducen a 4 horas, en total aproximadamente **22-27 horas**.

## Material utilizado

- El portatil de clase de ambos y el ordenador
- Visual Studio.

### Ordenador de casa de Marcos Z.

Componente	Nombre
Sistema Operativo	Windows 11 Pro
Procesador	AMD Ryzen 5 2600X
Tarjeta gráfica	NVIDIA GeForce GTX 1650 4GB
Memoria RAM	16 GB
Placa base	B450M-A PRO MAX
Disco duro 1#	KIOXIA EXCERIA G2 SSD 1 TB
Disco duro 2#	ST1000DM010-2EP102 HDD 1 TB

### Ordenador de casa de Izan López

Componente	Nombre
Sistema Operativo	Windows 10
Procesador	12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
Tarjeta gráfica	NVIDIA GeForce 3060RTX
Memoria RAM	16 GB
Placa base	
Disco duro 1#	500 GB SSD
Disco duro 2#	2TB HDD

### Portátiles de clase

Componente	Nombre
Sistema Operativo	Windows 10 Pro
Procesador	Intel(R) Core(TM) i5-7300U @ 2.60GHz 2.71 GHz
Tarjeta gráfica	(Integrada)
Modelo portátil	HP EliteBook 840 G5
Memoria RAM	16 GB
Placa base	HP 83B2
Disco duro	Disco duro HDD 256GB

## Objetivos de la práctica

- Realizar una interfaz con ICommands en MAUI.
- Entender el funcionamiento de como usarlos ICommand.
- Entender el funcionamiento de los Binding y utilizarlos.
- Diseñar una interfaz interesante de ver.
- Ser capaz de traspasar información entre una página y otra.

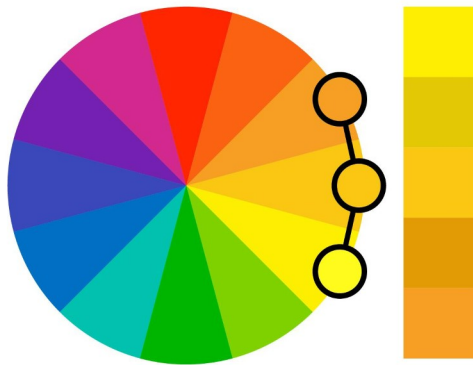
## Pre-desarrollo del Proyecto

### Teoría del color

Hemos utilizado la documentación que nos entregaste para basarnos en una forma de encontrar los colores adecuados a nuestra aplicación, principalmente en esta:

> <https://www.significados.com/teoria-del-color/>

A partir de esa página web hemos sacado que combinación nos gustaba y para basarnos en alguno seguro hemos utilizado la **análoga**, que contiene positivas como que se complementan bien al ser colores cercanos.



Después de elegir el tipo de combinación que íbamos a utilizar tocaba elegir el color, en este caso utilice la página web que nos creaba el propio círculo y nos daba los código HEX utilizando el método que queramos y es la web de **Figma**, que también nos dejaste en el foro.

> <https://www.figma.com/es-es/circulo-cromatico/>

Seleccione el tipo de paleta análoga y de ahí fui probando los colores viendo que tal quedaban, en su momento elegí el azul y quise ver la paleta de colores que nos podía mostrar y le di a “**Get color palette**”, al darle nos genera una imagen con varias combinaciones posibles, que ahora mostraré.

Me dio **4 paletas de colores** distintas:

- > Paleta de colores azules
- > Paleta de colores azules más oscuros
- > Paleta de colores cyan
- > Paleta de colores grises

Al final decidimos que la primera de todas era la más bonita visualmente y quedaría bien ([enlace de la paleta](#)).



Es decir, paleta de colores definitiva es la siguiente:

> [#E7F2FF](#) - [#B0D7FF](#) - [#3DB4FF](#) - [#008DCF](#) - [#00689A](#) - [#004467](#) - [#002439](#) <





# Desarrollo del Proyecto

## Idea inicial

Nuestra idea inicial fue hacer una empresa que creaba videojuegos y luego los vendía en su propia tienda, nosotros íbamos a diseñar esa tienda online que era una aplicación desde la cual podrías comprar los juegos, jugarlos y los trabajadores de la empresa podían gestionar sus tareas desde otra pantalla especial para ellos, sobre lo último se tuvo que cancelar por falta de tiempo y experiencia pero viendo el resultado final no hemos ido tan mal encaminados de nuestra imaginación.

## Bocetos

Para hacer las distintas pantallas que requería la aplicación hicimos bocetos de cada pantalla, con esto conseguíamos plasmar nuestra idea y opinar sobre que podría ser mejor o no, tras dialogar nuestras opiniones intentábamos combinarlas y de ahí salía un resultado que a ambos nos gustaba.

