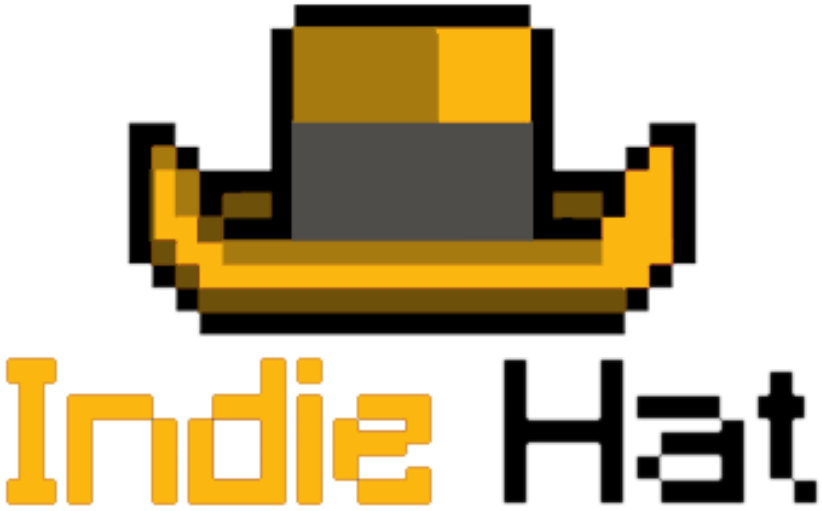
Proyecto DAW



Oscar Fontan González

72147238X

Desarrollo de Aplicaciones WEB(DAW)

7 de Junio de 2019

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

El proyecto (Indie Hat) se basa en una plataforma web, orientada a la visualización y fácil acceso a los proyectos (videojuegos) de estudios independientes.

Dada la creciente actividad en esta industria, se está produciendo una marcada proliferación de los estudios pequeños (que dependen de su propia financiación), lo que supone una cantidad enorme de lanzamientos al mes, y ninguna plataforma marcada para su visualización.

En la plataforma a desarrollar, se pretende dar cabida a todos estos proyectos, facilitando su visualización, proporcionando un sitio centralizado donde poder acceder y comprar el producto, dando opción a los usuarios a dejar su opinión, y colaborar con el proyecto a través de donaciones si así lo desean.

Del mismo modo, existirá una segunda parte de la plataforma, orientada específicamente a los desarrolladores, dándoles la posibilidad de compartir su perfil, y sus trabajos.

Cada desarrollador tendrá un portafolio asociado a su perfil, donde podrá compartir todos los proyectos en los que ha participado que desee compartir, destacando su rol en dicho desarrollo (programador, artista de concepto, animador, modelador 3d, analista, etc.), de modo que los estudios interesados en perfiles como el suyo puedan acceder dicho portafolio, observar sus trabajos, y contactar con el si así lo desearan, a través de un sistema de mensajería interno.

Del mismo modo, un estudio de desarrollo ya existente puede tener un perfil de estudio en la plataforma, donde se muestren todos los videojuegos desarrollados, y los que estén en desarrollo, publicando ofertas para que los usuarios anteriormente mencionados puedan acceder a ellas y ponerse en contacto, en el caso de estar interesados.

Palabras clave

Independiente, videojuego, buscador, Indie, desarrollador, estudio.

1

Introducción

El proyecto de Indie Hat surge tras varios años de desarrollos fallidos en el mundo de los videojuegos independientes.

Por una parte, he podido observar una falta absoluta de organización por parte de los estudios (y aficionados), dificultando enormemente la labor de encontrar desarrolladores que colaboren con el proyecto que se pretende desarrollar. Esto desemboca en que, gracias a la imposibilidad de encontrar el perfil deseado y la falta de recursos por parte del estudio, son los propios desarrolladores ya existentes los que tienen que realizar dichas labores, para las que no están formados, haciendo que el producto final sea de una peor calidad, y aumentando enormemente los plazos de cada parte del desarrollo.

Tras una investigación y leve contacto, he advertido que, a pesar de haber una cantidad mayúscula de estudios independientes de desarrollo, y un número aún mayor de videojuegos desarrollados cada año, no hay ningún tipo de comunidad creada

alrededor de estos, lo que supone una clara desventaja para los estudios Indie, ante un mercado sobreexplotado y que favorece a los que más recursos tienen.

Del mismo modo, al haber semejante de lanzamientos al año, la posibilidad de éxito, entendiendo como éxito la obtención un beneficio que supere los gastos producidos en el desarrollo y promoción del producto, se vuelve muy remota por diversos factores:

- La industria favorece a las grandes empresas, promocionando únicamente videojuegos "AAA" (Un juego AAA es una clasificación informal utilizada para los videojuegos producidos y distribuidos por una distribuidora importante o editor importante), haciendo al resto de videojuegos invisibles ante el público generalizado.
- El presupuesto necesario para la realización de un videojuego, a pesar de ser considerado de bajo presupuesto, es muy alto para poder financiarse de forma propia en la mayoría de los casos, obligando a los estudios a recurrir a técnicas como el crowdfunding, muchas veces asociado erróneamente con estafas o financiaciones fraudulentas, y suponiendo una negativa para la realización del producto.

Por otra parte, muchas veces el mundo del videojuego independiente es víctima de una concepción errónea del mismo, por parte tanto de la prensa, como de los jugadores, e incluso a veces de los propios desarrolladores. A menudo el sector es identificado como una industria "amateur", lo que supone una falta de seriedad enorme por parte de muchos interesados (desarrolladores), ya que se concibe más como una afición o un hobby, que una actividad o trabajo principal.

Dicha falta de seriedad, supone una negatividad a la hora del desarrollo, ya que los plazos marcados no se cumplen, o simplemente la gente acaba saliendo del proyecto a mitad del desarrollo, si no antes de empezar.

A todos estos problemas se le suma la falta de una centralización de la industria Indie, ya que no existe ninguna página grande (o seria) que se centre en este sector. Al no haber una plataforma centralizada, se acaba recurriendo a las plataformas existentes, centradas en los desarrollos de grandes estudios, y perjudicando enormemente a los estudios Indie.

Con este proyecto, se pretende aportar una solución a estos problemas, aprovechando la falta de una plataforma similar en el mercado. Indie Hat pretende:

- Facilitar la visibilización de los estudios indies, ofreciendo una plataforma donde se puedan exponer de forma centralizada, todos los proyectos de los estudios que deseen publicar su trabajo. Esto supondría una cantidad relativamente elevada de visitas por parte de los usuarios aficionados a este "subgénero" de videojuegos, lo que se traduce en un punto de un interés muy alto para los estudios.
- Ofrecer un punto de reunión entre los usuarios, y los desarrolladores, donde los usuarios puedan ofrecer sugerencias para los desarrolladores, y del mismo modo, opiniones para los próximos usuarios que estén interesados en cada videojuego.
- Crear ofertas de trabajo para los interesados en este sector, estableciendo un lugar centralizado para las ofertas de este tipo.
- Facilitar a los estudios la búsqueda de perfiles concretos para cada tipo de desarrollo.
- Facilitar a los desarrolladores un lugar donde exponer sus trabajos, y ofrecer una posibilidad adecuada de contactar con estudios serios.

En conclusión, se pretende facilitar la visualización, contribuir con la proliferación y en general, dar cabida a un sector muy importante del videojuego, que a menudo es ignorado por la industria, por la prensa, y por los propios jugadores.

2

Objetivos

Objetivo Principal

Ofrecer una plataforma donde los desarrolladores puedan exponer su trabajo, donde se pueda ofrecer imágenes, videos, consultar/dejar opiniones, y un acceso al propio videojuego.

Objetivos específicos

- **OBJ 1**
  Ofrecer un catálogo de videojuegos al usuario, filtrando las búsquedas, y con un sistema de etiquetas de modo que la pagina pueda recomendar productos en base a la actividad del usuario.

- El catálogo de videojuegos tendrá un buscador que permita filtrar por nombre, estudio desarrollador, fecha de publicación, género o país de procedencia.

- **OBJ 2**

  Consolidar una plataforma, donde cada desarrollador pueda publicar un portafolio donde publicar sus trabajos, y optar a ofertas de trabajo. Del mismo modo, los estudios que trabajen con Indie Hat podrán buscar perfiles de trabajadores, y enviar ofertas de forma directa.

  - Cada usuario tendrá la opción de tener un perfil de desarrollador asociado a sí.
  - Acceder a ofertas y solicitudes de trabajo, asociadas al perfil de desarrollador de diferentes estudios/desarrolladores.

- **OBJ 3**

  Ofrecer un sistema de comunicación entre usuarios, que funcione como un correo electrónico interno en la página (sistema de mensajes privados), para facilitar las primeras comunicaciones estudio/desarrollador.

  - La comunicación entre usuarios será privada (cifrado asimétrico), y de ningún modo podrá ser leída por ningún tercero.
  - El tiempo de demora entre el envío y recepción de cada mensaje será menor a 5 minutos.

3

Análisis del contexto y estado del arte

Durante los últimos años, el sector del desarrollo de videojuegos ha crecido a pasos agigantados, sobre todo cuando nos referimos a la sección independiente de estos.

Desde 2007 el número de estudios independientes ha crecido de manera exponencial. Según Evans Data (firma de estudio de mercado especializada en desarrollo de Software), se estima que para 2020 habrá 12 millones de desarrolladores independientes.

En la plataforma de videojuegos Steam, plataforma dedicada al soporte y venta de videojuegos, se puede observar un crecimiento muy importante de este sector. Cabe a destacar que Steam no da demasiado soporte a los estudios independientes, de modo que la gran mayoría no suelen trabajar con esta plataforma, aunque se puede observar un crecimiento e interés por estos estudios desde 2015.

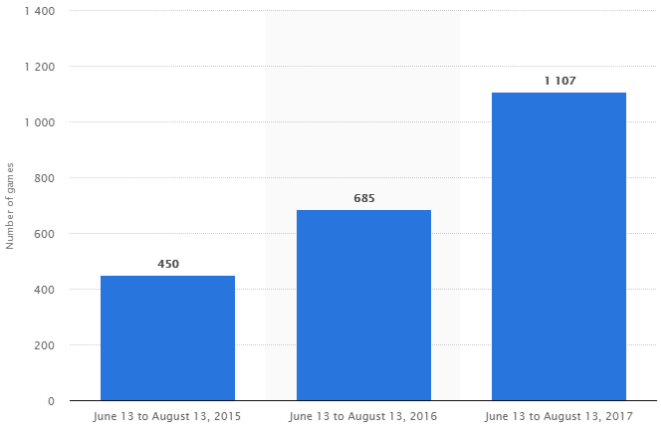


Figura 3.1: Lanzamientos independientes en Steam en 2016, 2017 y 2018.

Del mismo modo se puede observar una media mensual aproximada de lanzamientos a nivel global, indicándonos que hay un especial aumento de lanzamientos en los meses de agosto, septiembre y octubre.

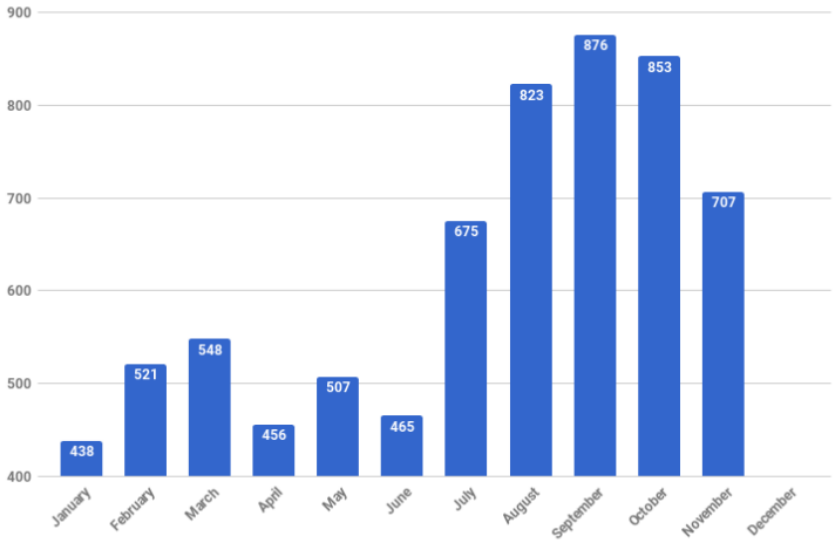


Figura 3.2: Lanzamientos por mes.

3.1 Mercado y clientes potenciales

Es conveniente entra a definir los dos tipos de clientes diferenciados que pueden existir en la plataforma:

Usuario: Este tipo de cliente se interesa por el mundo del videojuego desde el punto de vista de un usuario (jugador). Le interesa la visualización de los mismos y sus noticias, pero siempre desde dicho punto de vista. A lo largo de los últimos años, la

industria del videojuego independiente (comúnmente conocida como Indie) ha crecido lo suficiente, como para suponga una especie de género en sí mismo, donde hay muchos usuarios que únicamente se interesan por este tipo de videojuegos.

Figura 3.3: Perfil medio del jugador en España

Desarrollador: Por otra parte, se encuentran los desarrolladores, los cuales pueden cumplir perfectamente el perfil anteriormente explicado, no obstante, se interesan por los aspectos más técnicos del desarrollo. Son clientes con un conocimiento mucho más profundo del medio, del desarrollo web, y del entorno de los videojuegos. Del mismo modo, se interesa en el perfil de otros desarrolladores, ya sea llegar a un desarrollo conjunto, o para colaboraciones publicitarias, etc. Es común que muchos estudios se

Figura 3.4: Distribución del sector del videojuego

formen con desarrolladores de diferentes partes del mundo, sin necesidad del desplazamiento de estos.

3.2 Análisis de la competencia

Respecto a la competencia, es necesario diferenciar las dos partes que ofrece a la plataforma (distribución de videojuegos, y herramienta de contactos para facilitar el trabajo en el entorno del desarrollo).

Las principales plataformas de competencia de distribución de videojuegos son grandes empresas con cientos de miles de usuarios, no obstante, ninguna de estas se especializa en la industria Indie, haciendo que algunos estudios sean reacios a colaborar con estas.

Steam: Es una plataforma de distribución digital, gestión digital de derechos, comunicaciones y servicios multijugador desarrollada por ValveCorporation. Steam además ofrece varias maneras para la comunicación entre los miembros de la comunidad, la posibilidad de utilizar chat de voz en cualquier momento y actualizaciones automáticas para todos los juegos que ofrece.

Figura 3.5: Logotipo de Steam

Steam apareció originalmente en 2003 como una utilidad para agregar parches y servidores a juegos4 ya existentes como Counter Strike, Day of Defeat, etc.

Para 2008 la página se relanzó como una tienda para comprar juegos en digital.

En enero de 2016, Steam contaba con más de 7300 juegos disponibles,9 de los cuales más de 2700 son compatibles con OS X y más de 1700 con Linux. Además, contaba con cerca de 142 millones de cuentas de usuario activas.10 El 27 de noviembre de

2017 batió el récord de jugadores simultáneos, con 17 millones de jugadores simultáneos.

GOG.com: Previamente conocido como Good Old Games, es un servicio de venta y distribución de videojuegos de la empresa polaca CD Projekt. Este servicio vende principalmente videojuegos de PC antiguos. Para asegurar la compatibilidad con las nuevas versiones de Microsoft Windows, algunos videojuegos son pre parcheados o empaquetados con emuladores de código abierto, como ScummVM y DOSBox.

A diferencia de otros servicios, los videojuegos no usan gestión digital de derechos (DRM),2 por lo tanto el usuario no tiene que instalar un cliente especial para descargar o ejecutar los videojuegos,3 aunque el servicio ofrece un gestor de descargas.

Figura 3.6: Logotipo de GOG

Junto con los videojuegos comprados, los compradores también pueden descargar material extra relacionado con el videojuego que compran. Este material extra puede incluir la banda sonora del videojuego, fondos de pantalla, avatares, manuales, entre otros. GOG también ofrece soporte al cliente para todas sus ventas.

Jump: servicio de suscripción, que te da acceso a una biblioteca de videojuegos independientes. Es una empresa de reciente creación y no hay demasiados datos sobre ella. Se podría considerar competencia directa, no obstante, solo se enfoca a juegos independientes que hayan recibido premios, o hayan sido ampliamente exitosos, por lo que se aleja bastante del objetivo principal de la plataforma a desarrollar (dar cobertura a los estudios con menos recursos).

Figura 3.7:: Logotipo de Jump

La siguiente tabla muestra un resumen comparativo entre todas las plataformas comentadas.

Tabla 3.1: Comparativa de plataformas

Plataforma	Fortalezas	Debilidades	¿Que ofrece Indie Hat?
STEAM	- **Catalogo** amplio. - **Principal** plataforma del sector. - Sistema de **ofertas** exclusivo.	- **Poco soporte** a los estudios indies. - Resulta **difícil** (y caro) **publicar** tu videojuego.	- **Cobertura exclusiva** a los estudios **indies**. - Publicación **gratuita**.
gog com	- Amplio catálogo de **desarrollo propio**. - **Material exclusivo** (bandas sonoras, arte conceptual, etc.)	- El **material** ofrecido que no es de su propio desarrollo, resulta **escaso**.	- **Todo producto** en la plataforma, ofrecerá **elementos extras** como banda sonora, arte, etc. (Ya sea de forma gratuita, o de pago).
Jump	- **Catálogo** dedicado **exclusivamente** al sector **Indie**. - Ofrece un gran número de **títulos de renombre** en el sector.	- **No** da **cobertura** a **estudios nuevos**, solo ofrece juegos premiados y reconocidos.	- Acceso **libre** y **gratuito** a cualquier estudio.

