Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

# Proyecto Fin de Grado

Ingeniería del Software

Desarrollo para Comunidad Web de Videojuegos

Portal Web de Videojuegos Online Multijugador

Héctor Antruejo Escalante

Junio 2016

# Agradecimientos

Creo que la parte de los agradecimientos es una de las partes más complicadas. A lo largo de la vida hay tanta gente que de una forma o de otra llegan a marcarte, que es difícil nombrarlas a todas. Cada uno de esos pequeños actos que pueden realizar dichas personas han marcado mi forma de ser, de tal forma, que he llegado hasta el punto de finalizar el grado de ingeniería informática, lo cual hace unos años me habría sido impensable. ¡Gracias a todos de corazón!

Quiero agradecer en primero lugar a Iker Boyra por su paciencia conmigo a la hora de dar explicaciones y su consejo continuo que me ha ayudado a lo largo de toda la carrera. También agradezco su forma de ser y nuestra mutua cabezonería para hacer las cosas, la cual en gran medida, es la causante de que iniciemos la iniciativa IHEGames.

A todos aquellos profesores como Montserrat Hermo, José Ángel Vadillo y en especial a José Miguel Blanco los cuales van más allá de su labor como profesores y se involucran con los alumnos, motivándonos y ayudándonos de una forma que claramente excede de su labor docente.

A amigos y familiares por mostrarme siempre vuestro apoyo y motivarme en todo momento.

A todos vosotros que de una forma o de otra me habéis ayudado y motivado a acabar la carrera. ¡Gracias a todos!

# Resumen

A raíz de la iniciativa empresarial IHEGames, surgen dos proyectos de fin de grado paralelos con el objetivo de desarrollar unas primeras versiones de los sistemas que forman el núcleo de esta iniciativa empresarial.

En este proyecto de fin de grado se ha desarrollado tres servicios web cada uno con una finalidad distinta. El servicio web de cuentas de usuario es el encargado de almacenar tanto las credenciales como algunos datos referentes a los usuarios, el servicio web de honor se encarga de evaluar la integridad de los usuarios y finalmente el servicio de Elo evalúa la habilidad y experiencia de los usuarios en los juegos de IHEGames.

Finalmente se desarrolla una página web, la cual hace uso de los servicios web desarrollados e integra el juego de K-Tan desarrollado en el proyecto paralelo a éste.

# Índice

# Figuras

# Tablas

# 1

# Introducción

En este capítulo se describe el contenido de esta memoria, por un lado se mencionan los capítulos que la componen junto con una breve descripción del contenido de éstos, y por otro lado, se explica la estructura y organización del documento.

## 1.1 Introducción del proyecto

Este proyecto consiste en el desarrollo de una parte de las funcionalidades que compondrán la base de la iniciativa IHEGames. Una iniciativa que ha sido posible gracias a los conocimientos adquiridos a lo largo de la carrera, la predisposición en ir más allá en algunas tareas, las ganas y la ambición que nos caracteriza.

El desarrollo del proyecto presentado en esta memoria ha sido realizado durante el trascurso del curso 2015-2016 del Grado de Ingeniería del Software y culmina la primera etapa o el inicio de IHEGames junto con el proyecto de *Desarrollo para Comunidad Web de Videojuegos: Videojuego 3D Online Multijugador Multiplataforma*[1] desarrollado en el proyecto de fin de grado por Iker Boyra Sarachaga. Las partes comunes que afectan en ambos proyectos y han sido incluidas en ambos proyectos se denotan con un color de letra más grisáceo.

En los primeros capítulos se realiza una introducción de las bases de IHEGames junto con los objetivos del proyecto. En el capítulo de *Antecedentes* se explica en qué consiste IHEGames, su arquitectura y una explicación más detallada de sus objetivos. Posteriormente En el capítulo de *Objetivos* se habla de las metas propias que definen este proyecto, es decir, en qué consiste la parte de la base de IHEGames que se va a diseñar e implementar. El capítulo *Ciclo de vida* del proyecto, describe el proceso que sigue IHEGames para el desarrollo de sus proyectos, el cual ha determinado la forma en la que se ha desarrollado este proyecto.

Tras finalizar con los capítulos introductorios anteriormente mencionados, se pasa a hablar de las tecnologías utilizadas, las cuales tienen un gran impacto en el desarrollo del proyecto. El capítulo de *Windows Communication Foundation* introduce al lector en los servicios web de ASP.NET de Windows Communication Foundation, cuáles son sus puntos fuertes y que posibilidades ofrece. Por otro lado en el capítulo de ASP.NET MVC. Se explica en qué consiste ASP.NET MVC y se habla de algunas características de MVC 6 respecto a versiones anteriores.

Una vez se ha hablado de las tecnologías utilizadas se pasa a los capítulos relacionados con el diseño e la implementación del proyecto. Se diferencia por un lado los servicios web y por otro lado el portal web. Se comienza hablando en el capítulo de *Servicios web* sobre las partes comunes que tienen todos los servicios web. Su arquitectura, similitudes en el modelo de datos, estructura y las librerías comunes. Tras ello se habla más en detalle de los propios servicios web en los capítulos *Servicio Web de cuentas de usuario*, *Servicio Web de Honor* y *Servicio Web de Elo*. De el mismo modo, en el capítulo de *Portal Web D20* se habla de cómo se ha adaptado el modelo de ASP.NET MVC 6 para adecuarse a los objetivos del proyecto, haciendo posible la intercomunicación entre los servicios web previamente implementados con el portal web D20, y como se integra el primer juego desarrollado en IHEGames en el portal web.

Finalmente se habla de cómo se ha gestionado el proyecto en el capítulo *Gestión*, cuáles han sido los hitos más importantes en el desarrollo, que planificación ha habido tanto a nivel individual como a nivel de organización, y finalmente que desviaciones han surgido en el desarrollo del proyecto. Y una buena gestión no podría acabar sin unas conclusiones, que embarcan las lecciones aprendidas, tanto a nivel personal como organizacional del desarrollo de este proyecto, éstas quedan reflejadas en el último capítulo *Conclusión*.

## 1.2 Formato del documento

Este proyecto se realiza en paralelo a otro proyecto de fin de grado con el que tiene grandes dependencias. Por ello es inevitable que algunos capítulos de esta memoria contengan partes comunes que se comparten en las memorias de ambos proyectos.

Por otro lado, ambos proyectos de fin de grado se presentan en el mismo periodo. Para facilitar al lector la comprensión del documento y el saber si lo que lee es algo común de las dos memorias o algo específico de ésta, se han presentado las partes comunes con un color de letra más claro y grisáceo, mientras que las partes específicas de este proyecto se mantienen en color negro.

# 2

# Antecedentes

Las motivaciones personales y empresariales que han impulsado la realización de este proyecto son recogidas en este capítulo, así como las pautas que dicha iniciativa empresarial ha marcado en el desarrollo del mismo.

## 2.1 IHEGames

A lo largo del año 2014, Héctor Antruejo e Iker Boyra deciden unirse con la idea de aplicar sus conocimientos profesionales y académicos adquiridos a lo largo del grado de informática para crear la iniciativa empresarial IHEGames.

Tras haber trabajado conjuntamente en varios proyectos desde que sus caminos se cruzaron en Ingeniería del Software, y debido a su afán compartido por los videojuegos, deciden enfocar la iniciativa empresarial IHEGames al desarrollo de videojuegos online.

A comienzos del 2015 cuando la iniciativa aun está comenzando a tomar forma, las ventas de los videojuegos son casi equiparables a las ventas en grandes industrias como podría ser la del cine, tal y como recoge la Asociación Española de Videojuegos[2], la cual asegura que: *"La industria del videojuego ha mantenido en los últimos años su posición como principal industria de ocio audiovisual e interactivo, con una cuota de mercado muy superior a la del cine y la música".*

La iniciativa IHEGames pretende crear, entre otros, un portal de juegos denominado D20 y desarrollar los juegos que alojara en él.