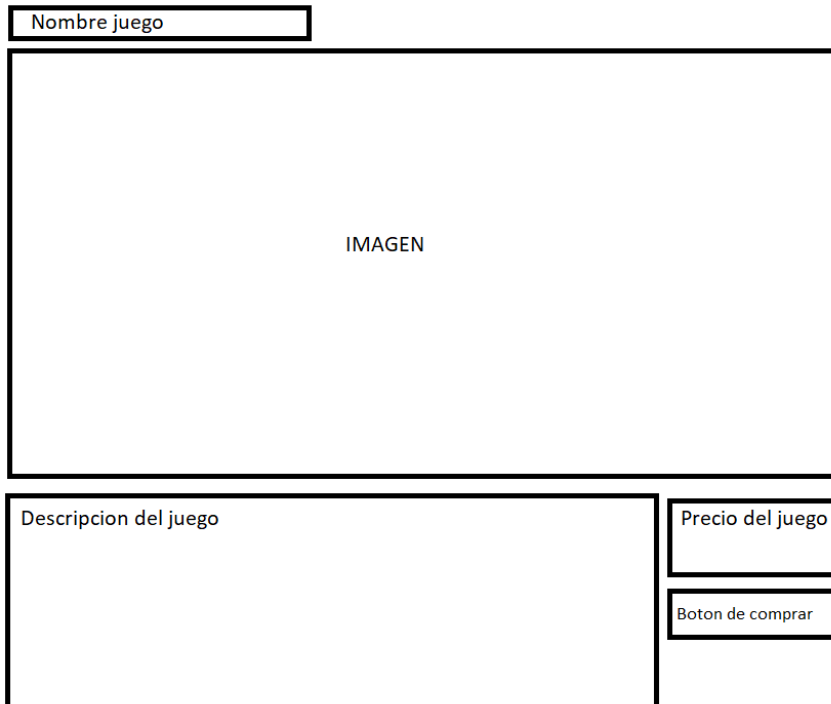
## Boceto Perfil

En este boceto buscábamos algo con bastantes cosas, las cuales igual cambiaron después pero ese era nuestro objetivo.

	<div>Foto perfil</div>	<div>Nombre de usuario</div>	<div>Sueldo actual €</div>	
		<div>Descripción</div>	<div>Cargar sueldo</div>	
	<div>Estadísticas</div> <div>(Cantidad de juegos   Horas jugadas )</div>			
	<div>LISTA DE JUEGOS</div>			

## Boceto página principal provisional

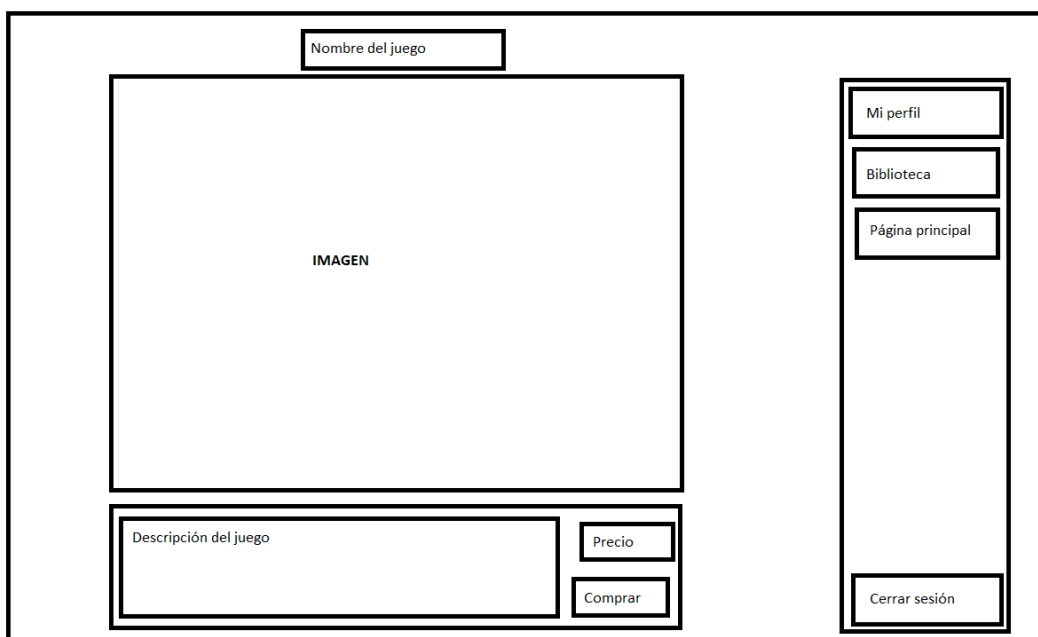
Este boceto fue muy sencillo ya que no teníamos mucha idea de como íbamos a terminar uniendo todo el proyecto y esto era algo provisional.



A provisional layout sketch for a game page. It consists of a header box labeled "Nombre juego". Below it is a large rectangular area labeled "IMAGEN". To the left of the image area is a box labeled "Descripcion del juego". To the right of the image area are two stacked boxes: "Precio del juego" on top and "Boton de comprar" on the bottom.

## Boceto página principal definitivo

Aquí se nota la incorporación del menú lateral, y acaba siendo más profesional.



A definitive layout sketch for a game page, featuring a sidebar menu. The main content area contains a header box "Nombre del juego", a large "IMAGEN" box, a "Descripción del juego" box, and a "Precio" box with a "Comprar" button below it. The sidebar on the right contains a menu with "Mi perfil", "Biblioteca", and "Página principal" buttons, and a "Cerrar sesión" button at the bottom.