3.3 Análisis tecnológico

El entorno tecnológico en España siempre ha ido un poco rezagado en comparación con otros países. Sin embargo, en los últimos tiempos se ha producido un cambio significativo hasta el punto que las posibilidades que ofrece este sector son realmente cuantiosas. De entre las empresas destacadas, las operadoras se encuentran en primera fila, y son las que más probabilidades tienen de crecer en esta materia.

El sector tecnológico en el mercado español cada vez suma mayores enteros. Aunque se ha movido en un letargo de gran envergadura, su momento actual es mucho más esperanzador. Solo hay que ver los últimos datos. Y es que esta industria consiguió una cifra de negocio global de 28.002 millones de euros el pasado ejercicio, lo que representa un 9,1% más que del año anterior.

Sin embargo, lo relevante es las previsiones que existen para los años que están por venir. La disrupción y la investigación en el entramado de la tecnología es una de las áreas que más réditos pueden dar a muchas compañías españolas. Según fuentes del mercado "las operadoras, las consultoras, todas aquellas que también están apostando por el tratamiento de datos tienen posibilidades inmensas para seguir una dinámica creciente en su volumen general".

Figura 3.8: Distribución de empresas informáticas en España

Los parques tecnológicos, muchos de ellos promovidos por universidades, ejercen de polos de atracción para empresas digitales y tecnológicas que basan su modelo de negocio en la innovación y la transferencia de conocimiento. Según datos de la Asociación de Parques Científicos y Tecnológicos de España, estos centros de investigación albergaban en 2015 a 7.736 empresas en nuestro país. El 22,7% de estas empresas e instituciones se enmarcan en el ámbito de la información, la informática y las telecomunicaciones.

3.4 Innovación

El principal objetivo de la plataforma es apoyar a los estudios independientes, y con escasos recursos, creando una comunidad, y favoreciendo la economía circular, de modo que los usuarios compartan archivos de utilidad común, ahorrando mucho tiempo y costes a la hora del desarrollo. Del mismo modo se pretende dar mayor cabida y acceso a los proyectos que puedan desarrollar dichos estudios.

Por otra parte, se pretende favorecer un mercado de oportunidades para la gente especializada (o que quiera trabajar) en el sector de los videojuegos, ofreciendo la posibilidad de exponer tu perfil laboral, o pedir uno, en un mercado especializado.

A largo plazo, contando con un relativo éxito, y extensión en el público, se apoyará a la industria ofreciendo otro tipo de servicios, además de los ya citados, tales como, cursos online especializados en ciertos aspectos del negocio, charlas de desarrolladores experimentados, premios favoreciendo un desarrollo sostenible, etc.

4

Análisis de requisitos

4.1 Requisitos funcionales

Numero de requisito	**RF1**
Nombre del requisito	Alojamiento y descarga de archivos
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 1* La plataforma puede gestionar el alojamiento y descarga de archivos de gran tamaño.
Prioridad	Alta

Numero de requisito	**RF2**
Nombre del requisito	Gestión de compras
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 1* La plataforma debe de ser capaz de gestionar compras de productos.
Prioridad	Alta

Numero de requisito	**RF3**
Nombre del requisito	Registro
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 2* El usuario tendrá la opción de registrarse
Prioridad	Alta

Numero de requisito	**RF4**
Nombre del requisito	Sistema de mensajería interno
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 3* La plataforma debe incluir un sistema de mensajes privados, para la comunicación entre usuarios. La comunicación entre usuarios se cifrará a través de un sistema de clave asimétrica, protegiéndola así de terceros.
Prioridad	Media

Numero de requisito	**RF5**
Nombre del requisito	Gestión de perfiles
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 2* La plataforma, en su parte orientada a desarrolladores, deberá incluir un sistema de perfiles, con portafolio, y perfil laboral.
Prioridad	Media

Numero de requisito	**RF6**
Nombre del requisito	Búsqueda de ofertas
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 2* La plataforma incluirá un sistema de visualización para ofertas de empleo específicas, pudiendo filtrarse las mismas para su búsqueda.
Prioridad	Media

Numero de requisito	**RF7**
Nombre del requisito	Buscador de videojuegos
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 1* La plataforma ofrecerá un servicio de búsqueda de videojuegos, con filtros por género, procedencia, nombre, etc.
Prioridad	Media

Numero de requisito	**RF8**
Nombre del requisito	Recomendaciones.
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 1* La plataforma ofrecerá una serie de recomendaciones de videojuegos, en base a la actividad (compras y visualización) del usuario.
Prioridad	Baja

Numero de requisito	**RF9**
Nombre del requisito	Valoraciones y opiniones.
Tipo	Requisito
Fuente del requisito	*Objetivo específico OBJ 1* Cada videojuego podrá ser valorado de 1 a 5 estrellas, mostrándose en la visualización general una media de las puntuaciones obtenidas. Del mismo modo, a la hora de publicar una opinión de cada videojuego, estas podrán ser valoradas de 1 a 5 estrellas, con un funcionamiento igual al anteriormente descrito.
Prioridad	Media

4.2 Requisitos no-funcionales

Numero de requisito	**RNF1**
Nombre del requisito	Compatibilidad de navegador
Tipo	Restricción
Fuente del requisito	La plataforma debe funcionar en los principales navegadores (Google Chrome, Mozilla Firefox y Microsoft Edge).
Prioridad	Alta

Numero de requisito	**RNF2**
Nombre del requisito	Respuesta en transacciones
Tipo	Restricción
Fuente del requisito	Toda funcionalidad y transacción de pago debe responder al usuario en menos de 6 segundos.
Prioridad	Media

Numero de requisito	**RNF3**
Nombre del requisito	Capacidad de usuarios
Tipo	Restricción
Fuente del requisito	El sistema debe ser capaz de operar adecuadamente hasta con 1.000 usuarios a la vez.
Prioridad	Media

Numero de requisito	**RF4**
Nombre del requisito	Login
Tipo	Restricción
Fuente del requisito	El usuario debe de ser capaz de poder acceder al sistema a través de un sistema de login/registro, quedando el acceso a la página completamente restringido de no estarlo.
Prioridad	Alta

5

Diseño

5.1 Diagrama físico y/o lógico de red

El proyecto está formado por:

- Una base de datos MySQL que se encargará de almacenar todos los datos referentes a la plata forma, desde todo lo referente a los usuarios y sus configuraciones, como a la administración de los videojuegos o la propia administración interna de la página.
- Un servidor Ubuntu Server 18.04 LTS (con apache). Dicho servidor contendrá toda la programación (clases y páginas PHP), y todas las páginas Web (HTML), a las que el cliente accederá.
- Además, en dicho servidor, habrá una serie de clases que se comunicarán con APIs externas al servidor, tales como YouTube, IGDB (API que ofrece una base de datos internacional de videojuegos, fechas de lanzamiento etc.), que servirán para la visualización de ciertos elementos de la ficha de cada videojuego, como videos informativos, tráiler, etc.
- Una pasarela de pago externa a la página (proporcionada por una entidad bancaria), para gestionar los pagos que se vayan a realizar en la plataforma.

5.2 Diagrama E/R

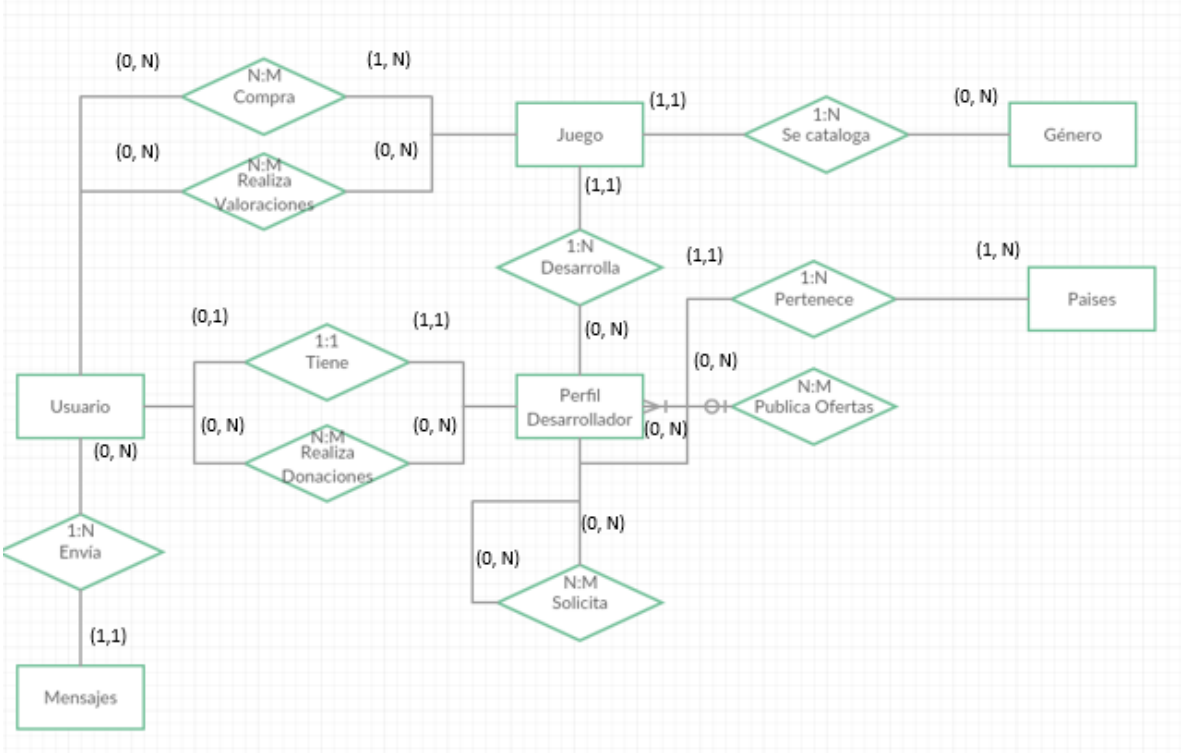


Figura 5.1: Diagrama E/R

5.3 Explicación de la Base de Datos

La base de datos está formada por 11 tablas que contienen toda la información relativa al proyecto, desde la información de los usuarios y los videojuegos, a las opiniones generadas por los usuarios, o los mensajes enviados entre los mismos.

Las principales tablas son **usuario**, perfil_**desarrollo** y **juego**, debido a la información que contienen y las relaciones que dependen de estas.

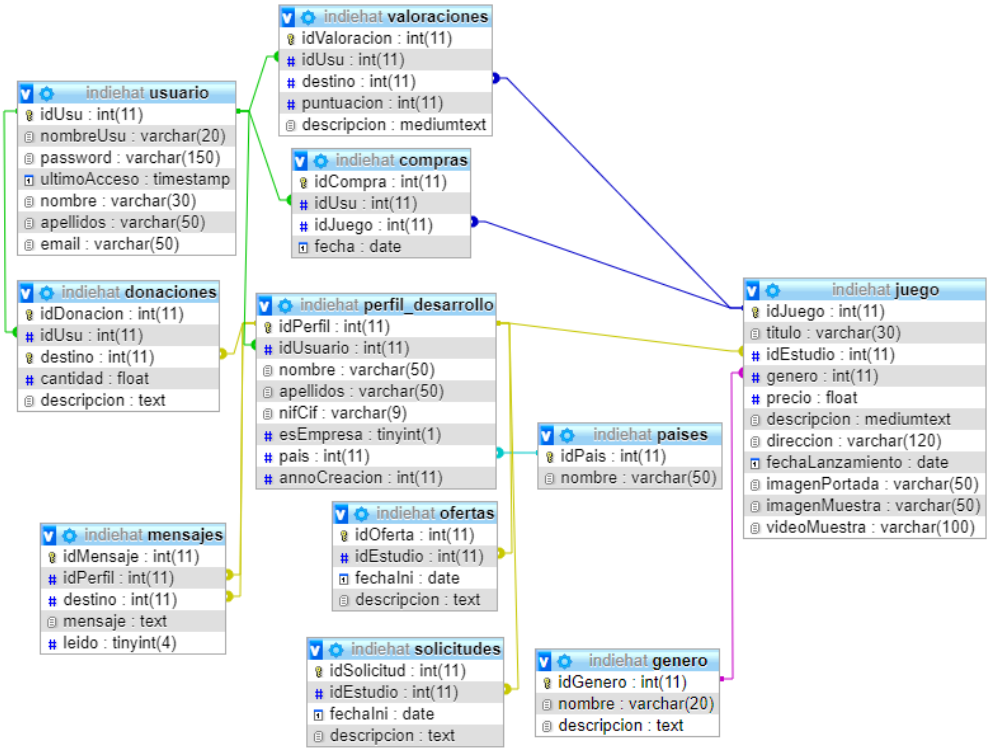


Figura 5.2: Descripción de la base de datos

- usuario: Esta tabla hace referencia a los datos del usuario (como usuario consumidor de videojuegos, no como desarrollador).
    - idUsu: Identificador numérico autoincremental. Clave única para cada usuario.
    - nombreUsu: Nombre asignado a cada usuario para identificarse en la página.
    - password: Contraseña con la que accede. Esta ira cifrada en md5.
    - ultimoAcceso: Campo que registra la fecha y hora del último acceso de cada usuario.
    - nombre: Nombre del propietario de la cuenta.
    - apellidos: Apellidos del propietario de la cuenta.
    - email: Correo electrónico del propietario de la cuenta.
- mensajes: Esta tabla guarda los mensajes enviados entre los usuarios por el sistema de mensajería interno de la página.
    - idMensaje: Identificador numérico autoincremental. Clave única para cada registro.
    - idUsu: Hace referencia al usuario que envía el mensaje.

  - destino: Hace referencia al usuario al que va destinado el mensaje.
  - mensaje: Texto enviado.
  - leído: Variable de tipo booleano, que indica si el mensaje ha sido leído o no.

- donaciones: Esta tabla esta está contemplada para que los usuarios puedan realizar donaciones a los estudios que deseen. (Esta tabla y su funcionalidad han quedado excluidas en la implementación de la primera versión).
  - idDonacion: Identificador numérico autoincremental. Clave única para cada registro.
  - idUsu: Identificador del usuario que realiza la donación.
  - destino: Identificador del estudio al que va dicha donación.
  - cantidad: Cantidad registrada a donar.
  - descripción: Campo de tipo texto, en el caso de que el usuario desee añadir un texto con su donación.
- valoraciones: Contiene todos los datos referentes a las valoraciones realizadas por los usuarios en cada videojuego.
  - idValoracion: Identificador numérico autoincremental. Clave única para cada registro.
  - idUsu: Identificador del usuario que realiza la valoración.
  - destino: Identificador del videojuego sobre el que se realiza la valoración.
  - puntuación: Numero entero de 1 a 5 (número de estrellas) con el que el usuario valora el videojuego en cuestión.
  - descripción: Campo de tipo texto, donde el usuario realizará una breve descripción de su valoración.

- compras: En esta tabla se registrarán todas las compras realizadas por cada usuario.
  - idCompra: Identificador numérico autoincremental. Clave única para cada registro.
  - idUsu: Identificador del usuario que realiza la compra.
  - idJuego: Identificador del videojuego a comprar.
  - fecha: Fecha en la que se realiza la transacción.
- perfil_desarrollo: Esta tabla contiene todos los datos de los usuarios, que además de consumidores de videojuegos, se identifican como desarrolladores.

- idPerfil: Identificador numérico autoincremental. Clave única para cada registro.
- idUsuario: Identificador del usuario registrado como desarrollador.
- nombre: Nombre de persona al cargo como desarrollador, o nombre del estudio (no siempre es el mismo que el usuario que registra la cuenta, dado que a veces las cuentas pueden ser de estudios enteros).
- apellidos: Apellidos de la persona al cargo.
- nifCif: Identificador fiscal de la empresa o persona.
- esEmpresa: Campo de tipo boolean que nos dice si es una empresa o un desarrollador autónomo.
- pais: Identificador del país de procedencia.
- annoCreacion: Registra la fecha en la que empieza su actividad el estudio o desarrollador.

- ofertas: Registra todas las ofertas de trabajo que un usuario tiene asociadas a sí.
  - idOferta: Identificador numérico autoincremental. Clave única para cada registro.
  - idEstudio: Identificador del estudio que publica la oferta.
  - fechaIni: Campo de tipo fecha, que registra la fecha de publicación.
  - fechaFin: Campo de tipo fecha, que registra la fecha límite de la oferta (en caso de haberla).
  - descripcion: Descripción de la oferta.

- juego: Esta tabla registra toda la información relativa a cada videojuego.
  - idJuego: Identificador numérico autoincremental. Clave única para cada registro.
  - idEstudio: Identificador del estudio que publica el videojuego.
  - genero: Identificador que indica el género del videojuego.
  - precio: Precio con el que se publica dicho videojuego.
  - descripcion: Breve descripcion que el desarrollador ofrece como introducción al videojuego.
  - direccion: Dirección web que apunta al sitio oficial del juego, en caso de existir.
  - fechaLanzamiento: Fecha de lanzamiento del videojuego.