El portal D20 ofrecerá a los jugadores de todo la posibilidad de satisfacer sus necesidades de juego. El portal D20 se basa en una página web que dispondrá de una variedad de juegos de calidad que cuiden sus gráficos, aspecto visual y jugabilidad. A su vez, dichos juegos podrán ser descargables y jugables desde otras plataformas ajenas al portal, como por ejemplo, dispositivos móviles.

El portal D20 no es un mero portal web. D20 aspira a convertirse en una gran comunidad de jugadores de todo el mundo, una comunidad en la que los usuarios podrán interactuar y jugar con el resto de usuarios a la selecta variedad de juegos que tendrán a su disposición.

El ámbito de los videojuegos es muy extenso como para abarcarlo completamente, por lo que se decide priorizar el lanzamiento de videojuegos por categorías.

El sector de los juegos de mesa ha crecido significativamente en la última década. Este sector, que hasta hace poco se centraba en juegos para niños o para la familia, ha ampliado sus objetivos a jugadores más experimentados dando como resultado una enorme variedad de juegos de mesa de todo tipo y dificultad. Aunque este sector sigue siendo desconocido para gran parte de la sociedad las cifras de ventas de este tipo de juegos han aumentado considerablemente tal y como se muestra en esta noticia de El Mundo[3]: *"Dos juegos que ya se pueden considerar dos fenómenos mundiales con 20 millones de copias vendidas (Catán) y 10 millones de copias vendidas (Carcassonne)."*

Dada la carencia detectada, IHEGames decide aprovecharla y centrarse en sus inicios en el desarrollo de juegos de mesa (juegos de mesa, tablero, cartas, casino) de tipo multijugador.

## 2.2 Arquitectura de IHEGames

IHEGames establece siete sistemas principales, que compondrán la arquitectura base del portal D20.

Estos sistemas consisten en el propio portal D20, los juegos que se alojan en D20 y los servicios comunes que dan soporte tanto al portal como a los propios juegos.

En la siguiente figura se muestra la arquitectura general que componen los sistemas base de IHEGames y las dependencias que existen entre ellos:

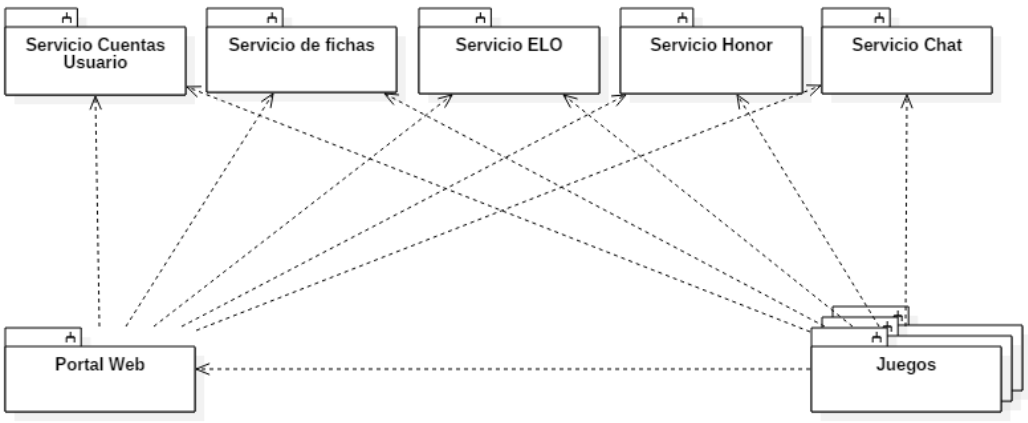


Figura 1: Sistemas de IHEGames

Todas las comunicaciones entre estos sistemas, se realizan mediante conexiones suficientemente seguras, de manera que se pueda garantizar la autenticidad y la privacidad de los datos transmitidos.

### 2.2.1 Servicio web de cuentas de usuario

Esta aplicación almacena las Credenciales de los usuarios de IHEGames y es el sistema de gestión de usuarios que utilizan el resto de sistemas de IHEGames. Este servicio es de acceso exclusivo de las aplicaciones de IHEGames.

Por otro lado almacena los datos básicos de perfil de los usuarios y sus avatares.

El servicio de cuentas limita la creación y modificación de cuentas de usuario a algunas aplicaciones exclusivas, permitiendo al resto únicamente obtener los datos de las cuentas y autentificar a los usuarios.

El servicio de cuentas permite que la autentificación de los usuarios se realice mediante usuario y contraseña, o mediante otros proveedores de cuentas como Google, Facebook y Twitter.

### 2.2.2 Servicio de fichas

Esta aplicación almacena y gestiona las fichas de los usuarios de IHEGames. Este servicio es de acceso exclusivo de las aplicaciones de IHEGames.

Las fichas consisten en una moneda virtual que los usuarios pueden apostar en las partidas y utilizarlas para tener acceso a funcionalidades premium. Estas fichas se pueden obtener mediante pagos con dinero real o como bonificaciones otorgadas a los usuarios en su uso de las aplicaciones de IHEGames.

### 2.2.3 Servicio web de Elo

Esta aplicación es la encargada de gestionar tanto la puntuación Elo como el nivel y la Experiencia de los usuarios en los diversos juegos de IHEGames.

El sistema de puntuación Elo[4] es un método matemático, basado en cálculo estadístico, para calcular la habilidad relativa de los jugadores en juegos como el ajedrez. La puntuación Elo de un jugador aumenta con las victorias y disminuye con las derrotas. La cantidad que aumenta o disminuye depende de su puntuación Elo y la de sus rivales.

La Experiencia de un usuario simboliza su experiencia de juego y la cantidad de partidas de un juego concreto en las que ha participado. El nivel de usuario se calcula dependiendo de su experiencia mediante otra fórmula matemática.

El servicio calcula y almacena una puntuación Elo, la Experiencia y el nivel para cada usuario en cada juego. Por otro lado, el servicio da la posibilidad de permitir que aplicaciones ajenas a IHEGames, con sus propios usuarios, sean capaces de consumir este servicio, siempre y cuando estén autorizadas para poder hacer uso de este servicio.

### 2.2.4 Servicio web Honor

Esta aplicación es la encargada de gestionar el Honor de los usuarios en los diversos juegos de IHEGames.

El Honor de un usuario es un concepto utilizado por IHEGames que simboliza la conducta o comportamiento general que ha demostrado en partidas anteriores y permite al resto de usuarios prever el tipo de jugador que es.

El Honor de un usuario se calcula mediante una fórmula matemática en base a los votos positivos o negativos recibidos, los abandonos y los retrasos de las últimas partidas en un juego concreto. Los votos positivos suben el Honor de un usuario. Por el contrario, los votos negativos, los abandonos y los retrasos disminuyen su Honor.

El servicio calcula y almacena unos datos de Honor para cada usuario en cada juego. Y como el servicio de Elo, este servicio también da la posibilidad de permitir que aplicaciones ajenas a IHEGames, con sus propios usuarios, sean capaces de consumir este servicio, siempre y cuando estén autorizadas para poder hacer uso de este servicio.

### 2.2.5 Servicio de Chat

Esta aplicación permite que los usuarios de IHEGames se puedan comunicar entre ellos mediante el envío de mensajes. Este servicio está limitado a los usuarios de IHEGames, pudiendo solo estos acceder al servicio.

Este servicio permite enviar mensajes a los usuarios de una misma aplicación o a usuarios en otras aplicaciones. Por otro lado el servicio permite enviar mensajes privados a usuarios concretos o a los usuarios de la misma partida.