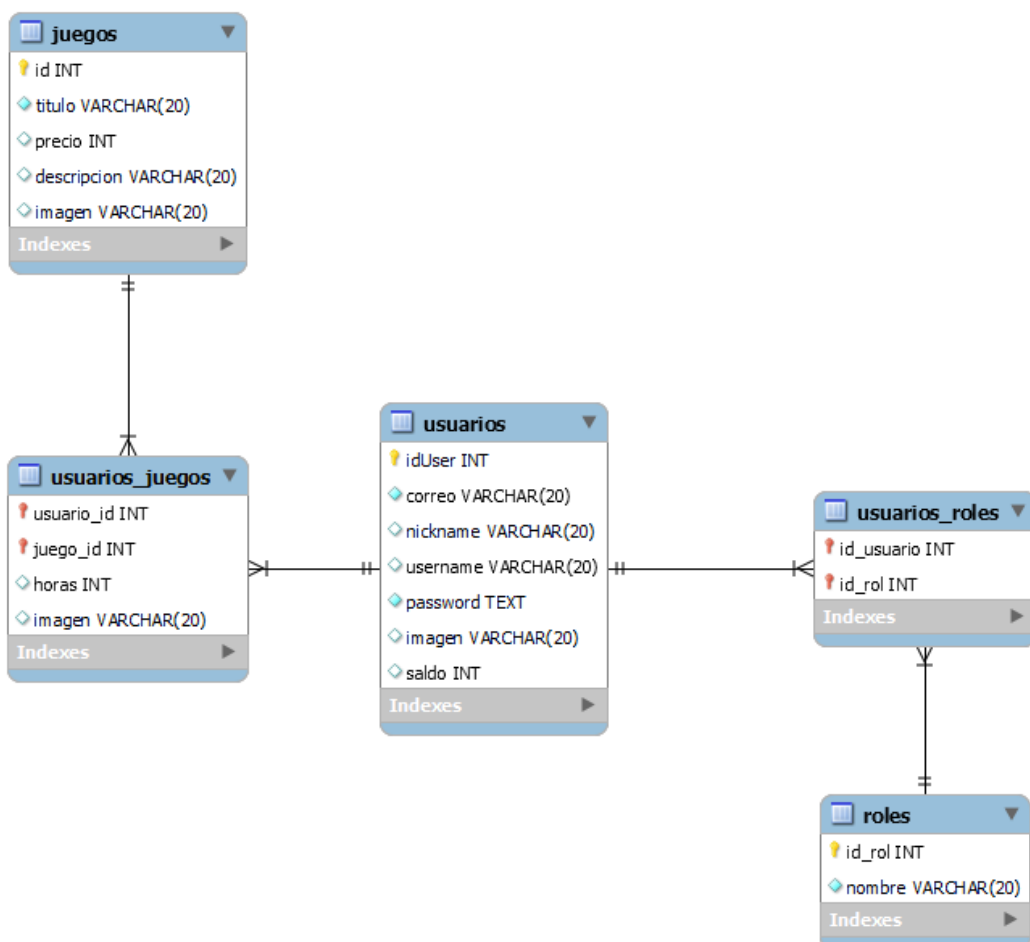
## Base de datos utilizada

La base de datos que hemos utilizado al final ha sido SQLite, por su facilidad para realizar todo tipo de tarea ya que también es muy fácil de manejar, hemos establecido todo tipo de método en la cuál tenia algo que ver con la base de datos en una clase llamada **MgtDatabase.cs** para gestionar todo tipo de consulta, añadir usuario, iniciar sesion, registrarse, etc...

## Aspecto gráfico de la base de datos

Esta es la ingeniería inversa hecha desde MySQL de nuestra base de datos:

- **Usuarios** esta relacionado con **usuarios\_juegos** y **usuarios\_rols** para ver los juegos que le pertenecen y luego en **usuarios\_rols** se puede ver que tipo de usuario es.



## Contenido utilizado del curso

### Bindings con su variable pública

Todo tipo de variable que estaba en la interfaz podíamos acceder desde ella con Bindings.

### ICommand

Cuando presionabas un botón en la aplicación este depende de un ICommand para ejecutar su función, así lo hacíamos de una forma ordenada con nombres visibles y todo más visual.

### Shell (Rutas)

El tema de la navegación en nuestra aplicación lo hemos hecho con el:

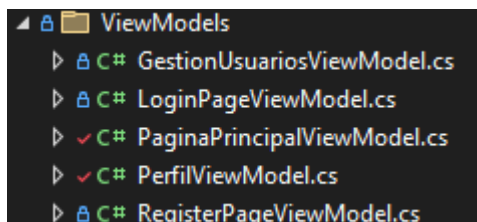
```
await Shell.Current.GoToAsync($"{pagina}");
```

Registrando previamente las rutas que queríamos conectar:

```
// Rutas de navegación
Routing.RegisterRoute(nameof(MainPage), typeof(LoginSystemPowerCode.MainPage));
Routing.RegisterRoute(nameof(RegisterPage),
typeof(LoginSystemPowerCode.Pages.RegisterPage));
Routing.RegisterRoute(nameof(Perfil), typeof(LoginSystemPowerCode.Pages.Perfil));
Routing.RegisterRoute(nameof(LoginPage),
typeof(LoginSystemPowerCode.Pages.LoginPage));
Routing.RegisterRoute(nameof(PaginaPrincipal),
typeof(LoginSystemPowerCode.Pages.PaginaPrincipal));
Routing.RegisterRoute(nameof(GestionUsuarios),
typeof(LoginSystemPowerCode.Pages.GestionUsuarios));
```

### ViewModels

La aplicación esta compuesta de ViewModels, con su correspondiente carpeta para mayor organización:



## DisplayPromptAsync

Para cambiar de usuario/contraseña/nickname hemos utilizado el DisplayPromptAsync para hacer que sea todo más fluido y realista.

```
string nuevoNickname = await _page.DisplayPromptAsync("Editar Nickname", "Introduce tu nuevo nickname:", "OK", "Cancelar", "Nuevo nickname", maxLength: 20, keyboard: Keyboard.Text);
```

## La interfaz OnPropertyChanged()

Para gestionar los cambios de propiedades en mitad de la aplicación hemos utilizado en varias ocasiones la interfaz `INotifyPropertyChanged` que nos permitía notificar los cambios y cambiar los datos a tiempo real.