  - imagenPortada: Almacena una cadena de caracteres que indica el nombre que tiene la imagen subida del videojuego que se mostrará como portada.
  - imagenPerfil: Almacena una cadena de caracteres que indica el nombre que tiene la imagen subida del videojuego que se mostrará como foto principal en su ficha.
  - videoMuestra: Almacena una cadena de caracteres que indica el nombre que tiene la dirección del video que se mostrará a modo de tráiler.

- paises: Registra todos los países soportados por la página.
  - idPais: Identificador numérico autoincremental. Clave única para cada registro.
  - nombre: Nombre del pais.
- solicitudes: Registra todas las solicitudes de trabajo que un usuario realiza.
  - idSolicitud: Identificador numérico autoincremental. Clave única para cada registro.
  - idEstudio: Identificador del estudio al que va enviada la solicitud.
  - fechaIni: Campo de tipo fecha, que registra la fecha de solicitud.
  - descripcion: Descripción de la solicitud.

- genero: Esta tabla registra los géneros de videojuegos registrados en la página.
  - idGenero: Identificador numérico autoincremental. Clave única para cada registro.
  - nombre: Nombre que recibe el género en cuestión.
  - descripcion: Debido a que muchas veces los nombres no son descriptivos para el género, se ofrece un campo con una breve descripcion del mismo.

5.4 Diagrama de clases

Las clases incluidas en el proyecto son las siguientes

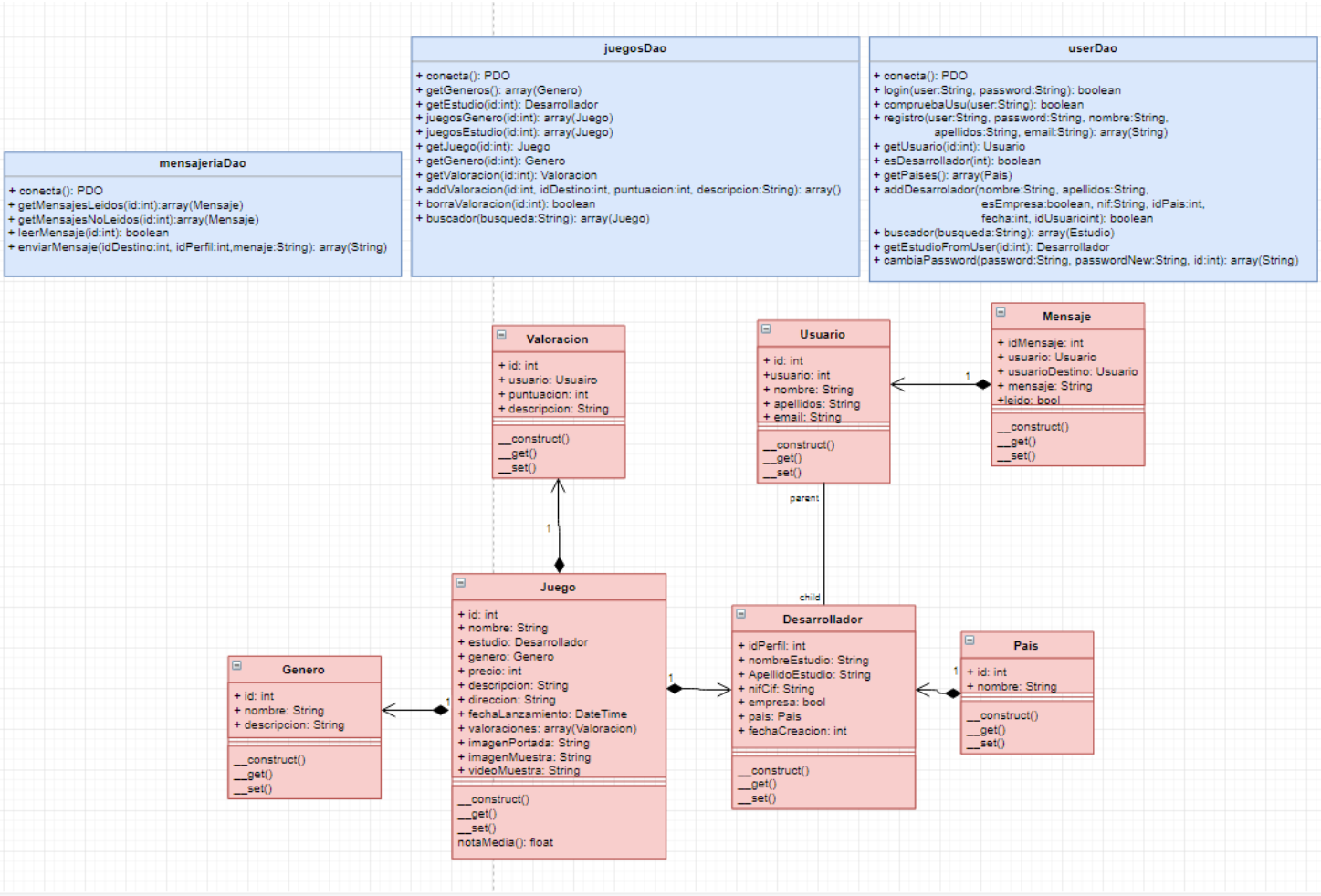


Figura 5.3: Diagrama UML

- userDao: Esta clase sirve como clase de acceso a los datos, y contiene todos los métodos necesarios para acceder a la base de datos, en relación a los usuarios de la base de datos.
    - conecta(): Devuelve un objeto tipo PDO. Necesario para realizar todas las querys necesarias, en el resto de métodos.
    - login(): Introduce la combinación de usuario y contraseña, y compara contra la base de datos, devolviendo verdadero en el caso de ser correcto. Del mismo modo guarda en sesión el nombre y la id del usuario, para controlar su sesión en la página.

- compruebaUsu(): Comprueba si ya existe, o no, otro usuario con el mismo nombre en la página.
- registro(): Guarda los datos del usuario para su posible acceso en la página.
- getUsuario(): Devuelve un objeto de tipo usuario, dada su id en la base de datos.
- esDesarrollador(): Dado un id de usuario, devuelve verdadero o falso, en caso de que tenga asociado un perfil de desarrollador, o no.
- getPaises(): Devuelve un array con todos los objetos de tipo Pais guardados en la base de datos.
- addDesarrollador(): Crea un perfil de desarrollador en la base de datos, asociado al usuario que lo crea.
- buscador(): En base a una palabra, o conjunto de ellas, busca en la base de datos los desarrolladores correspondientes a ella, comprobando si coincide con su nombre, o su país. Devuelve todas las coinciencias en forma de array de objetos tipo Desarrollador.
- getEstudioFromUser(): Devuelve un objeto tipo Desarrollador dada una id de Usuario.
- cambiaPassword(): Comprueba que la contraseña antigua coincida, y cambia la contraseña del usuario por la deseada.

- JuegosDao: Esta clase sirve como clase de acceso a los datos, y contiene todos los métodos necesarios para acceder a la base de datos, en relación a los videojuegos registrados en la base de datos.
  - conecta(): Devuelve un objeto tipo PDO. Necesario para realizar todas las querys necesarias, en el resto de métodos.
  - getGeneros(): Devuelve un array con todos los objetos de tipo Genero guardados en la Base de datos.
  - getEstudio(): Devuelve un objeto de tipo Desarrollador, dada una id de desarrollador.
  - juegosGenero(): Devuelve un array de objetos tipo Juego, dada la id de un género.
  - juegosEstudio(): Devuelve un array de objetos tipo Juego pertenecientes a un estudio, dada su id de Desarrollador.
  - getJuego(): Devuelve un objeto tipo Juego dada su id.
  - getGenero(): Devuelve un objeto tipo Genero, dada su id.
  - getValoracion(): Devuelve un objeto tipo Valoracion, dada su id.

  - addValoracion(): Añade en la base de datos una valoración establecida por el usuario, asociada a un juego.
  - borraValoracion(): Borra de la base de datos una valoración establecida por un usuario.
  - buscador(): En base a una palabra, o conjunto de ellas, busca en la base de datos los juegos correspondientes a ella, comprobando si coincide con su nombre, su género, o el nombre del estudio desarrollador. Devuelve todas las coincidencias en forma de array de objetos tipo Juego.

- mensajeriaDao:
  - conecta(): Devuelve un objeto tipo PDO. Necesario para realizar todas las querys necesarias, en el resto de métodos.
  - getMensajesLeidos(): Devuelve un array de objetos tipo Mensaje, dados todos los mensajes de un usuario con el campo "leido" en verdadero.
  - getMensajesNoLeidos():Devuelve un array de objetos tipo Mensaje, dados todos los mensajes de un usuario con el campo "leído" en falso.
  - leerMensaje(): Establece el campo "leído" de un determinado mensaje, en verdadero.
  - enviarMensaje(): Guarda en la base de datos un nuevo mensaje, asociado a un usuario.

- Usuario: Esta clase contiene todos los datos del usuario que esta logueado en cada momento.
  - id: Identificativo principal del usuario
  - usuario: Nombre con el que se registra
  - nombre: Nombre del usuario
  - apellidos: Apellidos del usuario
  - email: Correo electrónico asociado al usuario.

- Desarrollador: Clase que hereda de la clase Usuario. Esta clase es para los usuarios que también tengan asociado un perfil de desarrollador con ellos.
  - idPerfil: Clave numérica y única, identificativo de este objeto como Desarrollador.
  - nombreEstudio: Nombre de la persona al cargo o del estudio (No tiene por qué coincidir con el nombre en el campo Usuario, como ya se explicó en el diagrama E/R).
  - apellidosEstudio: Coincidiendo con el campo anterior, apellidos del desarrollador en caso de que no se trate de una empresa.

  - nifCif: Identificación fiscal del desarrollador.
  - pais: Objeto de tipo Pais, correspondiendo al país de actividad del estudio/desarrollador.
  - fechaCreacion: Año de creación del estudio.

- Mensaje: Clase que recoge un mensaje emitido.
  - idMensaje: identificador numérico del mensaje.
  - usuario: Objeto de tipo Usuario que recoge el usuario que ha emitido el mensaje
  - destino: Objeto de tipo Usuario que recoge destino del mensaje.
  - mensaje: Campo de texto que recoge el mensaje en cuestión.
  - leído: Campo de tipo booleano que indica si el mensaje ha sido leído o no.

- Juego: Clase principal de la plataforma junto a Desarrollador. Recoge todos los datos de cada videojuego a exponer.
  - id: identificador numérico de cada videojuego.
  - nombre: Campo de tipo String, que indica el título del juego.
  - idEstudio: Objeto de tipo estudio que hace referencia al estudio desarrollador del videojuego.
  - genero: Objeto de tipo Genero, correspondiente al género del juego.
  - precio: Precio del videojuego.
  - descripcion: Cadena de texto que el desarrollador ofrece como descripcion de su producto.
  - direccion: Cadena de texto que hace referencia (en caso de haberlo) a la dirección de donde se encuentra el archivo a descargar (funcionalidad no incluida en la primera versión).
  - fechaLanzamiento: Objeto tipo DateTime que indica en qué fecha se publicó el videojuego.
  - valoraciones: Array de objetos tipo Review, que recoge todas las valoraciones hechas al videojuego.
  - imagenPortada: Cadena de texto correspondiente a la imagen a mostrar en la portada del juego.
  - imagenMuestra: Cadena de texto correspondiente a la imagen a mostrar en el perfil del juego.

  - videoMuestra: Cadena de texto correspondiente a la dirección del video a mostrar en el perfil del juego.
  - notaMedia(): Devuelve la nota media del videojuego (de 1 a 5) obtenida por todas las valoraciones emitidas a este.
- Valoracion: Esta clase representa cada valoración emitida sobre el videojuego.
  - id: identificador numérico de la valoración.
  - usuario: Objeto de tipo Usuario, que recoge el usuario que emite dicha valoración.
  - descripcion: Cadena de texto que describe la opinión emitida por el usuario.
  - puntuacion: Numero entero de 1 a 5, que recoge la nota de la valoración emitida.
- Genero: Esta clase representa el género al que pertenece el videojuego, y por el que es catalogado en la página.
  - id: Identificador numérico del genero.
  - nombre: Nombre con el que es referido el género en cuestión.
  - descripcion: Campo de tipo texto que describe en lo que se basa el género.
- Pais: Esta clase representa los diferentes países a los que puede pertenecer un estudio registrado en la página.
  - id: Identificador numérico del país.
  - nombre: Nombre del país en cuestión.

6.5 Diagrama de interfaces

6.5.1 Diseño de la pagina

Colores utilizados

Los colores mostrados son los colores utilizados en la implementación final de la página. Como se puede observar, se ha optado por una combinación de colores completamente diferente a la propuesta en el diseño inicial, dado a que, entre otros motivos, el amarillo se relaciona más fácilmente con el mundo de la tecnología, e indirectamente con el mundo de los videojuegos. Del mismo modo, se puede observar

que la nueva combinación de colores le da a la página un aspecto más actualizado, más adecuada al año actual, ya que la combinación de gris y rojo hacia que pareciera mucho más antigua de lo que realmente es.

#FBB710	#FFB320	#131313	#1F2225
Tono Principal	**Tono Secundario**	**Fondo principal**	**Fondo secundario**

A continuación, se muestran 3 capturas referentes a las secciones principales de la plataforma, en contraposición al diseño final de la página, donde se pueden ver los cambios que ha sufrido la misma.

Login

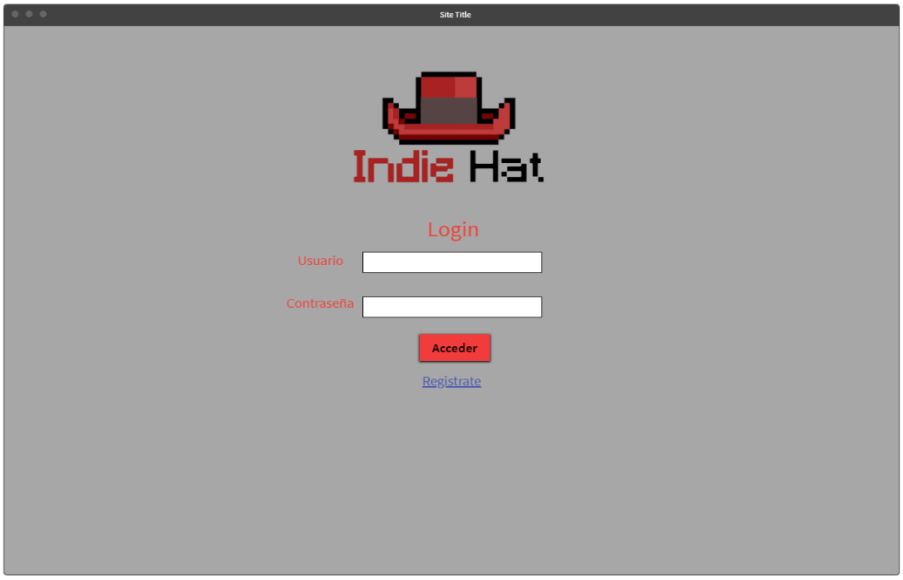


Figura 5.4: Diseño de Login

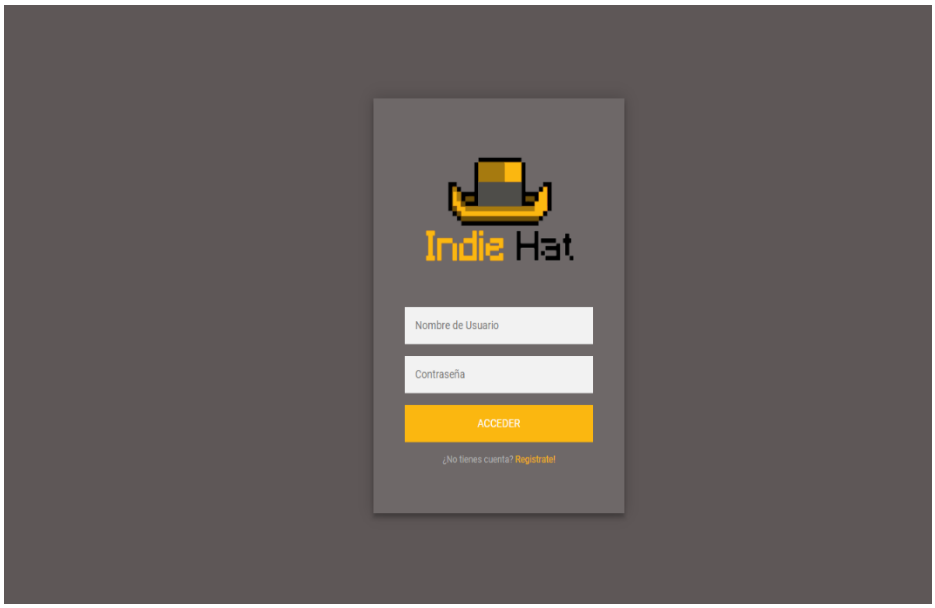


Figura 5.5: Login Final

Cartelera de juegos (página principal):

Esta página ofrece una cartelera de videojuegos, donde cada uno es mostrado con una imagen y una breve sección de la información del mismo. Cada imagen es un enlace a la ficha principal de cada videojuego donde se ofrece la información de cada uno.