### 2.2.6 Portal de juegos D20

Aplicación web que expone los juegos desarrollados por IHEGames y ofrece a sus usuarios un entorno para interactuar y jugar entre ellos.

En este portal D20 los usuarios pueden registrarse o modificar sus datos de cuenta, perfil y avatar. Los usuarios tienen también la posibilidad de añadir a otros usuarios a su lista de amigos. Los usuarios identificados pueden visualizar y acceder a los juegos alojados en el portal D20, así como visualizar su puntuación Elo, Experiencia nivel y Honor en los diversos juegos.

El portal D20 ofrece la posibilidad de visualizar su contenido en múltiples idiomas. También contiene un panel de administración, desde el cual se gestionan las noticias que se pueden publicar en el portal.

### 2.2.7 Juegos

Los juegos de IHEGames permiten a sus usuarios jugar partidas contra otros usuarios a través de internet en tiempo real. Los juegos son multiplataforma, estos están alojados en el portal D20 para jugarlos en un navegador y están también disponibles en Google Play y App Store para descargarlos y jugar en dispositivos Android e IOS respectivamente.

Los usuarios pueden crear nuevas partidas e invitar a otros usuarios a entrar a ellas. Los usuarios pueden visualizar todas las partidas disponibles y unirse a las partidas creadas por otros usuarios. Los usuarios pueden también visualizar partidas en curso uniéndose a ellas como espectador.

Al crear una partida los usuarios pueden configurar unos parámetros

- El número de jugadores: determina el número de jugadores que tiene la partida.
- Espectadores: determina si se permite que otros usuarios se unan como espectadores a la partida.
- Tipo de acceso: limita el acceso de otros jugadores a la partida a cualquiera, a amigos o a invitados.
- Partida competitiva: determina si el resultado de la partida influirá en las puntuaciones Elo de los jugadores.
- Limitar Elo: determina si solo los jugadores de una misma categoría de Elo o superior puedan acceder a la partida. La categoría de Elo de un usuario depende de su puntuación Elo. Las categorías existentes son “sin categoría”, bronce, plata, oro, diamante y platino.
- Limitar Honor: determina si se restringe el acceso a los jugadores no honorables. Los jugadores no honorables son aquellos usuarios que tienen un valor de Honor inferior a un valor determinado.
- Fichas: Determina el número de fichas que apuestan los jugadores en la partida.

Al igual que el portal D20, los juegos ofrecen la posibilidad de visualizar su contenido en múltiples idiomas

## 2.3 Objetivos de IHEGames

En septiembre de 2015 los promotores de IHEGames plantean una etapa previa a la construcción de IHEGames como entidad empresarial. En esta etapa, IHEGames pretende para noviembre de 2016 tener en funcionamiento la arquitectura base descrita en el apartado anterior con un primer juego en funcionamiento.

Dado el periodo académico de los promotores de IHEGames, se establece una sub-etapa hasta junio de 2016. A lo largo de este periodo se plantea desarrollar, en dos proyectos de fin de grado, unos prototipos funcionales de los servicios de cuentas de usuarios, Elo y Honor, así como el desarrollo del Portal D20 y del primer juego de IHEGames.

Como primer juego, IHEGames toma la determinación de desarrollar el juego K-Tan, basado la versión básica del juego de mesa “Colonos de Catan”. Con este primer juego se establece la base para el desarrollo de los futuros juegos de IHEGames.

Otro de los objetivos fundamentales en esta etapa consiste en desarrollar y perfeccionar una cultura organizativa que consolide la base de un adecuado trabajo en equipo y faciliten la gestión, tanto de este proyecto como de proyectos futuros.

Adicionalmente se pretende lograr una eficaz comunicación y visibilidad entre los distintos proyectos de IHEGames y sus equipos de trabajo.

## 2.4 Tecnologías de IHEGames

IHEGames establece y marca una serie de tecnologías concretas para el desarrollo y la elaboración de sus proyectos.

### 2.4.1 Unity

IHEGames establece Unity como su principal plataforma de desarrollo de juegos, que serán desarrollados con tecnología 3D.

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. El scripting de Unity viene a través de Mono. El script se basa en Mono, la implementación de código abierto de .NET Framework. Los programadores pueden utilizar UnityScript (un lenguaje personalizado inspirado en la sintaxis ECMAScript), C# o Boo (que tiene una sintaxis inspirada en Python).

Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X y Linux, y permite crear juegos para IOS, Android, Windows Phone, Tizen, Windows, Windows Store Apps, Mac, Linux/Steam OS,WebGL, Play Station, Xbox, Wii, Nintendo 3DS, Oculus Rift, Google CarBoard, Steam Vr, Gear Vr, Microsoft Hololens, Android Tv, Samsumg Smart TV, TvOS.

IHEGames establece que las versiones web de sus juegos se compilen en WebGL. WebGL está desarrollada por Mozilla Foundation y consiste en un estándar para motores gráficos de juegos en 3D para navegadores.

WebGL permite mostrar gráficos en 3D acelerados por hardware (GPU) en páginas web, sin la necesidad de plug-ins en cualquier plataforma que soporte OpenGL 2.0 u OpenGL ES 2.0. Técnicamente es un API para javascript que permite usar la implementación nativa de OpenGL ES 2.0 que será incorporada en los navegadores. WebGL es gestionado por el consorcio de tecnología sin ánimo de lucro Khronos Group .

Unity cuenta con Unity Asset Store que es un recurso disponible en el editor de Unity. Los usuarios de Unity pueden acceder a una colección de paquetes de Assets en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales y proyectos, paquetes de scripts, extensiones para el editor y servicios en línea.

### 2.4.2 ASP.NET

IHEGames establece ASP.NET como framework principal para el desarrollo de sus servicios y aplicaciones web. ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

### 2.4.3 Visual Studio 2015

IHEGames estable Visual Studio 2015 como su principal entorno de desarrollo para aplicaciones ASP.NET

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, etc.

Los usuarios de Visual Studio pueden utilizar su extensión integrada Nuget. Nuget es un gestor de paquetes gratuito y de software libre diseñado para la plataforma de desarrollo de Microsoft.

### 2.4.4 C#

IHEGames establece C# como lenguaje principal de desarrollo tanto en Unity como en las aplicaciones realizadas con Visual Studio.

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Por otro lado, IHEGames establece el uso del inglés para las definiciones de los nombres de variables, propiedades, clase, métodos, etc. Por ello, todos los diagramas mostrados en esta memoria que hacen referencia a elementos de código o al propio código mantendrán su nomenclatura anglosajona.

### 2.4.5 WCF

IHEGames establece WCF como principal plataforma de mensajería de sus servicios web. Windows Communication Foundation o WCF, es la plataforma de mensajería que forma parte de la API de la Plataforma .NET 3.0.

Fue creada con el fin de permitir una programación rápida de sistemas distribuidos y el desarrollo de aplicaciones basadas en arquitecturas orientadas a servicios (también conocido como SOA), con una API simple; y que puede ejecutarse en una máquina local, una LAN, o sobre Internet en una forma segura.

### 2.4.6 ASP.NET MVC

IHEGames establece el patrón ASP.NET MVC para la elaboración del portal D20 y del resto de sus páginas web.

El ASP.NET MVC Framework es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador (MVC). Basado en ASP.NET, permite a los desarrolladores de software construir una aplicación web como una composición de tres funciones: modelo, vista y controlador.

Uno de los módulos a destacar de ASP.NET MVC es ASP.NET Identity ASP.NET Identity es un sistema para la gestión de usuarios en aplicaciones web ASP.NET. Esto incluye la gestión de las Credenciales y almacenamiento de datos de los usuarios en las páginas web. Adicionalmente ASP.NET Identity permite la extensión de sus funciones de manera sencilla.