## QueryProperty

En nuestra aplicación esto tiene un trabajo bastante importante ya que así es como pasa nuestro usuario mediante pantallas, en este caso ya que no podemos pasar objetos pasamos la ID del usuario de la base de datos, este al pasar de nuevo a una pantalla al asignar el valor de la ID a la nueva pantalla llama al método `public Usuario ObtenerUsuarioPorId(int idUser)` que está en la clase `MgtDatabase.cs` y nos devuelve el Usuario, para gestionarlo.

## SQLite

Nuestra base de datos, que trabajemos en una actividad y en la cuál hemos decidido trabajar esta vez por su alta facilidad de gestionar los datos, también hay que tener en cuenta nuestra experiencia en SQL, nos ha sido más sencillo que con Firebase.

## Pantallas de la aplicación

La aplicación consta de 7 pantallas con contenido y cada pantalla tiene su ViewModel y 4 models:

### **Páginas:**

- RegisterPage
- LoginPage
- PaginaPrincipal
- Perfil
- ImageSelectionPopup
- GestionUsuarios
- Cartera

### **ViewModels:**

- GestionUsuariosViewModel
- LoginPageViewModel
- PaginaPrincipalViewModel
- PerfilViewModel
- RegisterPageViewModel

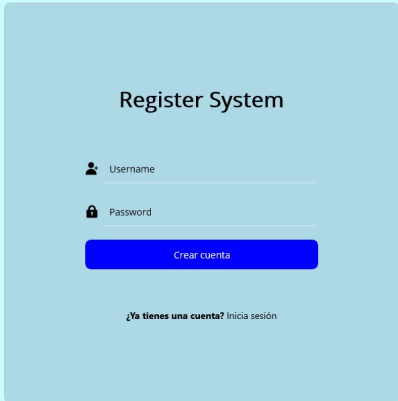
### **Models:**

- Usuario
- Juego
- ImageModel
- MgtDatabase

## RegisterPage

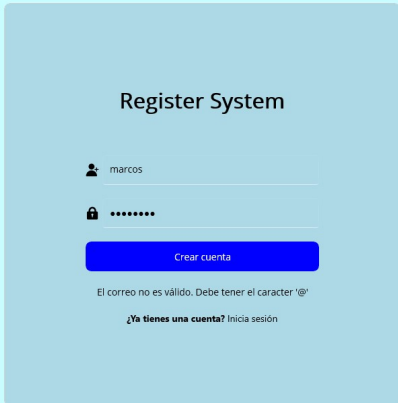
Esta es la pantalla con la que te registras en la base de datos, tiene 2 comprobaciones para evitar errores, una es que tu correo debe tener un "@" y otra es que debe ser mayor o igual de 6 caracteres.

Si ya tienes una cuenta solamente debes darle al label de abajo y te llevara a la pantalla de **Log in**.



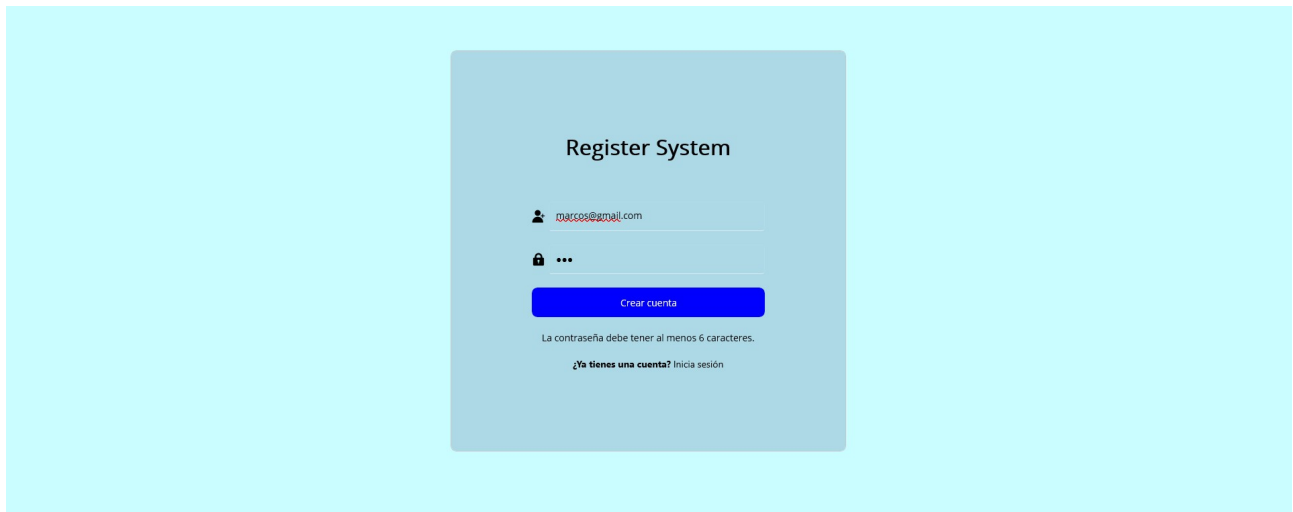
The image shows a 'Register System' form on a light blue background. The form is a darker blue rectangle with the title 'Register System' at the top. Below the title are two input fields: 'Username' with a person icon and 'Password' with a lock icon. A blue button labeled 'Crear cuenta' is positioned below the password field. At the bottom of the form, there is a link that says '¿Ya tienes una cuenta? Inicia sesión'.