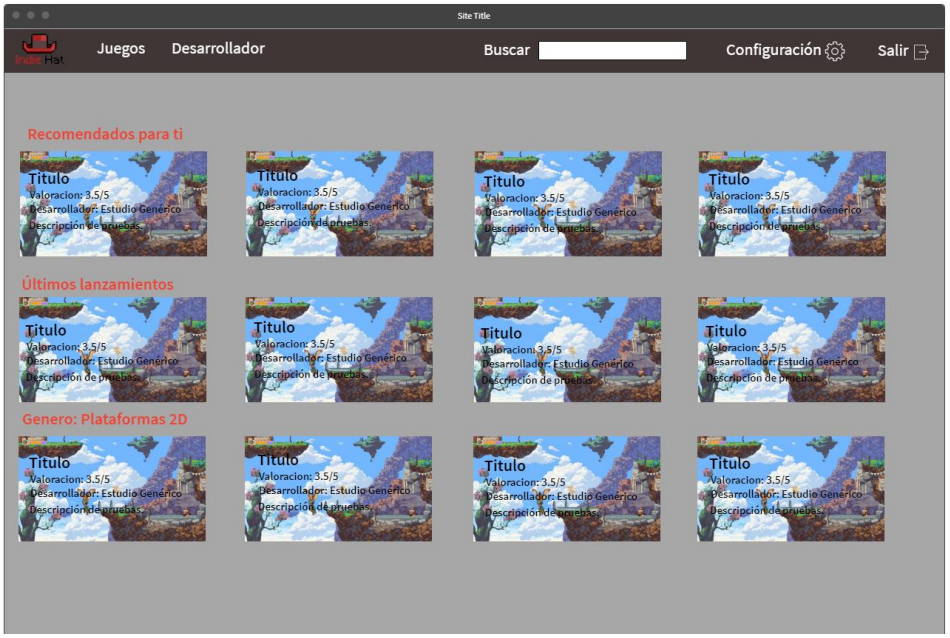


Figura 5.6: Diseño de sección de juegos

Del mismo modo, se ofrece un menú con secciones de configuración, enlace a las otras secciones de la página, y un buscador general para la página, donde se pueden realizar

búsquedas filtradas tanto de videojuegos, como de búsquedas referentes a la sección de desarrolladores.

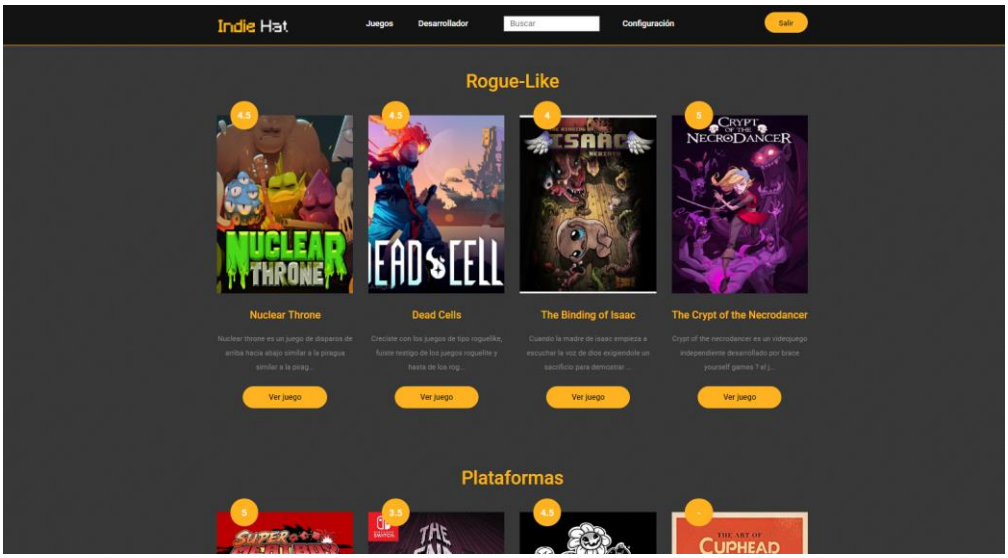


Figura 5.7: Diseño de sección de juegos final

Panel de desarrollador:

Esta página ofrece un panel de desarrollador donde cada usuario con un perfil de desarrollador asociado puede acceder, y ver sus mensajes recibidos, sus solicitudes enviadas (y en qué estado se encuentran), y sus ofertas seguidas (y en qué estado se encuentran).

Del mismo modo, a la derecha se puede acceder a los desarrollos en los que pueda estar ya involucrado el usuario, con una ficha de información de cada uno.

Finalmente se puede acceder a una sección de ofertas, donde se pueden ver las ofertas de trabajo, ordenadas de mayor a menor relevancia, funcionando de un modo similar a la cartelera de videojuegos.

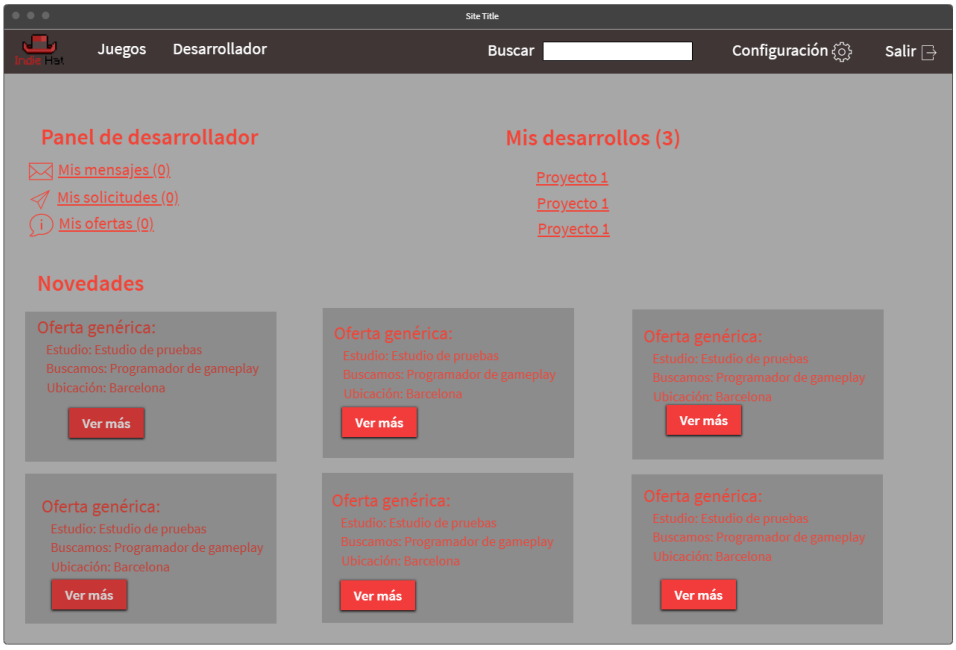


Figura 5.8: Diseño de sección del desarrollador

Figura 5.9: Sección del desarrollador final

6

Planificación

6.1 Diagrama de Gantt

En este diagrama de Gantt queda reflejada la planificación semanal, desde el 08/04/2019 hasta el 16/06/2019. En la sección descripción se puede ver una información más detallada de cada tarea, y cuanto se estima que se tardará.

Nº	Actividad	Descripción	Duracion	F.Inicio	F.Fin	Materiale	Recursos Humanos	Recursos Tecnologico	(08-04-2019 a 16-06-2019) 1 2 3 4 5 6 7 8 9
1	Proceso de investigación	Proceso previo al desarrollo. Investigar competencias, posibilidades de desarrollo, etc.	4 horas	08/04/2019	10/04/2019	Ninguno	Oscar Fontán	Microsoft Word.	
2.2	Diseño de BD	Diseño de tablas, campos y relaciones.	4 horas	11/04/2019	12/04/2019	Ninguno	Oscar Fontán	Gestor de BD (MySQL Workbench)	
2.3	Pruebas en BD	Pruebas de dichas relaciones. Inserciones y busquedas de prueba (con sus correspondientes correcciones).	2 horas	15/04/2019	15/04/2019	Ninguno	Oscar Fontán	Gestor de BD (MySQL Workbench)	
3.1	Diseño de la Plataforma	Diseño general de la plataforma. Su funcionamiento, clases y algoritmos principales.	10 horas	16/04/2019	19/04/2019	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text	
3.2	Diseño de las clases	Diseño completo de las clases que tendra la plataforma. Sus metodos mas importantes y sus relaciones (herencias, etc.)	8 horas	22/04/2019	########	Ninguno	Oscar Fontán	Lucid Charts	
3.3	Diseño de la web (frontend)	Diseño y creación de todo el frontend de la página (menús, secciones, etc.)	10 horas	23/04/2019	########	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
3.4	Desarrollo e Implementación de las clases	Creación e implementación de las clases anteriormente mencionadas.	10 horas	06/05/2019	10/05/2019	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
3.5	Pruebas de las clases	Con las clases y la base de datos creadas, realizar pruebas y correcciones sobre estas.	4 horas	13/05/2019	14/05/2019	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
4.1	Desarrollo de clases (que consumen APIs externas)	Diseño y desarrollo de las clases que consumiran APIs externas (youtube, para la muestra de videos o trailers si los hay).	10 horas	14/05/2019	17/05/2019	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
4.2	Pruebas de las APIs	Pruebas de las anteriores clases, y correcciones ante posibles errores.	8 horas	20/05/2019	########	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
5.1	Implementación de conjunto	Implementación en conjunto de todas las fases de desarrollo en la misma plataforma.	10 horas	24/05/2019	########	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
5.2	Pruebas finales	Pruebas y corrección de errores surgidos en la plataforma completamente implementada.	10 horas	30/05/2019	########	Ninguno	Oscar Fontán	VisualStudio Code. Sublime Text. Google	
6	Elaboración de estimaciones economicas	Elaboración de documentación, y estimaciones económicas del negocio.	4 horas	06/06/2019	########	Ninguno	Oscar Fontán	Microsoft Word. Microsoft Excel	
7	Documentación final del proyecto	Documentación final de la plataforma, y elaboración de la memoria.	14 horas	10/06/2019	14/06/2019	Ninguno	Oscar Fontán	Microsoft Word.	

6.2 Definición de recursos y logística necesarios

En la siguiente tabla se presentan todos los recursos de software, hardware y mantenimiento necesarios para completar el desarrollo.

Dado que es un desarrollo relativamente pequeño, y el presupuesto al que se accede se asume limitado, se ha buscado que la mayoría de elementos sean gratuitos. El mayor costo del proyecto recae sobre la propia contratación de personal (1 empleado).

Tabla 6.1: Logística

Elemento	Coste
Software	
VisualStudio Code	Gratuito
SO Windows 10 PRO	48,99€
XAMPP	Gratuito

Sublime Text	Gratuito
GIMP	Gratuito
Hardware	
Pantalla AOC 22B1H 21.5" LCD	79,99€
Periféricos	35,00€
Pc Sobremesa - PcCom StartUp Intel Core i5-8400/8GB/240SSD	598,59€
Mantenimiento	
Suministro eléctrico	50,00€/mes (150€)
Hosting Web (1&1 IONOS)	12,00€ (primer año)
Empleados	
Un empleado	900€/mes (2.700,00€)
TOTAL	**3.624,57 €**

7

Implementación

7.1 Resumen

La aplicación de Indie Hat ha sido desarrollada siguiendo una arquitectura Modelo Vista Controlador (MVC), correspondiéndose con tres clases de acceso a la Base de Datos desde el lado del Modelo, dos controladores correspondientes a las dos secciones principales de la plataforma, y once vistas correspondientes a las diferentes secciones de la página.

Figura 7.1: Esquema MVC

7.2 Clases Principales

Todas las clases de la aplicación siguen la misma estructura

```php
<?php
class Pais
{
    protected $id;
    protected $nombre;

    public function __construct($id, $nombre)
    {
        $this->id = $id;
        $this->nombre = $nombre;
    }

    public function __get($propiedad)
    {
        if (property_exists($this, $propiedad))
        {
            return $this->$propiedad;
        }
    }

    public function __set($propiedad, $valor)
    {
        if (property_exists($this, $propiedad))
        {
            $this->$propiedad = $valor;
        }
    }
}
?>
```

Figura 7.2: Ejemplo de clase

El constructor es definido al principio de la clase, y todos los métodos get y set son definidos a través de los llamados *métodos mágicos*, permitiendo llamar directamente al parámetro en cuestión, en vez de al método, facilitando la codificación el algún caso.

7.2.1 Herencias

La clase Desarrollador es una clase que hereda de la clase Usuario, ya que se trata de una adición de la misma, pero que no todos los usuarios tienen.

```php
class Desarrollador extends Usuario
{
    protected $idPerfil;
    protected $nombreEstudio;
    protected $apellidosEstudio;
    protected $nifCif;
    protected $empresa;
    protected $pais;
    protected $fechaCreacion;


    public function __construct($id, $usuario, $nombre, $apellidos, $email, $idPerfil, $nombreEstudio,
    $apellidosEstudio, $nifCif, $empresa, $pais, $fechaCreacion)
    {
        parent::__construct($id, $usuario, $nombre, $apellidos, $email);
        $this->idPerfil = $idPerfil;
        $this->nombreEstudio = $nombreEstudio;
        $this->apellidosEstudio = $apellidosEstudio;
        $this->nifCif = $nifCif;
        $this->empresa = $empresa;
        $this->pais = $pais;
        $this->fechaCreacion = $fechaCreacion;
    }
```

Figura 7.3: Clase con herencia

7.2.2 Objetos compuestos por otros objetos

Del mismo modo, se puede observar que hay varias clases cuyos elementos son otros objetos, o arrays compuestos por estos. Esto está diseñado de esta manera para facilitar el acceso a los datos, por ejemplo: Acceder a la descripción de una valoración asociada a un objeto.

7.2.3 Otros métodos

Un método a destacar en una de las clases definidas en la aplicación es el de notaMedia() de la clase juego, que obtiene la nota media de sí mismo en base a todas las valoraciones emitidas, recogidas en el array de Valoraciones de la propia clase.

```php
//Esta funcion obtiene la nota media de todas las valoraciones emitidas a si. En caso de no haber, devuelve '-'
public function notaMedia()
{
    $res = 0;
    foreach($this->valoraciones as $valAux)
    {
        $res+=$valAux->puntuacion;
    }
    if($res!=0)
    {
        $res = $res/sizeof($this->valoraciones);
        $res = round($res,2);
    }
    else
    {
        $res = "-";
    }

    return $res;
}
```

Figura 7.4: Método Nota-Media

7.3 Clases de Acceso a Datos (DAO)

Para poder acceder a la base de datos de forma sencilla, se han establecido 3 clases, correspondiendo a diferentes niveles de la aplicación, que acceden a la base de datos, y son llamadas a través del controlador para poder crear los objetos y la lógica de la plataforma correctamente.

Estas clases y todos sus métodos son estáticos, ya que carecería de sentido crear una nueva instanciación para cada vez que deseemos llamar a una de estas clases.