### 2.4.7 SSL/TLS

IHEGames estable como protocolo de comunicación el uso de SSL/TLS (Secure Socket Layer / Trasport Layer Security) en todas las comunicaciones de sus aplicaciones. Este protocolo proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

Desde comienzos del siglo XXI el uso de HTTPS se ha ido popularizando en las aplicaciones de internet. Esto se ve reflejado con la iniciativa de Https Everywhere[5] promocionada por la Electronic Frontier Foundation y fundada en el año 2010, que fomenta el uso de conexiones cifradas HTTPS. Por otro lado buscadores como Google comienzan a premiar el uso de este protocolo, aumentando el SEO de las URL HTTPS, como indican en su blog de seguridad[6]: *"we also started giving a slight ranking boost to HTTPS URLs in search results last year".*

### 2.4.8 Diseño Adaptativo o Responsive

IHEGames establece como política de diseño el uso de interfaces con un diseño adaptativo o Responsive.

Es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las aplicaciones al dispositivo que se esté utilizando para visualizarlas. Hoy día las aplicaciones se ejecutan y visualizan en multitud de dispositivos como tabletas, teléfonos inteligentes, libros electrónicos, portátiles, PCs, etcétera. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, sistema operativo o capacidad de memoria entre otras. Esta tecnología pretende que con un único diseño, se obtenga una visualización adecuada en cualquier dispositivo.

Esto se refleja en buscadores como el de Google que crea la iniciativa de Mobile Friendly[7] aumentando el valor SEO de páginas web con diseño Responsive y penalizando las páginas que no lo cumplan.

### 2.4.9 Mingle

*IHEGames establece Mingle como herramienta para la gestión y el seguimiento y control de sus proyectos.*

*Mingle es una herramienta online para la gestión de proyectos que permite a compañías de cualquier tamaño implementar, gestionar y escalar metodologías ágiles.*

*La herramienta Mingle ha sido desarrollada por Thoughtwork Inc. y es utilizada también por empresas conocidas de gran envergadura y mucho peso en el ámbito tecnológico como son: Siemens o Cisco.*

*Mingle tiene a disposición de sus usuarios, plantillas basadas en metodologías ágiles para el desarrollo de software como son el Kanban, Agile o Scrum.*

*Mingle, permite trabajar con cartas conocidas como "Index Card" las cuales pueden simbolizar una tarea, "story", bug, característica, y mucho más. Junto a las muchas características que las cartas pueden tener, las cartas contienen una descripción y un responsable.*

*Las cartas se sitúan en muros o "Card Walls", donde se pueden organizar de múltiples maneras, ya sea por el tipo de carta, por la etapa en la que se encuentra, por las características que compartan, etc.*

### 2.4.10 Mysql

IHEGames establece Mysql como sistema de gestión de bases de datos. Mysql es un sistema de gestión de bases de datos relacionales desarrollada bajo licencia dual GPL y Licencia comercial por Oracle Corporation. La base de datos relacional es un tipo de base de datos que cumple con el modelo relacional el cual permite establecer interconexiones o relaciones entre los datos almacenados.

A la fecha de la realización de este proyecto, Mysql es una de las bases de datos más utilizadas, estando en el segundo puesto de bases de datos más utilizadas según la página web de db-engines.com[8]

### 2.4.11 OAUTH 2

Es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones, permitiendo a aplicaciones el acceso limitado a los datos de usuario almacenados en otro servicio o aplicación. También permite comprobar la autorización de forma segura en la aplicación externas al sitio sin la necesidad de proporcionar sus Credenciales en. Es decir, en este caso, permite que las cuentas de Google, Facebook y Twitter puedan ser usadas en el portal web D20 sin tener la necesidad de compartir todos los datos que tiene los usuarios en dichos servicios ni las Credenciales de los usuarios en ellas.

En el caso del portal web D20 solo se comparte el nombre de usuario y el correo electrónico de los usuarios.

A continuación se muestra el flujo de datos del funcionamiento de Oauth2 entre el servicio Oauth2 de Google y el portal D20.

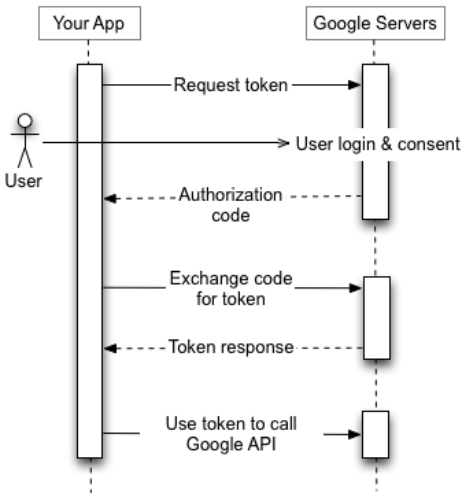


Figura 2: Usando Oauth 2.0 para acceder a las APIs de Google[9]

El funcionamiento es el mismo para acceder a las APIs de Facebook y Twitter.

# 3

# Objetivos

En este capítulo se explica brevemente en qué consiste la base de los sistemas de la iniciativa IHEGames y sus objetivos respecto al desarrollo de dichos sistemas base. También se habla de cuáles de dichos sistemas se van a desarrollar en este proyecto y cuál va a ser su alcance a través de los apartados de inclusiones e exclusiones.

## 3.1 Objetivos

El objetivo principal del proyecto consiste en el diseño y desarrollo de unos prototipos funcionales de una parte de los sistemas base de la iniciativa de IHEGames. Los sistemas a desarrollar consisten en los servicios web de cuentas de usuario, Elo y Honor así como de su portal web D20.

Tras la realización de los sistemas base y el portal D20 también se pretende que los servicios sean capaces de dar servicio al portal D20 así como al juego de K-Tan, pudiendo así, acceder a las funcionalidades ofrecidas por estos.

El portal D20 debe ser capaz de dar la posibilidad a nuevos usuarios de poder registrarse en él y a los ya existentes de poder modificar sus datos de cuenta de usuario. Dichos datos y registros tienen que quedar almacenados en el servicio de cuentas de usuario.

Los usuarios tienen que poder ver su puntuación de Elo y Honor así como su Experiencia de juego de K-Tan desde el apartado correspondiente en el portal web D20.

Finalmente se va a alojar el juego de K-tan en el portal web D20 y se probará tanto la funcionalidad del portal D20 como la posibilidad de jugar al juego K-Tan desde el propio portal web.

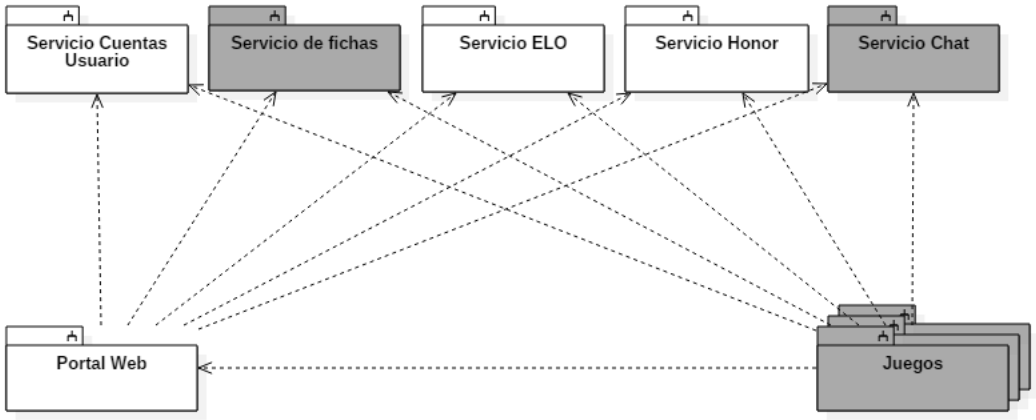


Figura 3: Sistema IHEGames

## 3.2 Alcance del proyecto

En este apartado se definen las inclusiones y exclusiones derivadas de las necesidades de IHEGames y las limitaciones que la realización del proyecto académico conlleva.