### Caso práctico 1# - No tiene @ puesta



The image shows the same 'Register System' form as before, but with the 'Username' field filled with 'marcos' and the 'Password' field filled with seven dots. A blue button labeled 'Crear cuenta' is still present. Below the button, an error message is displayed: 'El correo no es válido. Debe tener el caracter '@''. At the bottom, the link '¿Ya tienes una cuenta? Inicia sesión' is visible.

## Caso práctico 2# - No tiene la longitud suficiente

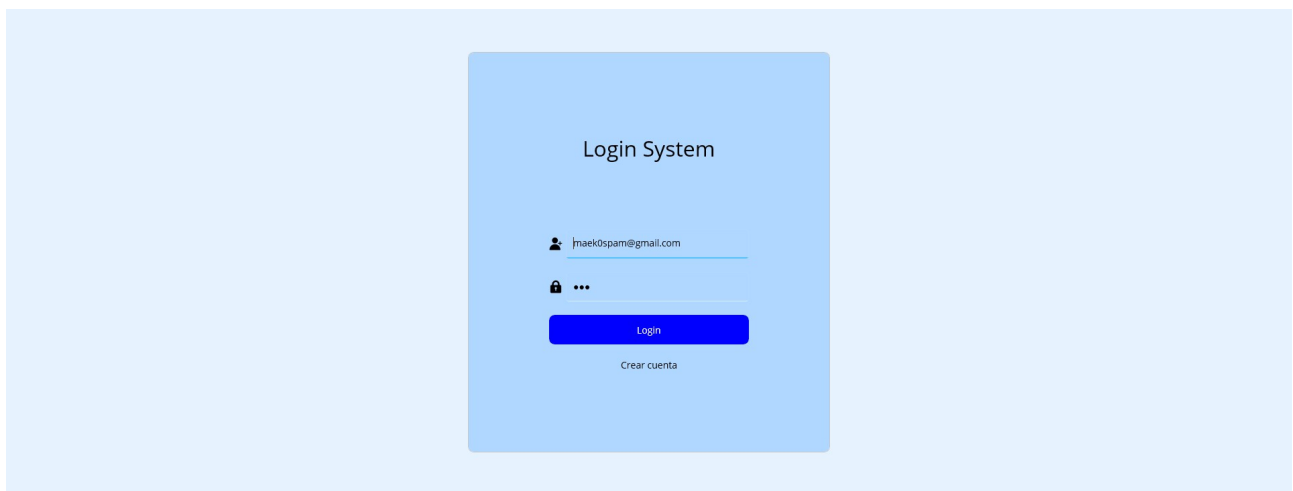


## LoginPage

Esta pantalla sirve para iniciar sesión con las credenciales de tu cuenta, las cuales deben estar en la base de datos, sino no funcionará.

En este caso sale lo siguiente, en este momento esta por defecto para evitar largo tiempo al hacer pruebas y así que todo sea más fluído.

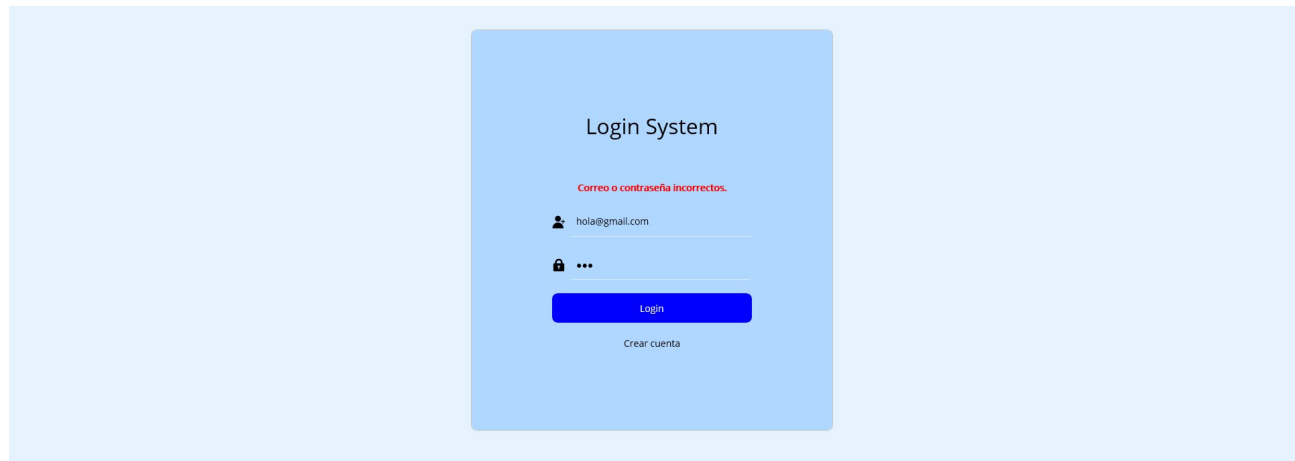
Por supuesto las credenciales deben ser correctas o estar en la base de datos sino dará error como vamos a ver a continuación.