7.3.1 Métodos Interesantes

Algunos métodos interesantes para la obtención de datos son los siguientes:

conecta()

```php
private const HOST = 'localhost';
private const USER = 'root';
private const PASS = '';
private const DATABASE = 'indieHat';

//Conexion a la base de datos
public static function conecta()
{
    $conexion = new PDO('mysql:host='.JuegosDao::HOST.';dbname='.JuegosDao::DATABASE , JuegosDao::USER, JuegosDao::PASS);
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    return $conexion;
}
```

Figura 7.5: Método conecta

El método conecta establece una conexión con la base de datos, devolviendo un objeto tipo PDO, que será llamado por el resto de métodos para hacer las consultas correspondientes.

juegosGenero()

```php
//Devolvemos un array con los juegos de cada genero
public static function juegosGenero($genero)
{
    $conexion = self::conecta();

    $consultaJuego = "SELECT * FROM juego WHERE genero = ?";
    $queryJuego = $conexion->prepare($consultaJuego);
    $queryJuego->bindParam(1, $genero);
    $queryJuego->execute();

    //Obtenemos todos los juegos con sus reviews
    $arrJuegos = [];
    while($filaJuego = $queryJuego->fetch())
    {
        $arrJuegos[] = new Juego($filaJuego['idJuego'],$filaJuego['titulo'],JuegosDao::getEstudio($filaJuego['idEstudio']),$filaJuego['genero'],$filaJuego['precio'],$filaJuego
        ['descripcion'],$filaJuego['direccion'],$filaJuego['fechaLanzamiento'],[], $filaJuego['imagenPortada'], $filaJuego['imagenMuestra'], $filaJuego['videoMuestra']);
    }

    foreach($arrJuegos as $juego)
    {
        $consultaReview = "SELECT * FROM valoraciones
        INNER JOIN usuario USING(idUsu)
        WHERE destino = ?";
        $queryReview = $conexion->prepare($consultaReview);
        $queryReview->bindParam(1, $juego->id);
        $queryReview->execute();

        while($filaReview = $queryReview->fetch())
        {
            $juego->valoraciones[] = new Valoracion($filaReview['idValoracion'],new Usuario($filaReview['idUsu'], $filaReview['nombreUsu'], $filaReview['nombre'], $filaReview
            ['apellidos'], $filaReview['email']), $filaReview['puntuacion'], $filaReview['descripcion']);
        }
    }
    return $arrJuegos;
}
```

Figura 7.6: Método juegos Genero

Este método hace una búsqueda de todos los juegos de un determinado género, devolviendo un array de objetos tipo Juego para su posterior manipulación desde la vista. Esta estructura de método se repite a lo largo de toda la aplicación en métodos similares como juegosEstudio, o getMensajesLeidos.

buscador()

```php
//Devuelve un array de objetos tipo objeto, buscando por nombre, por estudio y por genero (Sin repetir juegos)
public static function buscador($busqueda)
{
    $conexion = self::conecta();
    //Sacamos los juegos por titulo
    $busqueda = "%$busqueda%";
    $query = $conexion->prepare("SELECT idJuego FROM juego WHERE titulo like :name ");
    $query->bindParam(':name', $busqueda);
    $query->execute();
    $arrMaestro = [];
    while($fila = $query->fetch())
    {
        $arrMaestro[] = JuegosDao::getJuego($fila['idJuego']);
    }

    //Sacamos los juegos por genero
    $busqueda = "%$busqueda%";
    $query = $conexion->prepare("SELECT idJuego FROM juego
    INNER JOIN genero ON idGenero = genero WHERE genero.nombre like :name ");
    $query->bindParam(':name', $busqueda);
    $query->execute();
    while($fila = $query->fetch())
    {
        $flag = false;
        foreach($arrMaestro as $juegoAux)
        {
            if($juegoAux->id == $fila['idJuego'])
            {
                $flag = true;
            }
        }
        if(!$flag)
        {
            $arrMaestro[] = JuegosDao::getJuego($fila['idJuego']);
        }
    }
```

```php
    //Sacamos los juegos por estudio
    $busqueda = "%$busqueda%";
    $query = $conexion->prepare("SELECT idJuego FROM juego
    INNER JOIN perfil_desarrollo ON idEstudio = idPerfil WHERE  perfil_desarrollo.nombre like :name ");
    $query->bindParam(':name', $busqueda);
    $query->execute();
    while($fila = $query->fetch())
    {
        $flag = false;
        foreach($arrMaestro as $juegoAux)
        {
            if($juegoAux->id == $fila['idJuego'])
            {
                $flag = true;
            }
        }
        if(!$flag)
        {
            $arrMaestro[] = JuegosDao::getJuego($fila['idJuego']);
        }
    }
    return $arrMaestro;
}
}
```

Figura 7.7: Buscador de Juegos

Este método es uno de los más interesantes de toda la plataforma. Realiza una consulta de todos los juegos cuyo título coincide con la búsqueda y los añade al array de respuesta. Acto seguido hace lo mismo con los juegos cuyo género coincide, y controla que no hayan sido anteriormente introducidos para evitar repetidos, ya que podría darse el caso que la palabra buscada este incluida tanto en el nombre del estudio como del género (entre otros). Finalmente hace el mismo procedimiento añadiendo las búsquedas que coincidan con el nombre del estudio desarrollador, para posteriormente devolver el array.

getEstudioFromUser()

```php
//Devolvemos un objeto tipo estudio, dada una id de tipo USUARIO
public static function getEstudioFromUser($id)
{
    $conexion = self::conecta();
    $consulta = "SELECT idUsu, nombreUsu AS nickName, usuario.nombre AS nombreUsu, usuario.apellidos AS apellidosUsu, usuario.email AS emailUsu,
    perfil_desarrollo.idPerfil AS idDes, perfil_desarrollo.nombre AS nombreDes, perfil_desarrollo.apellidos AS apellidosDes,
    perfil_desarrollo.nifCif AS nifDes, esEmpresa AS empresaDes, idPais, paises.nombre AS nombrePais, annoCreacion
    FROM perfil_desarrollo
    INNER JOIN usuario ON usuario.idUsu = perfil_desarrollo.idUsuario
    INNER JOIN paises ON perfil_desarrollo.pais = paises.idPais
    WHERE idUsu = ?";

    $query = $conexion->prepare($consulta);
    $query->bindParam(1, $id);
    $query->execute();

    $fila = $query->fetch();
    $resDesarrollador = new Desarrollador($fila['idUsu'], $fila['nickName'], $fila['nombreUsu'],$fila['apellidosUsu'],$fila['emailUsu'],$fila['idDes'],$fila['nombreDes'],$fila
    ['apellidosDes'],$fila['nifDes'],$fila['empresaDes'], new Pais($fila['idPais'],$fila['nombrePais']),$fila['annoCreacion']);

    return $resDesarrollador;
}
```

Figura 7.8: Método getEstudioFromUser

Otro método clave, dada su importancia, es el método getEstudioFromUser, que nos permite obtener un perfil de Desarrollador, dada su id de usuario. Esto es especialmente útil dado que la id del usuario es guardada en la sesión, y con este método podemos obtener acceso a sus datos de desarrollador, tales como mensajes, etc.

cambiaPassword()

```php
//Actualiza la contraseña del usuario
public static function cambiaPassword($passVieja, $passNueva, $user)
{
    $conexion = self::conecta();

    $consultaPass = "SELECT idUsu FROM usuario WHERE idUsu = ? AND password = ?";

    $queryPass = $conexion->prepare($consultaPass);
    $queryPass->bindParam(1, $user);
    $queryPass->bindValue(2, md5($passVieja));
    $queryPass->execute();

     if($queryPass->fetchColumn() > 0)
     {
        $consulta = "UPDATE usuario SET password = ? WHERE idUsu =  ?";

        $query = $conexion->prepare($consulta);
        $query->bindValue(1, md5($passNueva));
        $query->bindParam(2, $user);

        if($query->execute())
        {
            return ["error"=>0, "descripcion" => "Contraseña actualizada!"];
        }
        else
        {
            return ["error"=>1, "descripcion" => "Ha ocurrido un error. Intentelo mas tarde."];
        }
    }
    else
    {
        return ["error"=>1, "descripcion" => "La contraseña actual no es correcta"];
    }
}
```

Figura 7.9: Método cambiaPassword

Este método comprueba que la contraseña antigua introducida sea correcta, y solo en ese caso procede a actualizar la contraseña por la deseada. La forma de devolver los datos se repite a lo largo de toda la aplicación, devolviéndose en forma de un array asociativo con un código de error y una descripción, lo que es particularmente útil para su tratado posterior desde un método AJAX.

getJuego()

```php
//Devolvemos un objeto tipo Juego dada su id en base de datos
public static function getJuego($id)
{
    $conexion = self::conecta();

    $consultaJuego = "SELECT * FROM juego WHERE idJuego = ?";
    $queryJuego = $conexion->prepare($consultaJuego);
    $queryJuego->bindParam(1, $id);
    $queryJuego->execute();

    $filaJuego = $queryJuego->fetch();

    $juegoRes = new Juego($filaJuego['idJuego'],$filaJuego['titulo'],JuegosDao::getEstudio($filaJuego['idEstudio']),$filaJuego['genero'],$filaJuego['precio'],$filaJuego['descripcion']
    ,$filaJuego['direccion'],$filaJuego['fechaLanzamiento'],[], $filaJuego['imagenPortada'], $filaJuego['imagenMuestra'], $filaJuego['videoMuestra']);


    $consultaReview = "SELECT * FROM valoraciones
    INNER JOIN usuario USING(idUsu)
    WHERE destino = ?";
    $queryReview = $conexion->prepare($consultaReview);
    $queryReview->bindParam(1, $juegoRes->id);
    $queryReview->execute();

    while($filaReview = $queryReview->fetch())
    {
        $juegoRes->valoraciones[] = new Valoracion($filaReview['idValoracion'],new Usuario($filaReview['idUsu'], $filaReview['nombreUsu'], $filaReview['nombre'], $filaReview
        ['apellidos'], $filaReview['email']), $filaReview['puntuacion'], $filaReview['descripcion']);
    }

    return $juegoRes;
}
```

Figura 7.10: Método getJuego

Otro método clave, y cuya estructura es repetida en diferentes métodos a lo largo de toda la plataforma es el método getJuego, el cual nos devuelve un objeto (en este caso Juego), dada una id almacenada en base de datos. Este caso es particularmente interesante porque además requiere la adición de más objetos dentro de sí dadas las características del objeto Juego, tales como Valoracion, Usuario o Estudio.

addValoracion()

```php
//Insertamos una valoracion en base de datos
public static function addValoracion($idUsu, $destino, $puntuacion, $descripcion)
{
    require_once('../clases/genero.class.php');
    require_once('../clases/juego.class.php');
    require_once('../clases/usuario.class.php');
    require_once('../clases/valoracion.class.php');
    $conexion = self::conecta();
    //Agregamos la valoracion
    $consulta = "INSERT INTO valoraciones (idUsu, destino, puntuacion, descripcion) VALUES (?,?,?,?);";
    $query = $conexion->prepare($consulta);
    $query->bindParam(1, $idUsu);
    $query->bindParam(2, $destino);
    $query->bindParam(3, $puntuacion);
    $query->bindParam(4, $descripcion);
    $query->execute();

    $consultaId = "SELECT LAST_INSERT_ID() AS id;";
    $queryId = $conexion->prepare($consultaId);
    $queryId->execute();
    $filaLastId = $queryId->fetch();
    //Obtenemos los datos necesarios para introducir la valoracion por jquery
    $valRes = juegosDao::getValoracion($filaLastId['id']);

    $arrRes['id'] = $valRes->id;
    $arrRes['puntuacion'] = $valRes->puntuacion;
    $arrRes['descripcion'] = $valRes->descripcion;
    $arrRes['usuario'] = $valRes->usuario->nombre;

    //Devolvemos el array que devolvera los datos para insertar la valoracion por jQuery
    return $arrRes;
}
```

Figura 7.11: Método addValoracion

El método addValoracion, incluye la inclusión previa de las clases a tratar, dado que desde el AJAX que se llama no las está recibiendo, por la propia estructura de la página. Es clave la obtención del id obtenido de la inserción realizada, para poder obtener la valoración y devolver los datos en forma de array.

borraValoracion()

```php
//Borramos una valoracion de la BD
public static function borraValoracion($id)
{
    $conexion = self::conecta();
    //Agregamos la valoracion
    $consulta = "DELETE FROM valoraciones WHERE idValoracion = ?;";
    $query = $conexion->prepare($consulta);
    $query->bindParam(1, $id);
    if($query->execute())
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Figura 7.12: Método borraValoracion

La aplicación no tiene más métodos de borrado dado que no hay necesidad de borrar más registros que las valoraciones. El método procede al borrado del registro en la Base de Datos, y devuelve un booleano para su posterior manejo desde el método AJAX que lo llama.

7.3.2 Llamadas a los objetos DAO

Dado que todas las llamadas a los objetos DAO serán o bien desde el controlador, o bien desde el entorno cliente, de forma dinámica (vía AJAX), se estudian previamente los casos en los que se necesitará y se preparan los métodos que deben estar "a la escucha" para dichas llamadas vía AJAX.

Estos métodos se establecen a la escucha forzando una comprobación cada vez que se llame al archivo del objeto, y en caso de existir un POST tipo función, que será el que acompañará a todas las llamadas de AJAX (además de los parámetros pertinentes), se establecerá que función debe ser llamada.

```php
//Filtro para las diferentes llamadas al objeto de acceso a la BD
if(isset($_POST['funcion']))
{
    switch($_POST['funcion'])
    {
        case "login":
            $res = UserDao::login($_POST['user'], $_POST['pass']);
            echo json_encode($res);
        break;
        case "register":
            $res = UserDao::registro($_POST['user'], $_POST['pass'],$_POST['nombre'], $_POST['apellidos'],$_POST['email']);
            echo json_encode($res);
        break;

        case "regDes":
            $res = UserDao::addDesarrollador($_POST['nomDes'],$_POST['apeDes'],$_POST['empDes'],$_POST['nifDes'],$_POST['paisDes'],$_POST['annoDes'],$_POST['usuario']);
            echo json_encode($res);
        break;
        case "cambiaPass":
            $res = UserDao::cambiaPassword($_POST['passVieja'],$_POST['passNueva'],$_POST['user']);
            echo json_encode($res);
        break;
    }
}
```

Figura 7.13: Filtro de funciones en los objetos DAO

7.4 Controlador

La plataforma consta de dos páginas que hacen la función de controlador. Dichas páginas establecen un filtro al principio, que controlan si hay sesión creada del usuario o no (si esta logueado). En caso positivo podrán continuar, en caso negativo serán redireccionados al login.

A continuación, hay otro control, en este caso del POST “pagina”, que controla a que página están accediendo en cada momento, incluyendo la vista correspondiente.

```php
<?php
session_start();
header('Content-Type: text/html; charset=utf8');
require_once('clases/genero.class.php');
require_once('clases/juego.class.php');
require_once('clases/usuario.class.php');
require_once('clases/valoracion.class.php');
require_once('clases/desarrollador.class.php');
require_once('clases/pais.class.php');
require_once('modelo/userDao.php');
require_once('modelo/juegosDao.php');

if(isset($_SESSION['user']))
{
    if(isset($_POST['pagina']))
    {
        switch($_POST['pagina'])
        {
            case 'detalle':
                include('vistas/vistaDetalle.php');
            break;
            case 'genero':
                include('vistas/vistaGenero.php');
            break;
            case 'busqueda':
                include('vistas/vistaBusqueda.php');
            break;
            case 'config':
                include('vistas/vistaConfiguracion.php');
            break;
            case 'logout':
                session_destroy();
                include('vistas/vistaLogin.php');
            break;
            default:
                include('vistas/vistaJuegos.php');
            break;
        }
    }
    else
    {
        include('vistas/vistaJuegos.php');
    }
}
else
{
    include('vistas/vistaLogin.php');
}
```

Figura 7.14: Index.php

```php
if(isset($_SESSION['user']))
{

    if(isset($_POST['pagina']))
    {
        switch($_POST['pagina'])
        {
            case 'busqueda':
                include('vistas/vistaBusquedaEstudio.php');
            break;
            case 'mensajes':
                include('vistas/vistaMensajes.php');
            break;
            case 'detalle':
                include('vistas/vistaDetalleEstudio.php');
            break;
            case 'logout':
                session_destroy();
                include('vistas/vistaLogin.php');
            break;
            default:
                if(UserDao::esDesarrollador($_SESSION['id']))
                {
                    include('vistas/vistaDesarrollador.php');
                }
                else
                {
                    include('vistas/vistaRegistro.php');
                }
            break;
        }
    }
    else
    {
        if(UserDao::esDesarrollador($_SESSION['id']))
        {
            include('vistas/vistaDesarrollador.php');
        }
        else
        {
            include('vistas/vistaRegistro.php');
        }
    }
}
else
{
    include('vistas/vistaLogin.php');
}
```

Figura 7.15: Desarrollo.php

7.5 Programación desde el Entorno Cliente

Toda la programación necesaria llevada desde el lado del cliente ha sido hecha a través de JavaScript, o jQuery en su defecto.

Para el desarrollo de esta aplicación ha sido clave la inclusión de la librería jQuery.redirect, para poder redireccionar desde el cliente, enviando POST a las paginas, y así permitiéndonos controlar a que función del controlador queremos acceder en cada momento.

```
$.redirect('index.php', {'pagina': 'principal'});
```

Figura 7.16: jQuery Redirect

7.5.1 Funciones interesantes

Función de login

```
//Funcion para acceder a la pagina
$(".login-form").submit(function(e)
{
    $("#errorLog").text("");
    e.preventDefault();
    var user = $("#userLogin").val();
    var pass = $("#passLogin").val();
    $.ajax({
        type: "POST",
        url: "modelo/userDao.php",
        dataType: "json",
        data:
            {
                "funcion":"login",
                "user":user,
                "pass":pass
            },
        success : function(data)
        {
            if(data)
            {
                $.redirect('index.php', {'pagina': 'principal'});
            }
            else
            {
                $("#errorLog").text("Usuario o Contraseña incorrecto/s");
                $("#userLogin").val('');
                $("#passLogin").val('');
            }
        }
    });
});
```

Figura 7.17: Login en jQuery

Al hacer submit en el formulario del login, se abre un AJAX que llamará a la función de login anteriormente explicada. En caso negativo, mostrara los mensajes de error correspondientes.

En caso positivo, seremos redirigidos a la página index, con el post que nos llevará a la función principal.

Funciones de redirección con POST

```
//----------------------FUNCIONES DE REDIRECCION A CONTROLADOR CON POST----------------------
//Detalle de la ficha del videojuego
$(".juegoDetalle").click(function()
{
    $.redirect('index.php', {'pagina': 'detalle', 'juego':$(this).attr('id')});
});
//Logout de la pagina
$("#logout").click(function()
{
    $.redirect('index.php', {'pagina': 'logout'});
});
//Pagina de configuracion
$("#config").click(function()
{
    $.redirect('index.php', {'pagina': 'config'});
});
//Pagina de genero concreto (click en el titulo del genero)
$(".enlaceGenero").click(function()
{
    $.redirect('index.php', {'pagina': 'genero', 'genero':$(this).attr('id')});
});
```

Figura 7.18: Redirecciones con POST

Añadir valoración

Esta es probablemente la función de jQuery más compleja de la aplicación, ya que posterior a su llamada, tiene que añadir la valoración establecida, y borrar el campo de inserción anteriormente usado. Todo esto se hace a través de elementos de JavaScript para manipular el DOM de la página.