### 3.2.1 Inclusiones

#### 3.2.1.1 Inclusiones generales para todos los servicios Web

- Comprobar la autenticidad de las aplicaciones que hagan uso del servicio web, es decir, comprobar que la aplicación cliente es quien dice ser.
- La comunicación entre el servicio y las aplicaciones cliente, debe garantiza la confidencialidad de los datos de los usuarios.
- El servicio ofrece la oportunidad de hacer un seguimiento de las acciones realizadas por las aplicaciones cliente.
- Realización y documentación de unas pequeñas pruebas en los servicios web que comprueben su funcionamiento.

#### 3.2.1.2 Servicio de cuentas de usuario

- Comprobar la autorización al servicio por parte de las aplicaciones cliente, limitando el uso en dos ámbitos, funciones que permitan la edición de las cuentas de usuario y funciones que obtengan datos de las cuentas de usuario.
- El servicio almacena los datos de usuarios de todas las aplicaciones de IHEGames.
- Contiene los métodos necesarios para comprobar las Credenciales de los usuarios en las aplicaciones.
- En caso de repetidos fallos en la comprobación de las Credenciales el servicio bloquea el usuario por un periodo establecido de 2 minutos, es decir, todas las peticiones de comprobación devolverán un estado de cuenta de usuario bloqueada.
- Métodos para la creación de nuevos usuarios en el servicio,
- Edición de los datos de usuario.
- Generación de Token para algunas aplicaciones que así lo requieran.
- Los usuarios pertenecen a grupos denominados roles, el servicio contiene métodos para la creación edición de roles, así como para añadir y eliminar usuarios de estos.

#### 3.2.1.3 Servicio web de Elo

- Calcular el Elo de los usuarios desde de los reportes recibidos.
- Calcular el nivel de Experiencia de los usuarios desde los reportes recibidos.
- Los reportes de los juegos con el resultado final de las partidas se almacena en el servicio web de Elo.
- Devolver el Elo y Experiencia de los usuarios pedidos.
- Obtener el Elo de los 10 jugadores con mayor nivel de Elo en la aplicación cliente.

#### 3.2.1.4 Servicio de Honor

- Las aplicaciones cliente pueden tener relación con otras aplicaciones mediante relaciones padre-hijo. Un hijo solo puede tener un padre, en cambio un padre puede tener más de un hijo.
- Las aplicaciones cliente solo pueden ver el nivel de Honor de los usuarios de su aplicación, y las aplicaciones hijo.
- Calcula el Honor en base a los reportes recibidos.
- Almacenar los reportes de Honor recibidos.
- Devolver el Honor de los usuarios pedidos.
- Devolver los 10 jugadores más honorables de la aplicación cliente junto con su valor de Honor.
- Devolver los 10 jugadores más honorables, con el valor ponderado entre la aplicación cliente y todas sus aplicaciones hijo.

#### 3.2.1.5 Portal web

- El portal puede ser visualizado en varios idiomas, español e inglés.
- Los usuarios se pueden registrar en el portal
- Se distinguen dos tipos de cuenta en el portal, cuenta local y cuentas externas.
    - Si el usuario se ha registrado con una cuenta local, tendrá que validar su correo electrónico.
    - Las cuentas externas son Facebook, Google y Twitter, y el usuario no tendrá que validar su correo electrónico.
- Los usuarios podrán identificarse y conectarse en el portal mediante sus Credenciales.
- Una vez autentificado el usuario, puede modificar sus datos de cuenta.
- El usuario no puede modificar su nombre de usuario ni su dirección de correo electrónico.
- El portal almacena los juegos proporcionados por IHEGames.
    - Los usuarios no autentificados en el portal podrán ver los juegos del portal
    - Solo los usuarios autentificados podrán jugar en los juegos del portal.
- En cada juego correspondiente, el portal muestra los jugadores más honorables, con una mayor puntuación de Elo y mayor nivel de Experiencia.

### 3.2.2 Exclusiones

Dado que este proyecto es una fase inicial cuyo objetivo principal es lograr un prototipo meramente funcional, se excluye del alcance del proyecto todo lo anteriormente no mencionado, entre lo que se destaca:

- El desarrollo y creación de los juegos que el portal alojará.
- La justificación de las alternativas de las tecnologías elegida para la realización de este proyecto.
- El despliegue online de los servicios web y el portal D20.
- Adquisiciones de certificados validados por entidades de certificación reconocidas.
- Diseños web y pruebas del diseño que acrediten que el portal web D20 cumple los estándares de accesibilidad establecidos por Web Accessibility Iniciative[10] o similares.
- Debido a que al finalizar la etapa del proyecto, éste no va a ser desplegado, por lo que los servicios y la página web permanecen en un ámbito privado, se excluye la implantación de los requerimientos exigidos por la Agencia Española de protección de datos[11].
- Análisis del rendimiento y disponibilidad, tanto de los servicios como de la web.

## 3.3 EDT

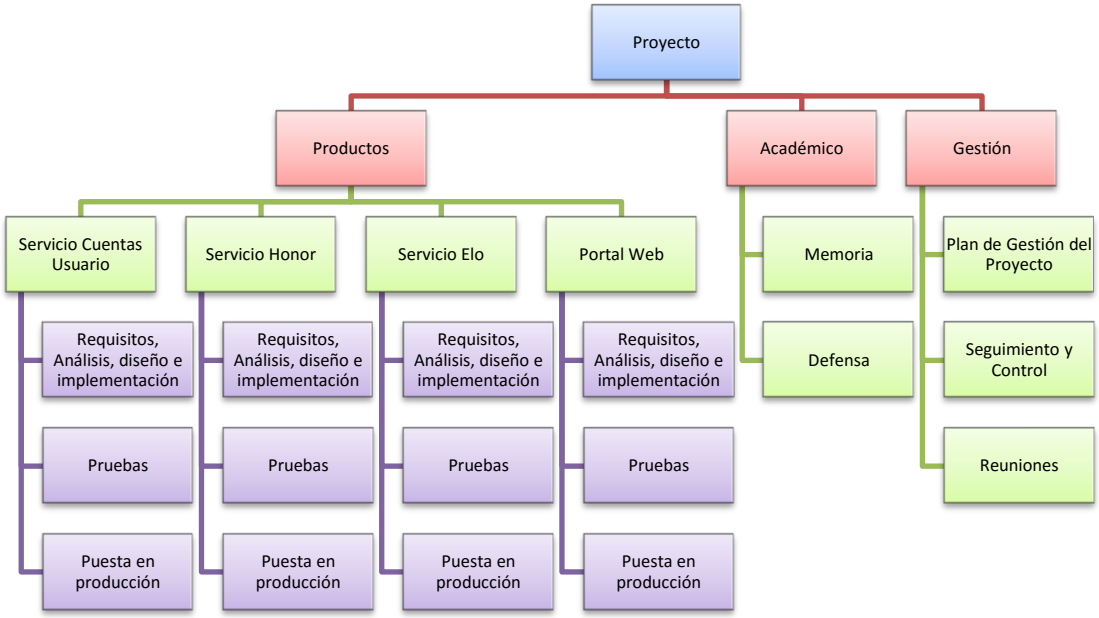


Figura 4: EDT del proyecto

# 4

# Ciclo de vida

Este capítulo expone el ciclo de vida establecido por IHEGames, cuáles son sus etapas y sus fases.

## 4.1 Descripción del ciclo de vida

El ciclo de vida de este proyecto está determinado por el ciclo de vida de los proyectos establecido por IHEGames, el cual establece una serie de fases a lo largo del desarrollo de un proyecto que se pueden repetir a lo largo de uno o varios de ciclos.