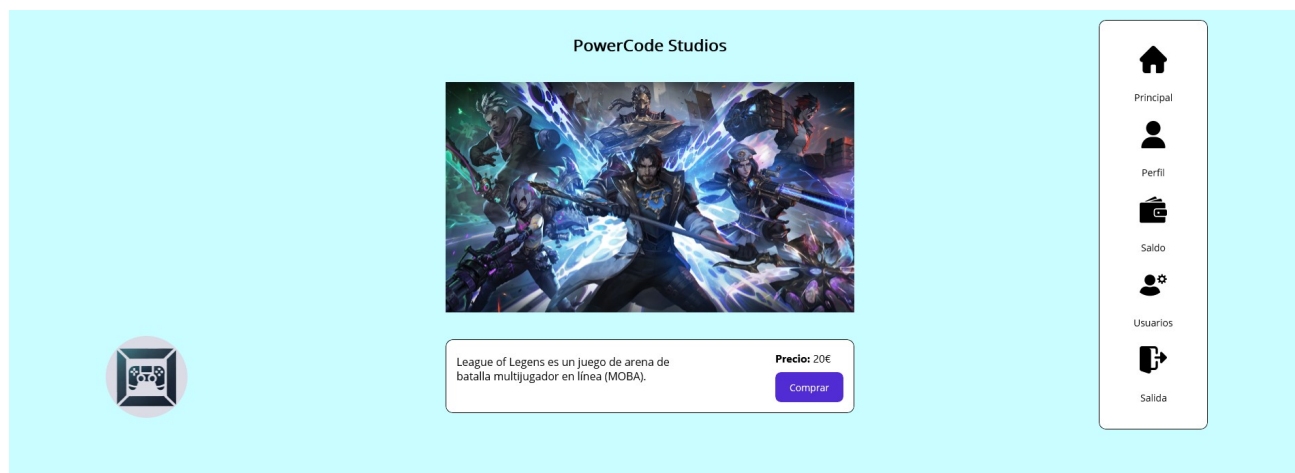
## Caso práctico 1# - Error de contraseña/correo

En este caso este correo no existe en la base de datos y para mantener más seguridad y privacidad al usuario no se dice si existe el correo, para evitar las pruebas constantes de contraseñas mediante algún programa externo.



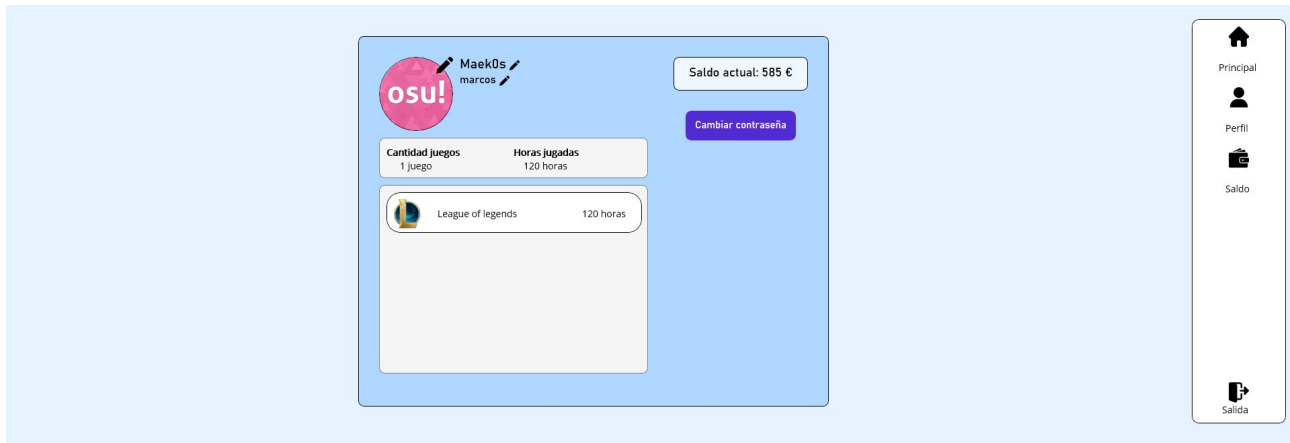
## Página principal

En esta página veremos los juegos disponibles de la aplicación. El juego cambia constantemente para mostrar los diferentes juegos.