```
$(".botVal").click(function()
{
    var destino = $("#destino").val();
    var idUsu = $("#idUsu").val();
    var descripcion = $("#descripcion").val();
    var puntuacion = $("#puntuacion").val();

    if($("#descripcion").val())
    {
        $.ajax({
            type: "POST",
            url: "modelo/juegosDao.php",
            dataType: "json",
            data:
            {
                "funcion":"addVal",
                "destino":destino,
                "idUsu":idUsu,
                "descripcion":descripcion,
                "puntuacion":puntuacion
            },
            success:function(data)
            {

                //Creamos el div Padre, con el id y el estilo necesario
                var divPadre = document.createElement("div");
                divPadre.classList.add("titSeccion");
                divPadre.style.border = "1px #BABABA solid";
                divPadre.style.padding = "20px";
                divPadre.style.textAlign = "justify";
                divPadre.setAttribute('data', "id: '"+data.id+"'");
                divPadre.setAttribute('id', "padre"+data.id);

                //Boton de eliminar
                var enlaceBorrar = document.createElement("a");
                enlaceBorrar.setAttribute('id', data.id);
                enlaceBorrar.classList.add("borraVal");
                enlaceBorrar.setAttribute("href", "javascript: void(0)");

                var borrarTxt = document.createTextNode("Eliminar");
                enlaceBorrar.append(borrarTxt);
```

Figura 7.19: Añadir Valoración I

```
//Creamos el evento dinamicamente para borrar sin recargar, una vez añadida la valoracion
enlaceBorrar.addEventListener('click',function(e)
{
    var id = $(this).attr('id');
    $.ajax({
        type: "POST",
        url: "modelo/juegosDao.php",
        dataType: "json",
        data:
        {
            "funcion":"delVal",
            "id":id,
        },
        success:function(data)
        {
            if(data)
            {
                $("#padre"+id).remove();
            }
        }
    });
});

//Creamos el titulo de usuario
var titUsuario = document.createElement("h5");
var usuTxt = document.createTextNode("Usuario: "+data.usuario);
titUsuario.append(usuTxt);
//Titulo de puntuacion
var titPuntuacion = document.createElement("h6");
var puntTxt = document.createTextNode("Puntuacion: "+data.puntuacion);
titPuntuacion.append(puntTxt);
//Descripcion
var desc = document.createElement("p");
var descTxt = document.createTextNode(data.descripcion);
desc.append(descTxt);

//Terminamos de crear el div añadiendo los elementos
divPadre.append(enlaceBorrar);
divPadre.append(titUsuario);
divPadre.append(titPuntuacion);
divPadre.append(desc);

//Lo añadimos al final de las valoraciones
$("#valoracionesDiv").append(divPadre);

//Borramos el formulario para que borre cosas el usuario
$("#valoracionUsuario").remove();
}
```

Figura 7.20: Añadir Valoración II

Función para leer un mensaje

```
//Marcar mensaje como leido
$(".leerMensaje").click(function(e)
{
    var idMensaje = $(this).attr('id');

    $.ajax({
        type: "POST",
        url: "modelo/mensajeriaDao.php",
        dataType: "json",
        data:
        {
            "funcion":"leerMensaje",
            "idMensaje":idMensaje
        },
        success:function(data)
        {
            if(data)
            {
                //Borramos el mensaje de leidos
                var miMensaje = $("#padre"+idMensaje);
                $("#padre"+idMensaje).remove();
                //Lo añadimos a no leidos como primer hijo
                $("#mensajesDivLeidos").prepend("<br>");
                $("#mensajesDivLeidos").prepend(miMensaje);
                //Borramos el enlace de marcar mo leido
                $("#"+idMensaje).remove();

            }
            else
            {
                alert("Ha ocurrido un error.")
            }
        }
    });
});
;
```

Figura 7.21: Marcar como leído

Esta función funciona de una forma similar a la anterior. Cuando se establece la conexión con el servidor, y la función se ejecuta correctamente, se guarda el elemento pulsado, para luego borrarle, y añadirle en la otra lista (Mensajes leídos).

7.6 Vistas

A continuación, se describen algunas de las vistas más interesantes, ofrecidas por la página.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<link rel="stylesheet" href="css/styleLogin.css"/>
<script src="js/mainLogin.js"></script>
<script src="js/funcionesLogin.js"></script>
<script src="js/jquery.redirect.js"></script>
```

Figura 7.22: Inclusión en vistas

Todas las vistas cuentan con los scripts de inclusión de jQuery, los archivos JS que llevan la programación del lado del cliente (llamadas AJAX, etc.), y la librería jQuery.redirect. Esta última es sumamente importante que sea la incluida después del script de jQuery, para evitar errores.

Vista Login:

```
<div class="login-page">
    <div class="form">
        <div style="margin: 0 auto;"><img src="img/mainLogo.png" height="215px"></div>

        <form action="index.php" class="register-form">
            <input type="text" id="userReg" placeholder="Nombre de Usuario" required/>
            <input type="password" id="passReg1" placeholder="Contraseña" required/>
            <input type="password" id="passReg2" placeholder="Confirmar contraseña" required/>
            <input type="text" id="emailReg" placeholder="Correo electrónico" required/>
            <input type="text" id="nombreReg" placeholder="Nombre" required/>
            <input type="text" id="apellidosReg" placeholder="Apellidos" required/>
            <button id="registrar">Registrar</button>
            <p class="message">¿Ya tienes cuenta?<a href="#">Acceder</a></p>
            <p id="errorReg" style="color:red;"></p>
        </form>


        <form class="login-form">
            <input type="text" id="userLogin" placeholder="Nombre de Usuario" required/>
            <input type="password" id="passLogin" placeholder="Contraseña" required/>
            <button id="login">Acceder</button>
            <p class="message">¿No tienes cuenta? <a href="#">Registrate!</a></p>
            <p id="errorLog" style="color:red;"></p>
        </form>
    </div>
</div>
```

Figura 7.23: Vista Login

Esta vista muestra dos formularios diferenciados de Login y Registro, los cuales son alternados a través de un evento jQuery.

Vista Juegos:

La sección principal de la página, que ofrece una cartelera de juegos ordenada por cada género. Se recorren todos los géneros con juegos registrados devueltos por la clase de JuegosDao. Una vez obtenido dicho array, se recorre otra vez, devolviendo 4 juegos de cada género, con el método JuegosGenero de la clase JuegosDao.

```php
<?php
foreach(JuegosDao::getGeneros() as $genAux)
{
    ?>
<section class="tournaments-section spad" data-setbg="img/pattern.png">
    <div class="container divJuegos">
        <div class="section-title titSeccion">
            <a class="enlaceGenero" id="<?php echo $genAux->id; ?>"  href="#"><h3><?php echo $genAux->nombre;?></h3></a>
        </div>
        <div class="row">
            <?php
                $i=0;
                foreach(JuegosDao::juegosGenero($genAux->id) as $juego)
                {
                    if($i<=3)
                    {
                        ?>
                        <div class="col-lg-3 col-md-6" style="margin-bottom:50px;">
                            <div class="review-item">
                                <div class="review-cover set-bg" data-setbg="imgJuegos/<?php echo $juego->imagenPortada; ?>">
                                    <div class="score yellow"><?php echo $juego->notaMedia();?></div>
                                </div>
                                <div class="review-text">
                                    <h5><?php echo $juego->nombre;?></h5>
                                    <p><?php echo substr($juego->descripcion,0,100)."...";?></p><br>
                                    <button id="<?php echo $juego->id; ?>" class="site-btn btn-sm juegoDetalle">Ver juego</button>
                                </div>
                            </div>
                        </div>
                        <?php
                    }
                    $i++;
                }
            ?>
        </div>
    </div>
</section>
    <?php
}
?>
```

Figura 7.24: Vista Juegos

Vista detalle:

Esta vista se divide en dos secciones.

La primera corresponde a los datos del juego en cuestión.

```php
<!-- Header section -->
<header class="header-section">
    <?php $juegoDetalle = JuegosDao::getJuego($_POST['juego']); ?>
    <div class="container">
```

Figura 7.25: Vista Detalle I

Se crea una instanciación del mismo, con los datos recibidos por POST, y se imprimen en su ficha.

```
<ul class="community-post-list">
    <li>
        <div class="section-title titSeccion">
            <h1><?php echo $juegoDetalle->nombre;?></h1> <br>
            <a href="index.php"><h5>Volver</h5><a href="index.php"></a>

            <br>
            <div class="review-item">
                <img src="imgJuegos\<?php echo $juegoDetalle->imagenMuestra;?>" height="350" width="700">
            </div>
            <br><p><?php echo $juegoDetalle->descripcion; ?></p><br>
            <div class="review-item">
                <img src="imgJuegos\<?php echo $juegoDetalle->videoMuestra;?>" height="350" width="700">
            </div>
        </div>

            <!-- <iframe style="display:block; margin: 0 auto;" width="720" height="480" src="<?php echo $juegoDetalle->videoMuestra;?>">
            </iframe> -->

        <br><br>
        <h3 class="fw-title">Precio: <?php echo $juegoDetalle->precio;?>€ <button style="float:right;" class="site-btn">Comprar</button></h4>
    </div>
    </li>
</ul>
```

Figura 7.26: Vista detalle II

La segunda sección, es la sección de valoraciones, donde se ofrece una lista de las valoraciones emitidas, y un cuadro para insertar una valoración (solo en el caso de que el usuario no lo haya hecho anteriormente). En caso de existir una valoración ya emitida por el usuario, se mostrará el enlace de Eliminar en la valoración.

```
<div class="container">
    <ul class="community-post-list">
        <li>
            <div class="section-title titSeccion">
                <h2>Valoraciones</h2><br>
                    <div id="valoracionesDiv" style="max-height: 500px; overflow-y: auto; padding:20px; background-color: rgba(100,100,100,0.1);">
                        <?php
                        foreach($juegoDetalle->valoraciones as $valor)
                        {
                            if($valor->usuario->id == $_SESSION['id'])
                            {
                                $flagValoracion = true;
                            }
                            echo "<div id='padre".$valor->id."' data-idVal='".$valor->id."' class='titSeccion' style='border:1px #BABABA solid; padding: 20px; text-align:
                            justify;'>";
                            if(isset($flagValoracion))
                            {
                                echo "<a id='".$valor->id."' class='borraVal'href='javascript: void(0)'>Eliminar</a>";
                            }
                            echo "<h5>Usuario: ".$valor->usuario->usuario."</h5>";
                            echo "<h6>Puntuación: ".$valor->puntuacion."</h6>";
                            echo "<p>".$valor->descripcion."</p>";
                            echo "</div><br>";
                        }
                        ?>
                    </div>
                <br>
                <?php
                if(!isset($flagValoracion))
                {
                ?>
                    <div id="valoracionUsuario" class='titSeccion' style='border:1px #BABABA solid; padding: 20px;'>
                        <h4>Valora este juego</h4><br>
                        <textarea name="descripcion" id="descripcion" cols="90" rows="5"></textarea>
                        <br><br>
                        <input id="destino" type="hidden" value="<?php echo $juegoDetalle->id;?>">
                        <input id="idUsu" type="hidden" value="<?php echo $_SESSION['id'];?>">
                        <h6>Puntuacion:<input onkeydown="return false" step="0.5" type="number" min="0" max="5" id="puntuacion" value ="0" style="margin-left:10px;"></
                        h6><br>
                        <button class="site-btn btn-sm botVal">Valorar</button>
                    </div>
                <?php
                }
                ?>
            </div>
        </li>
    </ul>
</div>
```

Figura 7.27: Vista detalle III

Vista Mensajes:

Esta vista muestra todos los mensajes recibidos por el usuario, agrupados en dos listas: Leídos y No Leídos. Del mismo modo que en la sección de valoraciones, se muestra un formulario para enviar un mensaje al usuario deseado.

```
<div class="section-title titSeccion">
    <h2>Mensajes</h2><br>
    <br>
    <a href="desarrollador.php"><h5>Volver</h5></a><br>
    <h5>No leidos</h5>
    <div id="mensajesDiv" style="border:1px #BABABA solid; padding: 20px; max-height: 300px; overflow-y: auto; padding:20px; background-color: rgba(100,100,
    100,0.1);">
            <?php
            foreach(MensajeriaDao::getMensajesNoLeidos($_POST['estudio']) as $mensaje)
            {
                echo "<div id='padre".$mensaje->idMensaje."' class='titSeccion' style='border:1px #BABABA solid; padding: 20px; text-align: justify;'>";
                echo "<a id='".$mensaje->idMensaje."' class='leerMensaje' href='javascript: void(0)'>Marcar como leido</a>";

                echo "<h5>Usuario: ".$mensaje->usuario->usuario."</h5>";
                echo "<p>".$mensaje->mensaje."</p>";
                echo "</div><br>";
            }
            ?>
        </div>
    <br>
    <h5>Leidos</h5>
    <div id="mensajesDivLeidos" style="border:1px #BABABA solid; padding: 20px; max-height: 300px; overflow-y: auto; padding:20px; background-color: rgba(100,
    100,100,0.1);">
            <?php
            foreach(MensajeriaDao::getMensajesLeidos($_POST['estudio']) as $mensaje)
            {
                echo "<div id='padre".$mensaje->idMensaje."' class='titSeccion' style='border:1px #BABABA solid; padding: 20px; text-align: justify;'>";

                echo "<h5>Usuario: ".$mensaje->usuario->usuario."</h5>";
                echo "<p>".$mensaje->mensaje."</p>";
                echo "</div><br>";
            }
            ?>
        </div>
</div>
        <div id="mensajeUsu" class='titSeccion' style='border:1px #BABABA solid; padding: 20px;'>
            <h4>Nuevo mensaje</h4><br>
            <h6>Destino:<input type="text" id="destino"  style="margin-left:10px;"></h6><br>
            <textarea name="descripcion" id="descripcion" cols="90" rows="5"></textarea>
            <br><br>
            <input id="idUsu" type="hidden" value="<?php echo UserDao::getEstudioFromUser($_SESSION['id'])->idPerfil;?>">

            <button class="site-btn btn-sm enviaMensaje">Enviar</button>
        </div>
</div>
```

Figura 7.28: Vista Mensajes

8

Puesta en marcha y explotación

Para la puesta en marcha del proyecto Indie Hat, será fundamental pasar un control de seguridad dada la delicadeza de los datos a tratar de cada usuario.

8.1 Control de seguridad

Se comprobará todo el código de la plataforma buscando posibles debilidades contra las principales amenazas.

- Almacenamiento de **contraseñas**:

Se comprobará que todas las contraseñas estén correctamente almacenadas en base de datos, con un **encriptamiento md5**.

- Protección contra **MySQL injection**:

Aunque las contraseñas estén protegidas con el encriptado md5, existen **diccionarios** y diversos tipos de aplicaciones, que hacen que conseguir una clave md5 sea relativamente fácil, de modo que es importante protegerse de todo tipo de inyecciones de código, ya que la plataforma está accediendo directamente a la base de datos. Los aspectos a comprobar son:

  - Todas las **conexiones** a base de datos serán realizadas a través de **PDO**.
  - Todas las consultas realizadas, serán a través de **consultas preparadas** (***PDO::prepare***), para así evitar la **inyección** de **código** no deseado.

- Protección contra **XSS** (Cross-site scripting) y **CSRF** (Cross-site request forgery):

Otra amenaza es la inserción de código no deseado desde el lado del cliente, dado que es una plataforma ejecutada en el navegador del mismo. Las medidas a tomar serán:

  - Asegurarse de que la **cabecera** de **protección** está activada en el servidor (en nuestro caso **Apache**): ***header always set x-xss-protection "1; mode=block"***
  - Todos los scripts de JS que reciban información introducida por el cliente (mensajes, valoraciones, etc.), serán pasados por un **filtro** que eliminará la etiqueta *"<script>"* de forma **recursiva**, controlando tanto **mayúsculas**, como **minúsculas**, como **caracteres especiales**.

- Protección contra ***Man in the Middle***:

Una amenaza tipo *Man in the Middle* consiste en una intercepción en la comunicación entre el emisor y receptor, de modo que el atacante pueda acceder a la información enviada, y luego reenviarla al servidor en cuestión, de modo que no se notifique dicho ataque.

- Para protegerse contra este tipo de ataques, todos los scripts que se comuniquen con el servidor, irán acompañados de una función que genere un ***token*** conformado por 10 dígitos generados **aleatoriamente** y encriptados por **md5**. Dicho **token** será generado por el servidor, y enviado por un campo **tipo hidden** al usuario, a la vez que es guardado en el mismo, de modo que cuando el cliente envíe cualquier tipo de petición al servidor, este comprobará si el **token** coincide con el guardado, asegurando así la autenticidad del usuario y el mensaje.

8.2 Explotación

Una vez pasado el análisis de seguridad (que será ejecutado de forma periódica en toda la plataforma), se pasará a la fase de implementación en un servidor real.

8.2.1 Elección de servidor, y configuración.

Se procederá a la elección de un servidor de hosting donde situar nuestra plataforma. En este caso se ha optado por 1&1 IONOS, dado que ofrece un paquete económico que se ajusta a las necesidades de la plataforma (Ubuntu server 18.04, Bases de Datos MySQL, PHP 7.2).