### 4.1.1 Fases en la etapa de desarrollo

El desarrollo de los proyectos de IHEGames consta de cuatro fases divididas a su vez en una o más iteraciones.

En la siguiente figura se recoge la estructura general de las fases en el desarrollo del proyecto.

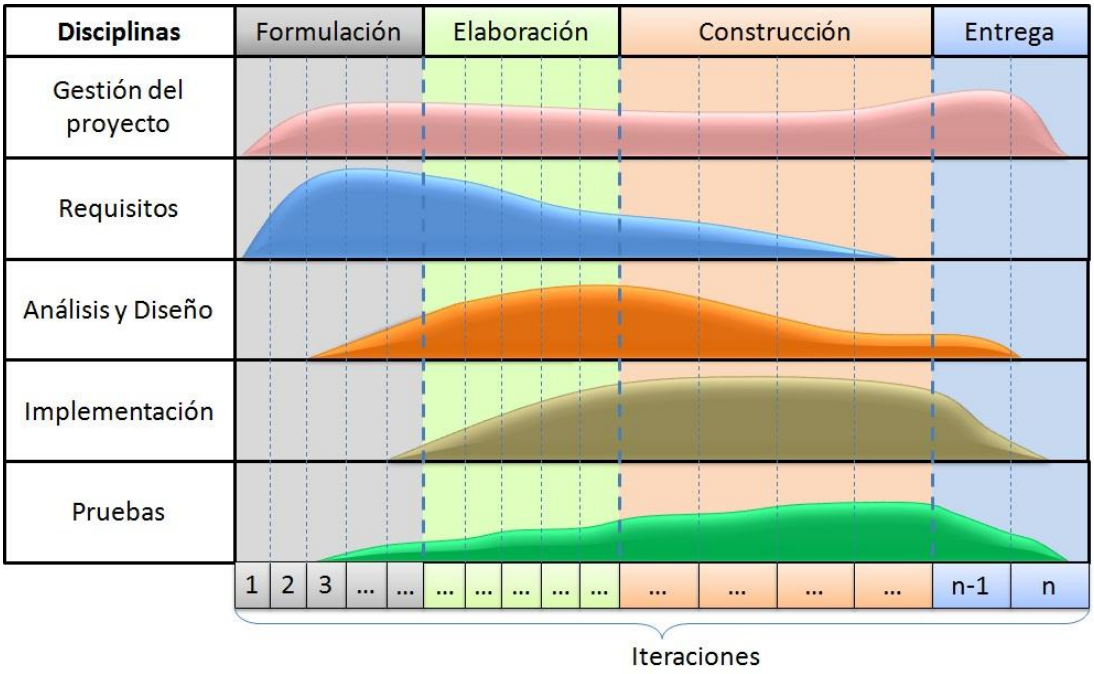


Figura 5: Fases en el desarrollo de los proyectos de IHEGames[1]

Cada fase, dentro del desarrollo de los proyectos de IHEGames, consta de unos objetivos y unos artefactos concretos que pueden llevarse a cabo dependiendo de la naturaleza del

[1] *Imagen obtenida de http://escritura.proyectolatin.org/gestion-de-proyectos-de-software/ejemplos-de-procesos/*

*2 Valores estimados dado que la tarea no ha terminado*

proyecto y que pueden refinarse en fases posteriores. A continuación se describe el objetivo de cada una de las fases y sus artefactos:

- Formulación: Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores. Se desarrollan una serie de artefactos en esta fase:
    1. Diagrama de casos de uso
    2. Especificación de requisitos.
- Elaboración: se realiza la especificación de los casos de uso que permiten definir la arquitectura base del sistema, el primer análisis del dominio del problema y se diseña la solución preliminar. Se desarrollan una serie de artefactos en esta fase:
    1. Diagrama de clases
    2. Modelo de entidad relación (E-R)
    3. Diagramas de secuencias
- Construcción: el propósito de esta fase es completar la funcionalidad del sistema. Los artefactos de esta etapa son los siguientes:
    1. Pruebas de los casos de uso desarrollado
- Entrega: El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales

Cada fase puede realizarse en una o más iteraciones. A lo largo de cada iteración se realiza una serie de tareas consecutivas agrupadas en las siguientes disciplinas: gestión del proyecto, requisitos, análisis y diseño, implementación y pruebas.

### 4.1.2 Etapas en un ciclo del proyecto

IHEGames establece que este desarrollo de cuatro fases se puede repetir a lo largo de uno o más ciclos que se descomponen a su vez en varias etapas. El objetivo de cada ciclo consiste en desarrollar una nueva versión del producto a partir de nuevas especificaciones. A partir del primer ciclo, dichas especificaciones se definen teniendo en cuenta los resultados obtenidos en pruebas con usuarios reales.

En la siguiente figura se muestran las etapas que componen un ciclo de un proyecto.


Figura 6: Etapas a lo largo de un ciclo de los proyectos de IHEGames

Al inicio de cada proyecto se definen unas especificaciones iniciales para la primera versión del producto a desarrollar. Una vez desarrollada la primera versión se procede a la validación del producto mediante pruebas con usuarios reales. De las pruebas realizadas se recogen las valoraciones de los usuarios y se definen unas nuevas especificaciones para la siguiente versión. Este proceso se continúa hasta obtener la versión final deseada.

Este proyecto concreto ha consistido en un primer ciclo, en el cual se han desarrollado unas primeras versiones del Portal D20 y los servicios de cuentas de usuario, Elo y Honor a partir de unas especificaciones iniciales y se han validado mediante pruebas con usuarios reales.

Dado que el ciclo de vida de los proyectos establecido por IHEGames no contempla la documentación académica que es inherente a este proyecto, se establece una cuarta etapa al final del ciclo (tras la validación mediante las pruebas con usuarios) para la redacción de la memoria y la defensa del proyecto.

# 5

# Windows Communication Foundation

Este capítulo expone el marco de trabajo utilizado en el desarrollo de los servicios web, con la finalidad de facilitar el entendimiento del desarrollo de los servicios web.

WCF o Windows Communication Foundation es un marco muy extenso, por ello, en este capítulo solo se exponen algunos rasgos generales que han afectado en el desarrollo del proyecto.

## 5.1 ¿Qué es?

Windows Communication Foundation es una capa que está por encima de los servicios web de ASP.NET de Windows. Su objetivo es crear aplicaciones orientadas a servicios web más seguras, ofreciendo diferentes características.

Los escenarios más comunes para su uso son:

- Servicio seguro para procesar transacciones de negocios.
- Servicio que suministre datos actuales a otros, como un informe de tráfico u otro servicio de monitoreo.
- Servicio de chat que permite a dos personas comunicarse o intercambiar datos en tiempo real.
- Una aplicación de encuestas que haga uso de uno o más servicios.

## 5.2 Características

WCF utiliza el estándar *Web Service* denominado *WS*, permitiendo el desarrollo de servicios web. WS es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Las organizaciones responsables de la arquitectura y la reglamentación de WS son Honor: El Honor de un usuario simboliza su conducta o comportamiento general que ha demostrado en partidas anteriores y permite al resto de jugadores prever el tipo de jugador que es. (Vid. Pág. 9, 10, 18, 20, 61, 64, 65)

OASIS y W3C.

Para cumplir con los estándar establecidos por WS, WCF utiliza SOA (Service-oriented architecture) el cual, establece como se envían y reciben los mensajes en el servicio.

El contenido de los mensajes intercambios en WCF está en formato XML, en el se determina unas cabeceras para los mensajes enviados así como el contenido de estos.