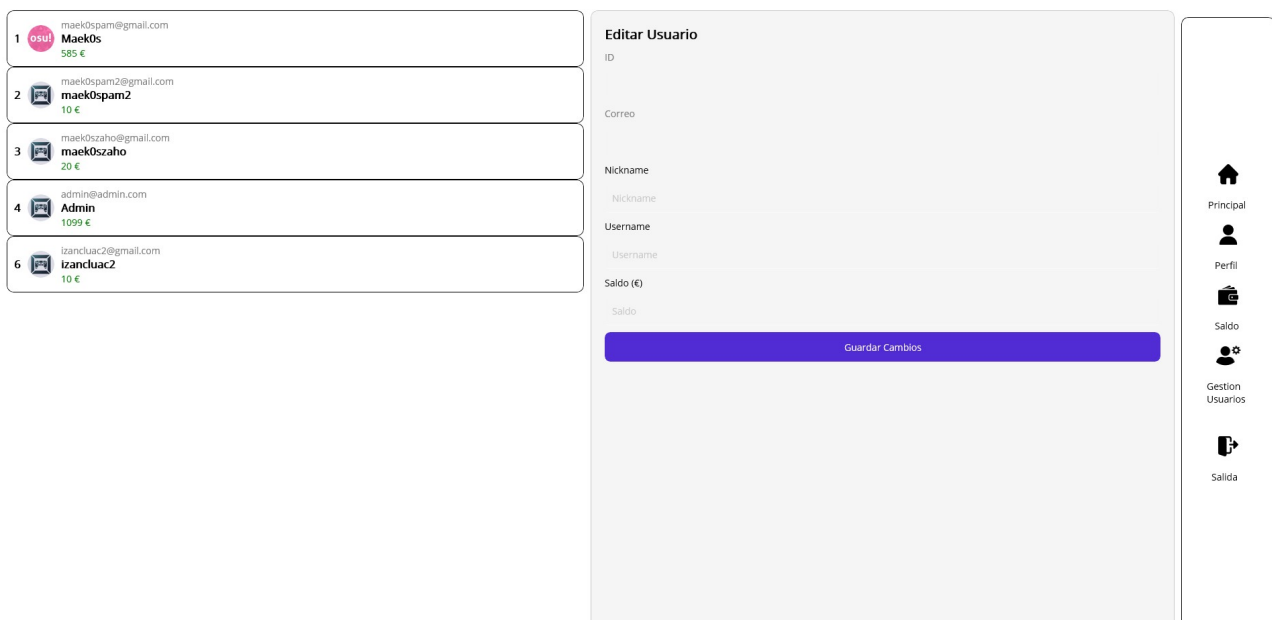
## Perfil

En el perfil podrás modificar tu nombre, nickname, foto de perfil y tu contraseña, también podrás consultar los juegos que tienes actualmente y ver el tiempo que has jugado.



## Gestión de usuarios

Desde aquí solo puede acceder un administrador, y esta opción solo será visible para los administradores desde el menú lateral, y es una pantalla para gestionar toda la base de datos viendo toda la información de todos los usuarios a tiempo real.



## Menú lateral

Con este menú que hemos ido viendo puedes moverte por la aplicación sin problema.

La opción de usuarios, que es la anterior pantalla como dije solo estará habilitada a usuarios que sean administradores, a los demás directamente no les saldrá esta opción.



## **Contenido a futuro**

Este apartado está más simple en el Problema 2# del documento, aquí vamos punto por punto explicando un poco más todo.

### **Añadir pantallas a cada juego**

Esto es algo muy típico y necesario en una aplicación para comprar videojuegos, cada juego tiene su pantalla individual, con su valoración, mucha más descripción y más detalles que en una pantalla principal no aparecen, esto da una vista más extensa al usuario de que es lo que podría estar comprando, lo cual es muy bueno.

### **Lista de amigos**

Muchas veces la razón o el motivo para probar/jugar videojuegos son nuestros amigos, en una aplicación así sería muy útil tener una lista de amigos con la que podrías hablar.

### **Hacer pantalla de biblioteca**

Al comprar un juego normalmente suele estar en una biblioteca, esto hace que el usuario pueda ver sus juegos más ordenados y la experiencia mejora mucho, ya que los juegos que ofrecemos no son jugables no era un punto tan necesario pero podría haber sido un buen detalle.

### **Compatibilidad con distintas monedas**

Con este tipo de aplicaciones no se busca ser solo de un tipo de moneda específica, debe ser una aplicación global, es por eso que estaría bien permitir ver los precios y cargar el sueldo dependiendo de la moneda que tengas, esto permite que la aplicación sea global.

### **Hacer documentación con Docfx**

Por lo que hemos investigado era una forma optima de documentar todo el código y es algo necesario para darle mantenimiento y mejorarlo día a día sin perder tiempo en leerlo de nuevo, lo malo es que esto conlleva mucho tiempo para que sea una buena documentación.

## Tecnologías utilizadas

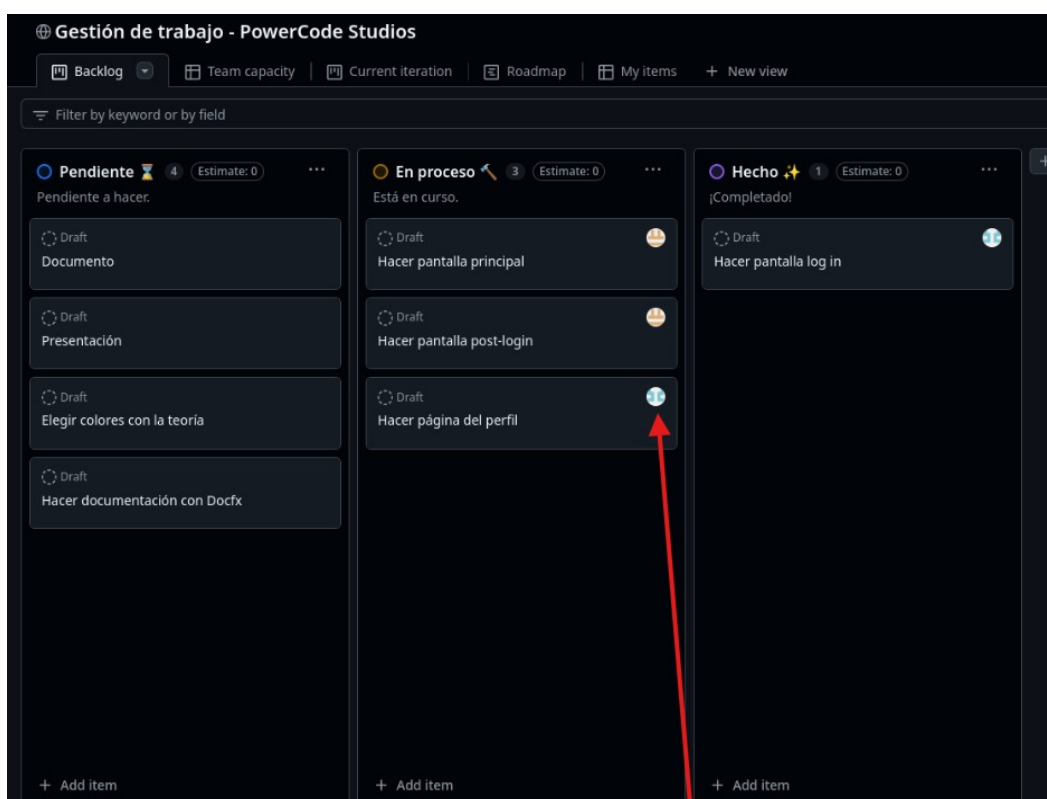
Para nuestro proyecto hemos utilizado una serie de tecnologías para hacer nuestro trabajo más efectivo y más eficiente en un futuro, como por ejemplo el uso de **GitHub**, nos ha servido para llevar un control de versiones ordenado para evitar problemas de posibles pérdidas de datos.