Los pasos a seguir serán:

Base de datos

Se importará la base de datos previamente descrita.

Se crearán los usuarios con los permisos necesarios para la gestión de la página, ya que en ningún caso esta deberá acceder con el usuario root.

Servidor Web

Se cambiarán las opciones necesarias en los archivos de configuración de Apache (principalmente el archivo *httpd.conf*). Por ejemplo: Habilitar la cabecera de protección contra XSS previamente mencionada.

Nos aseguraremos que, en el DNS, la url del dominio de Indie hat, apunte al directorio donde desplegaremos nuestra plataforma.

Del mismo modo que con las opciones de *httpd.conf*, nos aseguraremos de que la configuración de PHP (*php.ini*) cumpla los requisitos de nuestra plataforma, dado que muchas veces la configuración por defecto de los hosting difiere de la que se tiene en un servidor local.

Pasos previos a la subida

Cambiaremos las conexiones de la Base de Datos, a las nuevas (facilitadas por 1&1).

Obtención de la pasarela de pago, e implementación en la plataforma. (Esta pasarela será facilitada por una entidad bancaria, que será decidida en el momento de la creación de la empresa, dado que las condiciones de dichas entidades varían enormemente a lo largo del tiempo, y debe ser algo tomado en cuenta en el momento de la creación).

8.2.2 Periodo de Beta-Testing

Una vez implementada la plataforma, se pasará al periodo de pruebas, o fase de beta, donde se contactará con los estudios (y usuarios) que deseen probar dicha plataforma antes de su lanzamiento oficial, informando de posibles errores previos a la etapa de producción.

8.2.3 Pruebas y Control de Calidad

Tabla 8.1: Caso 1

Caso de prueba: 1	**Requisito testeado: OBJ 2**
Descripción:	Comprobar el funcionamiento del sistema de registro (y cifrado de contraseña).
Prerrequisitos:	Ninguno.
Procedimientos:	Acceder al menú de registro de usuario, y ejecutar el registro de cuenta.
Resultado esperado:	La página debe controlar la existencia de usuarios con ese nombre, y en caso de no haber, se creará la cuenta, comprobando en la base de datos el encriptado md5.

Resultado obtenido:	Se controla que no haya usuarios con dicho nombre, y en caso de no haber, la cuenta se crea correctamente, encriptándose la contraseña en md5.

Tabla 8.2: Caso 2

Caso de prueba: 2	**Requisito testeado: OBJ 2**
Descripción:	Comprobar el funcionamiento del login, y los datos de sesión almacenados.
Prerrequisitos:	Tener cuenta creada.
Procedimientos:	Entrar a la página a través de la pantalla de login.
Resultado esperado:	La página debe restringir el acceso a no ser que se introduzca correctamente la combinación de usuario y contraseña. Una vez dentro, se deben almacenar en el Array de sesión, los datos del usuario guardados en la base de datos.
Resultado obtenido:	El proceso de login se ejecuta de forma correcta, y los datos de Sesión son creados correctamente, y borrados en el logout.

Tabla 8.3: Caso 3

Caso de prueba: 3	**Requisito testeado: OBJ 2**
Descripción:	Comprobar el proceso de creación de un perfil de desarrollador.
Prerrequisitos:	Tener cuenta de usuario normal.
Procedimientos:	Acceder al menú de desarrollador, e ingresar los datos correspondientes.
Resultado esperado:	El perfil se crea de forma correcta y se almacena en base de datos. Se permite el acceso al panel de desarrollador que antes estaba restringido.
Resultado obtenido:	El perfil de desarrollador se crea correctamente, y es almacenado en la base de datos. Se puede acceder al contenido de desarrollador que antes estaba restringido.

Tabla 8.4: Caso 4

Caso de prueba: 4	**Requisito testeado: OBJ 1**
Descripción:	Prueba de filtros del catálogo.
Prerrequisitos:	Tener cuenta de usuario estándar.
Procedimientos:	Acceder a la página, y comprobar los filtros de género establecidos en el catálogo de videojuegos.
Resultado esperado:	Los juegos deben mostrarse correctamente, agrupados por su género. Al pulsar el título del género, se mostrarán todos los juegos de dicho género.

Resultado obtenido:	Los juegos se muestran agrupados correctamente por su género. Al pulsar dicho título, se muestran todos los juegos almacenados en la Base de Datos.

Tabla 8.5: Caso 5

Caso de prueba: 5	**Requisito testeado: OBJ 1**
Descripción:	Prueba del buscador de videojuegos del catálogo.
Prerrequisitos:	Tener cuenta de usuario estándar.
Procedimientos:	Acceder a la página principal, y escribir en el buscador la palabra (o palabras) clave deseada.
Resultado esperado:	Al ingresar la palabra o palabras clave deseada, el buscador realiza la búsqueda deseada, buscando coincidencias en el género, título o estudio del juego.
Resultado obtenido:	El buscador funciona adecuadamente, en cualquiera de los casos, y sin repetirse ningún título

Tabla 8.6: Caso 6

Caso de prueba: 6	**Requisito testeado: OBJ 3**
Descripción:	Probar el sistema de mensajería interno
Prerrequisitos:	Tener cuenta de desarrollador.
Procedimientos:	Acceder al panel de desarrollador, acceder a la sección de mensajes, e introducir el mensaje y el usuario al que se desea enviar dicho mensaje.
Resultado esperado:	El proceso de enviar mensaje debería realizarse correctamente, y el desarrollador destino, debería recibir el mensaje en su bandeja de entrada. El tiempo desde que se envía el mensaje hasta que se recibe, debería ser inferior a 5 minutos. Se controlará antes del envío que dicho usuario exista.
Resultado obtenido:	El proceso de envío se realiza sin ningún problema. Cuando el desarrollador destino accede a su panel de desarrollador, puede ver el mensaje nuevo en su bandeja de entrada. Este proceso lleva un tiempo de espera menor a 1 minuto, aunque variará en función de los usuarios accediendo a la aplicación en ese momento. El control de la existencia del usuario se realiza de forma correcta.

8.3 Plan de Empresa

8.3.1 Ubicación

Se utilizará una oficina ya en uso, de forma gratuita, ubicada en:

Calle Prolongación Navas de Tolosa, 2, Bajo. 39400. Los Corrales de Buelna

8.3.2 Inmovilizado intangible

Propiedad industrial

Las tasas por servicios prestados por el Registro Central de la Propiedad Intelectual, están reguladas por el artículo 20 de la Ley 66/1997 de 30 de diciembre.

Inscripción de derechos en el Registro de la Propiedad Intelectual, coste de 13,20€. Para tener protección jurídica de la marca o nombre comercial, es necesario registrarla en la Oficina Española de Patentes y Marcas, la duración de la protección conferida es de diez años a partir de la fecha del depósito de la solicitud y pueden ser renovados indefinidamente.

Aplicaciones informáticas

Tabla 8.7: Aplicaciones informáticas

CONCEPTO	CARACTERISTICAS	CANTIDAD	PRECIO	IVA
LibreOffice	Software de Ofimática	1	0€	21%
GIMP	Software de edición de imagen	1	0€	21%
Avast Antivirus	Antivirus	1	0€	21%
TOTAL			**0€**	

8.3.3 Inmovilizado material

Instalaciones

La línea telefónica, y conexión por fibra, supone un coste total de 99.00€.

Mobiliario

Tabla 8.8: Mobiliario

CONCEPTO	CARACTERISTICAS	CANTIDAD	PRECIO	IVA
Mesa Escritorio		1	300€	21%
Silla Ergonómica de oficina		1	150,00€	21%
Estanterías			100,00€	21%
TOTAL			**550,00€**	

Equipos informáticos

Tabla 8.9: Equipos informáticos

CONCEPTO	CARACTERISTICAS	CANTIDAD	PRECIO	IVA
Ordenador	MSI GL63 8RD-407XES Intel Core i7-8750H/8GB/512GB SSD/GTX1050Ti/15.6".	1	957,99€	21%
Teléfono Inalámbrico	Ideus 13T	1	60,00€	21%
TOTAL			**1.017,99€**	

8.4 Presupuesto detallado

Se puede observar que se ahorra una gran cantidad de dinero en las aplicaciones informáticas, ya que todo el software que se utilizará, se distribuye de forma gratuita.

Tabla 8.10: Inversiones

INVERSIONES	% AMORT	AÑO 1	AÑO 2	AÑO 3
Inmovilizado Material				
Terrenos				
Instalaciones y Maquinaria	10%			
Mobiliario y Utillaje	10%	550,00		
Elementos de transporte	15%			
Equipo proceso de información	25%	1.017,99		
Otro inmovilizado material	10%			
Total Inmovilizado Material		**1.567,99**	**0,00**	**0,00**
Inmovilizado Intangible				
Investigación y Desarrollo	20%			
Patentes, licencias y marcas		89,81		
Aplicaciones informáticas	26%	0,00		
Derechos de traspaso				
Canon franquicia				
Total Inmovilizado Intangible		**89,81**	**0,00**	**0,00**
Inversiones Financieras		0,00	0,00	0,00
Amortización Inmovilizado Material		**309,50**	**309,50**	**309,50**
Amortización Inmovilizado Intangible		**0,00**	**0,00**	**0,00**
Fianzas y Depósitos			0,00	0,00

8.5 Resumen de inversiones

Tabla 8.11: Resumen de inversiones

INVERSIONES	AÑO 1

Total Inmovilizado Intangible	**89,81€**
Propiedad Intelectual	**89,81€**
Aplicaciones informáticas	**0,00€**
Total Inmovilizado material	**1.567,99€**
Mobiliario	**550,00€**
Equipos informáticos	**1.017,99€**
TOTAL Inversión (Sin I.V.A)	**1.657,80€**

8.6 Ingresos por ventas

Tabla 8.12: Ventas e ingresos

VENTAS E INGRESOS	AÑO 1	AÑO 2	AÑO 3
Ventas (Productos - Servicios)			
Ventas (Productos - Servicios)	4.500,00	8.500,00	13.000,00
Publicidad	777,00	1.554,00	4.788,00
Total ventas	5.277,00	10.054,00	17.788,00

8.6.1 Precios de la plataforma

Como ya se ha explicado anteriormente, la plataforma ofrece un uso completamente gratuito, no obstante, se ha establecido un sistema de beneficios de cara a los estudios que tienen un número determinado de ganancias, que publiquen en la plataforma.

La plataforma cobrara un porcentaje de cada venta en la plataforma, en base a los beneficios obtenidos por cada estudio (beneficios obtenidos íntegramente de ventas en la plataforma).

Tabla 8.13: Precios

Beneficios	Porcentaje
<1000€	GRATUITO
1000€ - 2000€	5% de cada venta.
2000€ - 5000€	10% de cada venta.
>5000€	15% de cada venta.

8.6.2 Publicidad

El anunciante pagará un precio fijo durante los dos primeros años el coste será de 12,95€ mensuales y en los años sucesivos se incrementará.

Se estima llegar en un tercer año a 19,95€ mensuales.

El primer año nos pondremos en contacto con diferentes Marcas para ofrecer nuestros servicios de publicidad.

Las estimaciones son 10 empresas el segundo año, y duplicarlo a 20 empresas el tercer año.

Tabla 8.14: Publicidad

	1er Año	2º Año	3º Año
Precio	**12.95€**	**12.95€**	**19.95€**
Nº Empresas	**5**	**10**	**20**
TOTAL	**777.00€**	**1554.00€**	**4788.00€**

8.7 Servicios Exteriores

Tabla 8.15: Servicios Exteriores

Servicios exteriores			
Arrendamientos			
Reparación y Conservación			
Servicios Profesionales	921,01	940,81	985,61
Transportes			
Primas de Seguros			
Servicios bancarios y similares			
Publicidad y Promoción	1.500,00	1.500,00	1.500,00
Suministros			
Gastos de Viaje	500,00	500,00	500,00
Tributos			
Gastos establecimiento			
Licencias y altas suministros	89,91	12,95	12,95
Otros Gastos			
Total servicios exteriores	**3.010,92**	**2.953,76**	**2.998,56**

Gastos excepcionales			

8.8 Gastos de Personal

Tabla 8.16. Gastos de personal

PERSONAL	AÑO 1	AÑO 2	AÑO 3
Sueldos y Salarios	12.600,00	16.380,00	16.380,00
Seguridad Social	1.119,54	2.724,24	3.204,18
Otros gastos de personal			
Gastos de personal	**13.719,54**	**19.104,24**	**19.584,18**

8.8.1 Gastos Financieros

Tabla 8.17: Gastos financiero

FINANCIACIÓN	AÑO 1	AÑO 2	AÑO 3
FINANCIACIÓN PROPIA			
Aportaciones dinerarias	3.005,00		
Subvenciones no reintegrables			
Subvenciones a la inversión			
FINANCIACIÓN AJENA			
Préstamos Bancarios Largo Plazo			
Cuantía del préstamo			
Tipo de interés %			
Plazo en años			
Préstamos Bancarios Corto Plazo			
Cuantía del préstamo			
Tipo de interés %			
Plazo en meses			
Gastos financieros	0,00	0,00	0,00

1.657,80€ de inversión + 1000,00€ Constitución de sociedad = 2.657€.

Aportación de capital ……. 3.005€

8.8.2 Cuenta de explotación previsional

Tabla 8.18: Cuenta de explotación previsional

Importe cifra de negocio				
	Importe cifra de ventas	5.277,00	10.054,00	17.788,00
Variación de existencias				
	Variación de existencias	0,00	0,00	0,00
Aprovisionamientos				
	Consumo de mercaderías	0,00	0,00	0,00
Otros ingresos de explotación				
	Otros ingresos	0,00	0,00	0,00
	Subvenciones de explotación	0,00	0,00	0,00
Gastos de personal				
	Gastos de personal	-13.719,54	-15.324,24	-15.804,18
Otros gastos de explotación				
	Arrendamientos	0,00	0,00	0,00
	Reparación y conservación	0,00	0,00	0,00
	Servicios profesionales	-921,01	-940,81	-985,61
	Transportes	0,00	0,00	0,00
	Primas de seguros	0,00	0,00	0,00
	Servicios bancarios y similares	0,00	0,00	0,00
	Publicidad y promoción	-1.500,00	-1.500,00	-1.500,00
	Suministros	0,00	0,00	0,00
	Gastos de viaje	-500,00	-500,00	-500,00
	Tributos	0,00	0,00	0,00
	Gastos establecimiento	0,00	0,00	0,00
	Licencias y altas suministros	-89,81	-12,95	-12,95
	Otros gastos	0,00	0,00	0,00
Amortización del inmovilizado				
	Amortización del inmovilizado	-309,50	-309,50	-309,50
Imputación de subvenciones de inmovilizado no financiero				
	Subvenciones a la inversión	0,00	0,00	0,00
Otros resultados				
	Ingresos Excepcionales	0,00	0,00	0,00
	Gastos excepcionales	0,00	0,00	0,00
RESULTADO DE EXPLOTACIÓN		**-11.762,86**	**-8.533,50**	**-1.324,24**
	Ingresos financieros	0,00	0,00	0,00
	Gastos financieros	0,00	0,00	0,00
RESULTADO FINANCIERO		**0,00**	**0,00**	**0,00**
RESULTADO ANTES DE IMPUESTOS		**-11.762,86**	**-8.533,50**	**-1.324,24**
	Impuesto sobre beneficios	0,00	0,00	0,00
BENEFICIO O PERDIDA DESPUÉS DE IMPUESTOS		**-11.762,86**	**-8.533,50**	**-1.324,24**

Podemos observar que el primer y segundo año, tenemos pérdidas considerables, no obstante, a partir del tercer año, se empiezan a recibir mayores beneficios, y pese a seguir presentando perdidas, son mínimas. Se puede estimar que, a partir del tercer año, se empezarían a tener beneficios, creciendo estos de manera considerable.

9

Valoración Personal

Mi valoración personal respecto al proyecto se divide en varias partes:

En primer lugar, considero la elaboración de un proyecto como este, una actividad interesante, ya que creo que muchos alumnos (entre los que me incluyo) nunca se han visto al mando de un proyecto de una envergadura relativamente grande, aunque sea solo a nivel teórico, lo que conlleva ciertas tareas de diseño y organizativas que no siempre resultan fáciles, y son de gran importancia.

Del mismo modo, aunque los profesores se presten a ayudar al alumno, resulta muy instructivo el hecho de enfrentarse a un proyecto en solitario, forzándote a la investigación, y a la aplicación de técnicas aprendidas en clase.

En segundo lugar, y estrictamente hablando del proyecto, estoy ampliamente satisfecho con el resultado. Aunque no ha sido posible toda la implementación del mismo, creo que es un campo lo suficientemente interesante como para que resulte posible su despliegue en un entorno real, en un futuro.

Una de las cosas más positivas de la informática es su bajo coste, ya que el comienzo del desarrollo de una aplicación, es relativamente barato, si no gratuito, lo que hace más interesante aún esta posibilidad.

Finalmente, me gustaría destacar ciertos aspectos tratados en clase y a lo largo del curso, que me han resultado muy útiles para la realización del proyecto, tales como, el proyecto realizado en la asignatura de DWES (tienda online), que conllevaba la implementación de una página web similar, aunque de menor amplitud, incluyendo una gran cantidad de métodos que han servido como plantilla para la elaboración del proyecto.