WCF hace uso del estándar Web Service Security (WS-Security) para que en caso de que sea necesario, pueda ser aplicado en sus comunicaciones. El estándar WS-Security, es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los servicios web. Originalmente fue desarrollado por IBM, Microsoft y VeriSign, y actualmente está desarrollado por un comité en Honor: El Honor de un usuario simboliza su conducta o comportamiento general que ha demostrado en partidas anteriores y permite al resto de jugadores prever el tipo de jugador que es. (Vid. Pág. 9, 10, 18, 20, 61, 64, 65)

OASIS. Este estándar contiene especificaciones sobre cómo debe garantizarse la integridad y seguridad en el envío de mensajes de servicios web.

WCF implementa además otros estándares web de la industrial de los servicios web, los cuales son reglamentados por Honor: El Honor de un usuario simboliza su conducta o comportamiento general que ha demostrado en partidas anteriores y permite al resto de jugadores prever el tipo de jugador que es. (Vid. Pág. 9, 10, 18, 20, 61, 64, 65)

OASIS y W3C.

Las características que provee WCF son las siguientes:

- Orientación a servicios
- Interoperabilidad entre diferentes servicios web o aplicaciones.
- Envío de mensajes mediante diferentes tipos de patrones.
- Publicación de los metadatos del servicio.
- Formalizar los tipos de clases y datos usados mediante contrato de datos.
- Tipos de seguridad.
- Múltiples formas de comunicarse.
- Envío de mensajes confiables y creación de gestión de colas de mensajes.
- Permitir la durabilidad de los mensajes, reenviando los mensajes perdidos en caso de pérdida.
- Permitir transacciones.
- Soporte para AJAX y REST.
- Permitir la extensibilidad de los servicios, permitiendo aceptar más de un protocolo para acceder al servicio.

## 5.3 Tipos de reglas disponibles (Bindings)

Los *Bindings* (reglas en castellano), especifican como se va a comunicar el servicio WCF con otras aplicaciones o servicios. En estas especificaciones se define el método de transporte, las reglas o mecanismos que se van a utilizar y la codificación de los mensajes.

- El método de transporte especifica qué tipo de protocolo se va a utilizar para el envío de mensajes.
- Las reglas o mecanismos especifican la seguridad, fiabilidad y el contexto en el que se va a comunicar el servicio WCF con los servicios o aplicaciones.
- Y finalmente la codificación de los mensajes determina como se van a codificar estos a la hora de mandarlos.

El servicio WCF tiene como mínimo una configuración *Binding* aunque puede tener más de una. Estas configuraciones pueden estar preestablecidas por WCF o pueden ser personalizadas. Las configuraciones *Bindings* disponibles por WCF se muestran a continuación:

- BasicHttpBinding
- WSHttpBinding
- WSDualHttpBinding
- WSFederationHttpBinding
- NetHttpBinding
- NetHttpsBinding
- NetTcpBinding

- NetNamedPipeBinding
- NetMsmqBinding
- NetPeerTcpBinding
- MsmqIntegrationBinding
- BasicHttpContextBinding
- NetTcpContextBinding
- WebHttpBinding
- WSHttpContextBinding

En la siguiente figura se muestran unas recomendaciones ofrecidas en un artículo de la página c-sharpcorner[12] para elegir el *Binding* adecuado, dependiendo la naturaleza del servicio WCF que se quiera desarrollar.

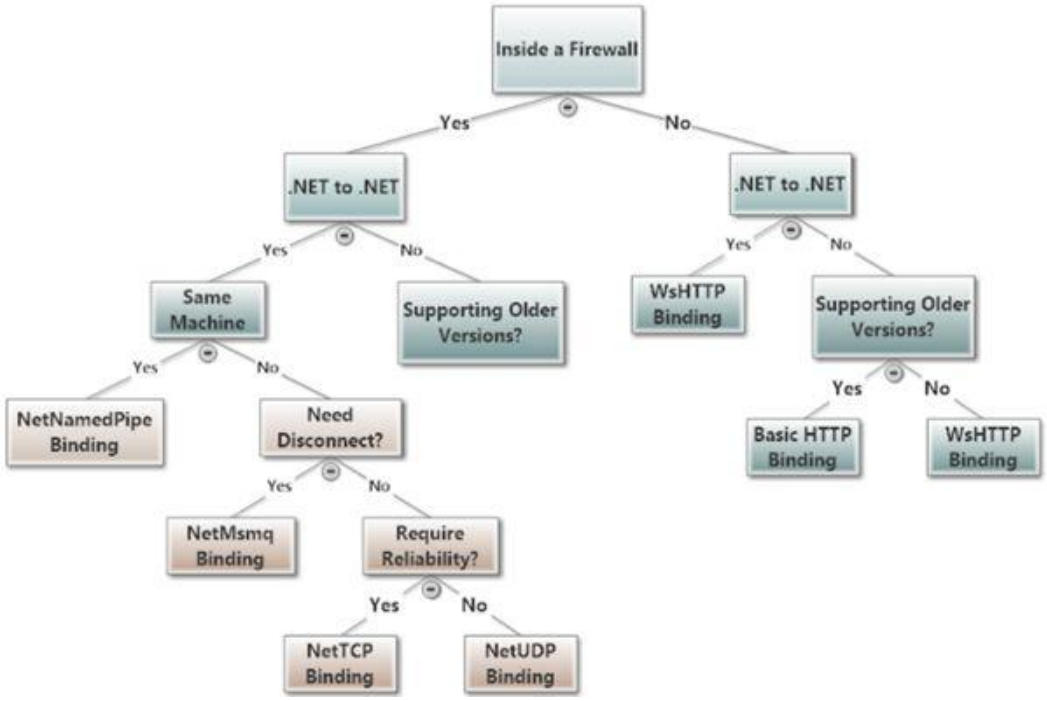


Figura 7 : Recomendaciones Binding WCF, imagen obtenida de c-sharpcorner[12]

## 5.4 Seguridad

WCF en el cumplimiento de WS-Security permite diferentes configuraciones a la hora de realizar la configuración del servicio. Cabe destacar, por un lado la seguridad de la autorización al servicio y por otro lado la seguridad de autentificación en éste.

### 5.4.1 Autentificación

La autentificación en el servicio WCF consiste en permitir el acceso al servicio WCF desde otro servicio o aplicación. A continuación se muestran las diferentes configuraciones de autentificación disponibles:

- Ninguno (None): Los mensajes no se cifran.
- Transporte (Transport): Los mensajes son cifrados usados el canal de comunicación TLS (Transport Layer Security, haciendo uso de certificados para ello.
- Mensaje (Message): Los mensajes son cifrados a nivel de mensaje usando la seguridad de mensaje, las cabeceras no son cifradas, y se hace uso de Token de seguridad para las diferentes peticiones que se realizan. En caso de múltiples peticiones, gracias a dichos Token se pueden distinguir las peticiones.
- Transporte con Credenciales de mensajes (TransportWithMessageCredential): La protección y autorización de los mensajes se basa en el canal de comunicación usando TLS y mensaje Se puede autentificar al cliente tanto por transporte mediante certificados, como mediante usuario y contraseña a nivel de mensaje. Los mensajes van doblemente encriptados, por un lado se encapsula todo el mensaje en un canal seguro TLS y por otro lado el mensaje SOAP enviado, el contenido de las etiquetas va cifrado gracias al nivel de mensaje.
- Transporte solo con Credenciales (TransportCredentialOnly): Las Credenciales son pasadas por el canal de comunicación usando TLS pero el mensaje no es encriptado. Se puede autentificar al cliente mediante los certificados utilizados, y los mensajes entre cliente y servidor van cifrados al completo.