**GitHub** también nos ha ayudado a llevar todo el tema de la gestión de tareas, como podrás ver después hemos organizado las tareas de cada uno mediante GitHub Projects para ordenarnos y ser más efectivos.

## GitHub – Control de versiones y gestión de tareas

GitHub ha cambiado nuestra forma de hacer un proyecto, como he comentado anteriormente el control de versiones ha sido muy efectivo para evitar perdidas de archivos, y a su vez hemos descubierto un apartado bastante interesante para gestionar nuestro trabajo, el saber que debe hacer cada uno, esto lo hemos gestionado gracias a los **GitHub Projects** que nos permitía marcar unas tareas y elegir quien debe hacerlas.

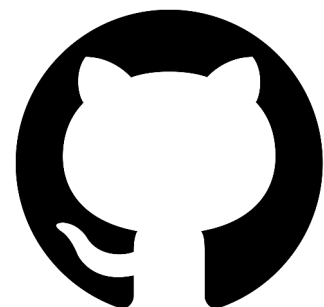
Aquí se puede ver como hay **3 columnas**, divididas entre **“Pendiente”**, **“En proceso”** y **“Hecho”**:



Como se puede ver **existen algunas tareas que tienen un icono**, estos iconos representan al usuario/trabajador que es encargado de esa tarea.

Enlace: [Maek0s/2DAM ProyectoDI](#)

Proyecto: <https://github.com/users/Maek0s/projects/2>



## Multimedia utilizado

Hemos sacado los iconos de la barra lateral de la siguiente web:

<https://www.flaticon.com/>

El logo lo generó la **IA de Designer** de Microsoft.

## Vídeo del funcionamiento

Enlace: <https://youtu.be/CNMiXJ1qw5M>

## Conclusión

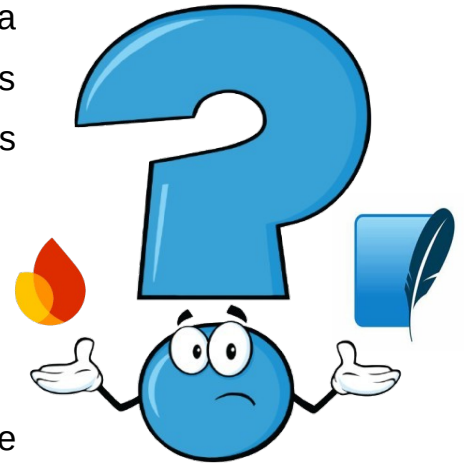
En conclusión, ha sido un proyecto entretenido de hacer, con muchos puntos a mejorar en el aspecto de planificación y gestión de tiempo, ya que nos costaba un poco compaginar esto con los trabajos constantes y más cosas, ambos hemos conseguido mucha experiencia de este proyecto y hemos hecho un gran trabajo individual como también en pareja, distribuyendo las tareas correctamente y teniendo una comunicación fluida y constructiva, recibiendo los comentarios del compañero con tal de mejorar la interfaz/aplicación.



## Problemas y/o sugerencias en la actividad

### Problema 1# - ¿Qué base de datos usar?

Al principio teníamos esto bastante más claro, y era usar **Firestore**, nos parecía complicada la idea pero posible, después cuando fuimos a intentar almacenar información a parte de los usuarios mediante Firestore Authentication fue un caos, y se nos ocurrió la idea de hacer 2 bases de datos, **Firestore** para usuarios y **SQLite** para almacenar la información de esos usuarios, y de alguna forma vincular ambas bases de datos, al intentarlo fue un dolor de cabeza increíble, y se terminó decidiendo por eliminar **Firestore** y quedarse con **SQLite**, esto nos ahorró mucho tiempo pero perdimos también bastante ya que la estructura ya estaba hecha.



### Problema 2# - Muchas ideas poco tiempo

Al principio hicimos muchas ideas posibles, muchos roles posibles, muchas pantallas pero al fin al cabo nos dimos cuenta que no íbamos a tener tiempo de todo y tuvimos que crear una sección de “A futuro” donde quedaron ideas como:

- > Añadir pantallas a cada juego
- > Lista de amigos
- > Pantalla de biblioteca (de juegos comprados)
- > Compatibilidad con otras monedas
- > Documentar el código con Docfx



## Webgrafía

- <https://visualstudio.microsoft.com/es/vs/community/>
- <https://dotnet.microsoft.com/es-es/apps/maui>

- **CollectionsView:**

<https://learn.microsoft.com/es-es/dotnet/maui/user-interface/controls/collectionview/>

- **ICommand:**

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/data-binding/commanding?view=net-maui-9.0>

- **ViewModels:**

<https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>

- **Vincular SQLite con MAUI:**

<https://learn.microsoft.com/en-us/dotnet/maui/data-cloud/database-sqlite?view=net-maui-9.0>