Del mismo modo, es destacable todo lo aprendido sobre el uso de jQuery, elemento que ha sido clave para la realización de este proyecto en todo momento, tanto para la realización de funcionalidades dinámicas en la página, como para el uso de métodos AJAX.

También ha sido de una importancia mayúscula el proyecto realizado en EIE, ya que se nos ha permitido hacer toda la planificación económica de un proyecto deseado, en muchos casos, el proyecto a desplegar, lo que ha facilitado enormemente la labor de la redacción y elaboración del mismo, sobretodo en muchas entregas relacionadas con los aspectos más estrictamente económicos.

Considero que ha sido un proyecto interesante, y estoy ampliamente satisfecho con el resultado obtenido, y la ayuda prestada por parte de los profesores.

Bibliografía

Contenido	Página	Autor
Explicación y protección contra XSS	http://www.elladodelmal.com/2010/10/un-xss-en-tuenti-que-permitia-hacer_28.html	Chema Alonso
Protección contra XSS	https://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/	Ignacio Pérez
Explicación y protección contra MySQL Injection	http://www.elladodelmal.com/2007/07/sql-injection-level-1.html	Chema Alonso
Protección contra MySQL Injection	https://diego.com.es/ataques-sql-injection-en-php	Diego Lázaro
Elaboración de diseños de clases	https://ingenieriadesoftware.es/las-mejores-guias-rapidas-para-uml/	Jordi Cabot
Elaboración de Diagramas E/R	https://www.lucidchart.com/pages/es/como-hacer-un-diagrama-entidad-relacion	LucidChart.com
Elaboración de Diagramas E/R	https://www.unirioja.es/cu/arjaime/Temas/02.Modelo_E_R.pdf	UniRoja
Data Access Objects (DAO)	https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/	Oscar Blancarte
Elaboración Modelo Vista Controlador (MVC)	https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf	Juan Pavón Mestras

Anexo I: Manual de Usuario

Para acceder a la aplicación, lo primero que necesitamos hacer es registrarnos. Para ello tendremos que hacer click en el botón “Regístrate” del login.

Figura A: Pantalla de Login

Esto nos abrirá el formulario de registro, donde introduciremos nuestros datos, y se nos dará de alta en la página.

Figura B: Formulario de registro

Una vez registrados, podremos volver a la pantalla de Login, y ejecutar el acceso. Esto nos llevará a la pantalla principal, donde veremos la cartelera de juegos, ordenados por enero. Encima de cada juego viene un circulo amarillo indicando la nota media de cada juego. En caso de no haber notas aun, se mostrará el carácter "-".

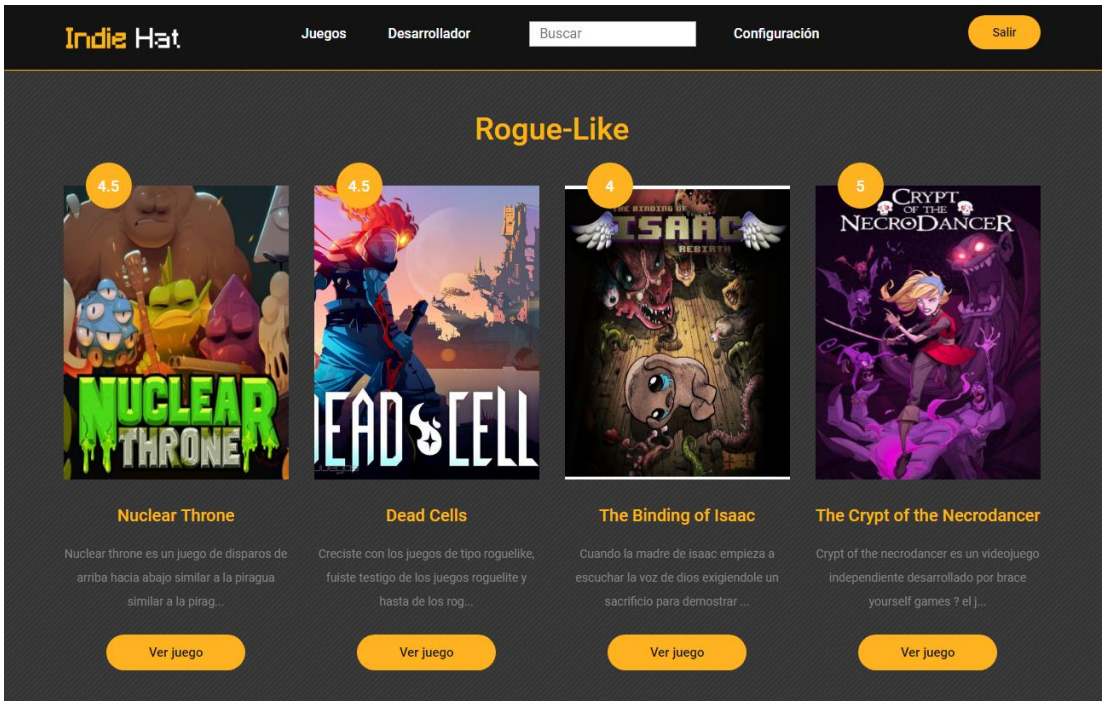


Figura C: Pantalla Principal

Si pulsamos en "Ver Juego", accederemos a la ficha de dicho juego, donde veremos sus fotos, su descripción, y su panel de valoraciones.

Figura D: Detalle de juego. Foto principal

Si bajamos desde la sección de Ver juego, accederemos a las valoraciones.

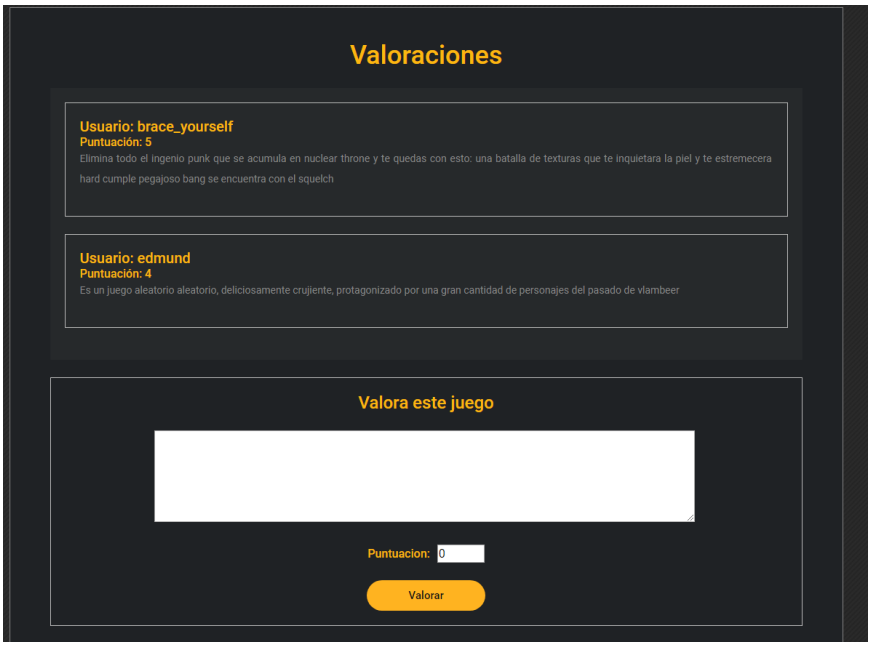


Figura E: Valoraciones

Aquí podemos ver lo que opinan otros usuarios del juego. Del mismo modo podemos dejar nuestra opinión, escribiendo en el cuadro de texto, poniendo la puntuación deseada en el cuadro Puntuación, y pulsando Valorar.

Figura F: Eliminar Valoración

Encima de nuestra valoración, aparecerá el botón eliminar, para borrarla en el caso que no deseemos que se muestre más. Volviendo a la sección principal, podemos ver que, si hacemos click en uno de los géneros, accederemos a la cartelera de juegos de ese género en concreto, mostrando todos sus títulos. Para volver, pulsaremos el botón "Volver".

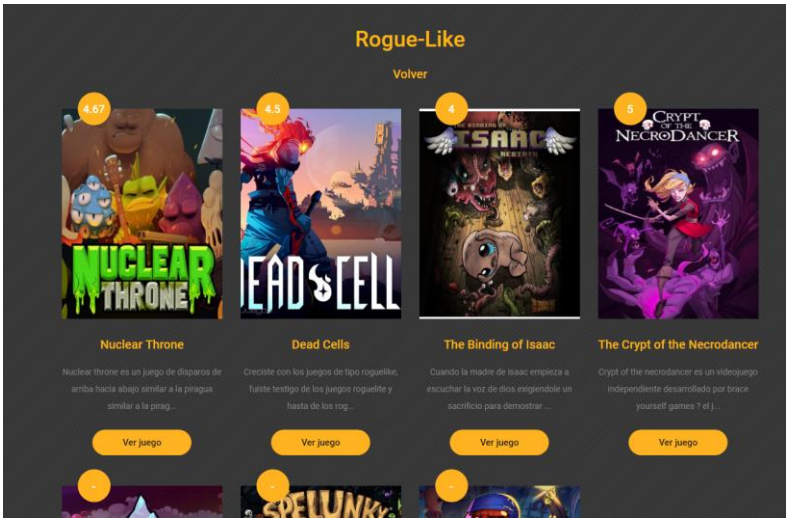


Figura G: Vista de Genero

Si introducimos una palabra en el buscador de la barra superior, la página nos mostrara todos los juegos relacionados con dicha palabra, ya sea por su título, su autor o su género.

Figura H: Buscador de juegos

Si accedemos a la sección de Configuración, podremos ver el formulario para cambiar la contraseña, donde debemos introducir la contraseña actual, y la nueva que deseamos.

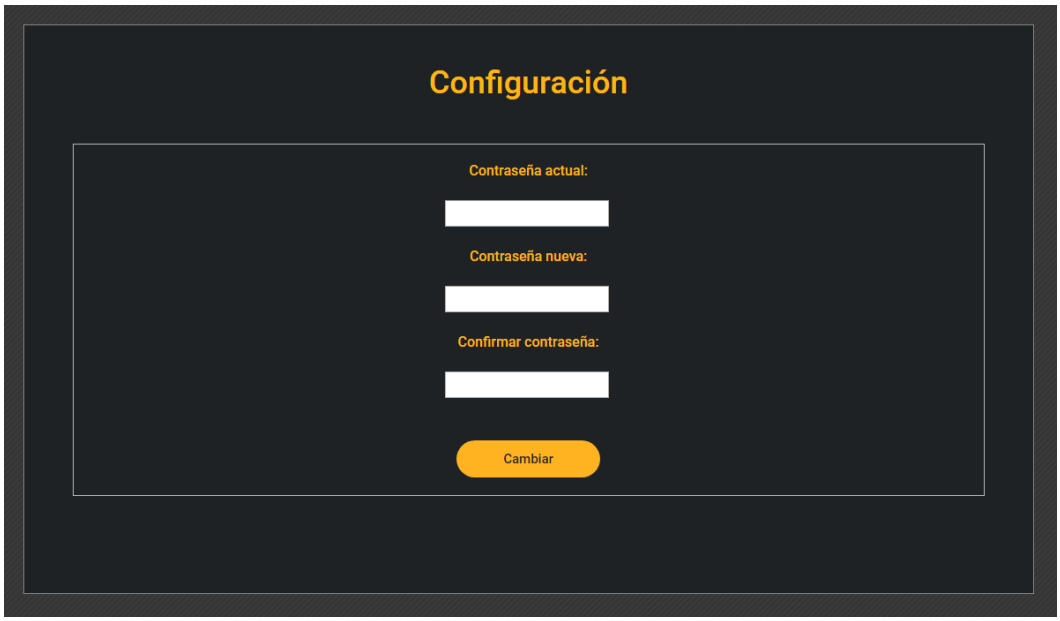


Figura I: Configuración de cuenta

Podemos observar que, en la sección desarrollador, se nos abre una sección de registro la primera vez que accedemos.

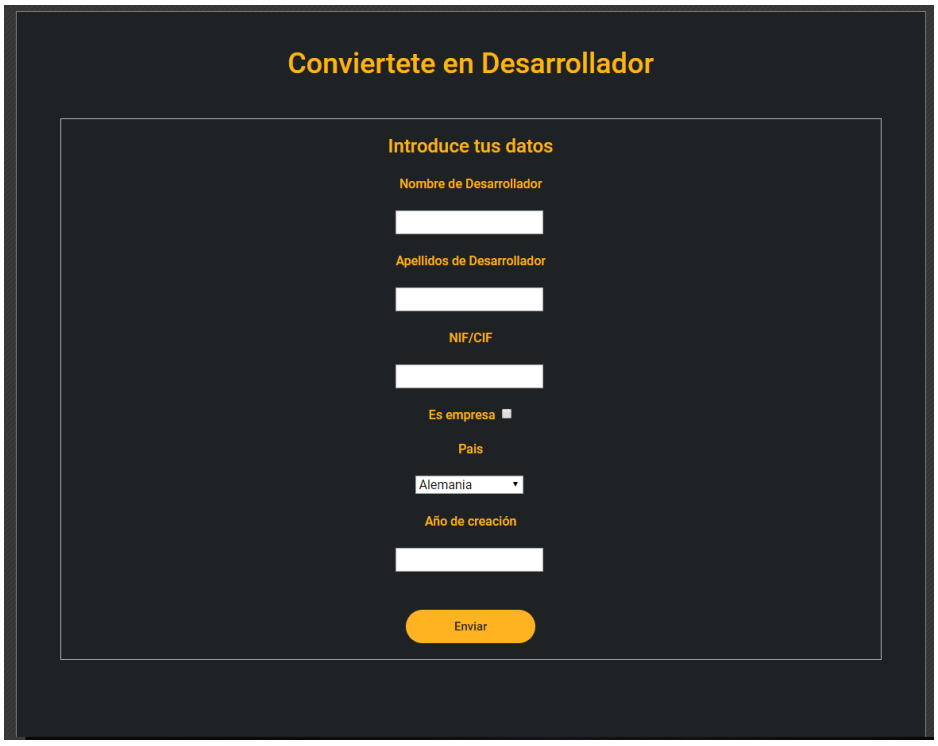


Figura J: Formulario de desarrollador

Una vez nos registramos como desarrolladores, veremos un panel que mostrara nuestros mensajes, y un buscador de estudios, para ver sus datos como Desarrolladores

Figura K: Panel de desarrollador

Si pulsamos mensajes, veremos la sección donde se muestran los mensajes recibidos, tanto leídos como sin leer (con una opción para marcarlos como leídos), y un formulario para enviar un mensaje al usuario deseado.

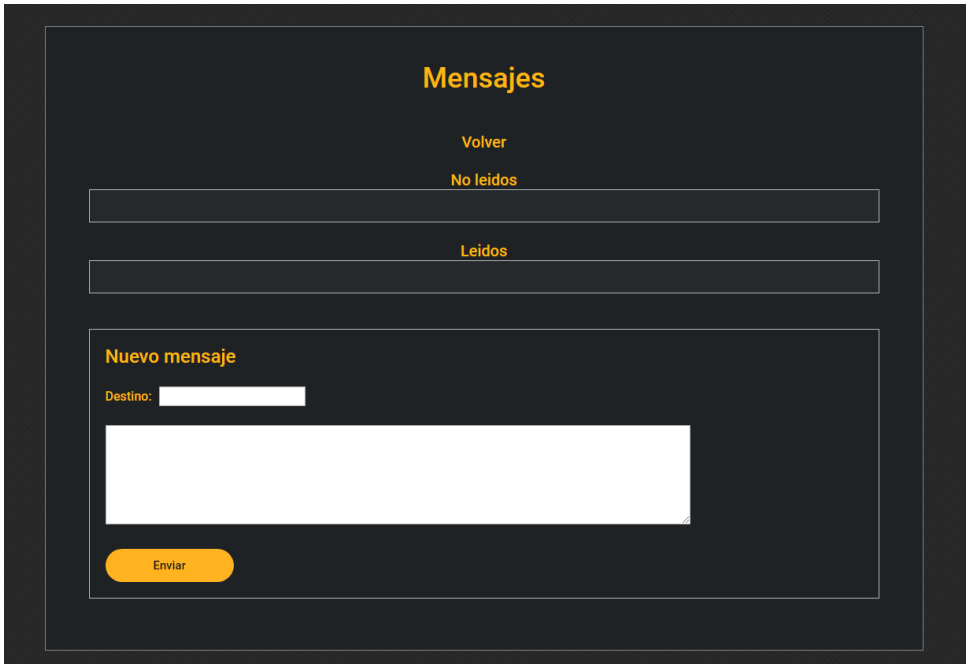


Figura L: Sección de mensajes

Por último, podemos ver que el buscador de estudios funciona como el buscador de juegos anteriormente explicado, buscando este tanto por nombre de estudio como por nombre de país al que pertenece dicho estudio.

Figura M: Buscador de estudios

Si accedemos a Ver estudio, veremos una ficha en detalle con todos los datos del estudio.

Figura N: Ficha de estudio

Para salir de la página, pulsaremos el botón “Salir” situado arriba a la derecha de la página, y este nos devolverá a la pantalla de Login.