### 5.4.2 Autorización

La autorización permite dar privilegios a los clientes, determinando que operaciones pueden llevar a cabo en el servicio. WCF soporta los siguientes tipos de autorización:

- Basado en roles (Role-based): El acceso al servicio y la realización de las operaciones se basa en roles de usuario.
- Basado en identificación (Identity based): El acceso al servicio y la realización de las operaciones se basa en las Credenciales del usuario.
- Basado en recursos (Resource based): El acceso al servicio y la realización de las operaciones se basan en la seguridad, utilizando el control de acceso al servicio, mediante una lista de control de acceso Windows Access Control Lists (ACLs).

Cuando la autorización está basada en roles, hay tres formas de determinar el rol del usuario.

- Grupos de Windows (Windows groups): Grupos de usuarios creados en el sistema Windows donde se encuentra el servicio.

- Grupos personalizados (Custom roles): crear y especificar roles personalizados para la aplicación.
- Grupos de ASP.net (ASP.NET role management): Se puede hacer uso del proveedor de roles y usos de roles de ASP.NET de una web ya creada.

# 6

# ASP.NET MVC

En este capítulo se explica brevemente en qué consiste ASP.NET MVC. Se exponen las partes relevantes que han afectado en el desarrollo del proyecto, para ofrecer una mejor comprensión en lo referente al desarrollo del portal web D20.

## 6.1 ¿Qué es ASP.NET MVC?

ASP.NET MVC es un framework para el desarrollo de páginas web en ASP.NET. Un framework es una estructura con módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

El framework ASP.NET MVC consiste en un conjunto de componentes para construir páginas web haciendo uso del modelo vista controlador:

- El modelo representa el núcleo de la aplicación, en la aplicación representa la lógica de los datos, ofreciendo métodos de almacenamiento y obtención de estos.
- La vista crea una representación, la cual puede contener datos del modelo de datos, es la parte en la que se interactúa con la aplicación. Ofrece vistas para posibilitar la interacción con la aplicación.
- El controlador gestiona la interacción con la aplicación. Hace una lectura de los datos obtenidos por las vistas y los trata acorde a los parámetros que tenga establecidos.

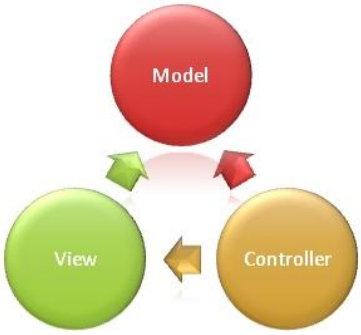


Figura 8 : Modelo Vista Controlador (Extraído de W3Schools [13])

En el proyecto, en la parte correspondiente al desarrollo del portal web D20 se hace uso de ASP.NET MVC 6, el cual se diferencia de sus predecesores por las características que se mencionan a continuación.

ASP.NET Core, es una versión modular de .NET Framework. Está diseñado para las aplicaciones desarrolladas con éste, sean portables entre plataformas como Windows, Linux y Mac. Uno de los componentes que forman ASP.NET Core es ASP.NET Identity.

ASP.NET Identity consiste en un sistema de gestión de los datos de la página web evitando así el tener que implementarlo manualmente. Gestiona los controladores y los modelos de datos que pueda tener la página web.

Para las vistas de la aplicación, ASP.NET MVC hace uso de Razor. Razor es un lenguaje de programación de ASP.NET usado para la creación de paginas dinámicas incluyendo lenguajes de programación C# o Visual Basic.

## 6.2 ASP.NET Core

ASP.NET Core es una versión modular de .NET Framework diseñada para que sea portable entre diferentes plataformas, con el objetivo de la reutilización del código y su uso compartido. Además, ASP.NET Core es de código abierto.

ASP.NET Core es modular y está disponible como paquetes más pequeños centrados en las características, lo que le da una modularidad mayor y evita la necesidad de tener todas las funcionalidades de las que éste dispone. Con ello se permite un modelo de desarrollo más ágil y da la opción de elegir las funcionalidades que se necesitan en las aplicaciones y bibliotecas que se va a desarrollar.

El código de ASP.NET Core, como ya se ha mencionado, es de código abierto, esto quiere decir que es de libre distribución y uso, y da la posibilidad a los usuarios que hagan uso del modificarlo o hacer peticiones para su modificación en las partes que considere oportunas. Para la gestión de los cambios por parte de la comunidad utiliza Git y almacena el código fuente en GitHub con el nombre de CoreFX[14].

### 6.2.1 ASP.NET Identity

Como ya se ha mencionado ASP.NET Identity pertenece a ASP.NET Core, ASP.NET Identity es un sistema ya implementado para la gestión y tratamiento de los datos en la página web. Consiste en clases llamadas *Managers* y almacenamiento de datos denominado *User Store*.

Los controladores son clases de alto nivel usadas para la realización y ejecución de operaciones en la aplicación.

Los almacenamientos de datos consisten por un lado en clases de bajo nivel que especifican como serán las entidades de datos que van a ser usadas, de las que cabe destacar clases de entidades rol y usuario. Y por otro lado especifican como se van a guardar y recuperar dichos datos.

El flujo de datos en ASP.NET Identity es el siguiente, los datos introducidos por el usuario en el portal web se reenvían al controlador correspondiente de ASP.NET Identity, éste a su vez hace uso de la clase de almacenamiento de datos correspondiente para obtener los datos que necesite. La clase de almacenamientos de datos obtiene los datos de la base de datos y se los devuelve al controlador. Finalmente el controlador evalúa los datos y ejecuta las acciones que tenga establecidas.

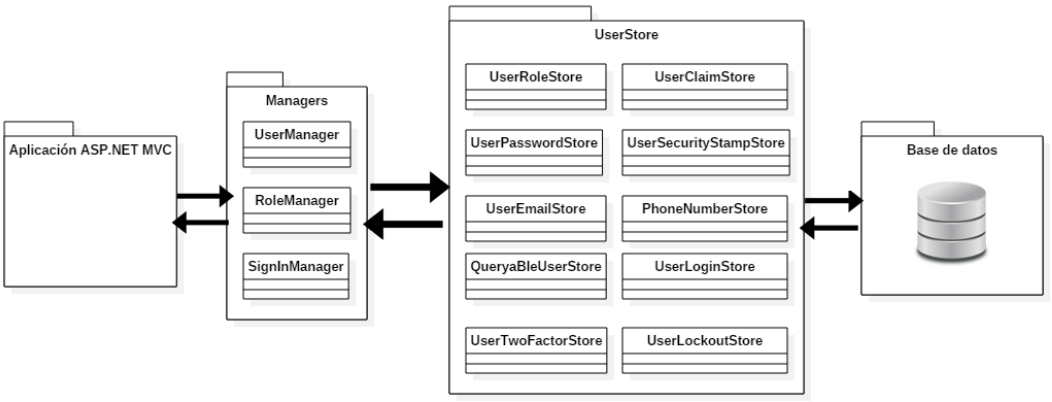


Figura 9: Arquitectura ASP.NET Identity

ASP.NET Identity permite personalizar todos los controladores denominados Managers de los cuales hace uso. Los tres controladores de los que hace uso como se puede ver en la figura anterior son: UserManager, RoleManager y SignInManager. Los tres consisten en clases abstractas, que permiten volver a ser implementadas en nuevas clases y sustituirlas en la aplicación.

Las clases relacionadas con el almacenamiento de datos vienen ya implementadas, ASP.NET Identity ofrece sus interfaces para que puedan ser fácilmente sustituidas por una implementación propia. Esto permite que los almacenamientos de datos, así como los tipos de datos que se usan en la aplicación puedan ser personalizados.

## 6.3 Controladores ASP.NET MVC

Los controladores en ASP.NET MVC son clases con funciones que gestionan las peticiones que llegan a la página web por parte de los usuarios. Las funciones encargadas de recibir y enviar las respuestas a los usuarios son denominadas acciones. Cada acción puede hacer uso de una vista y de un modelo de datos. O simplemente redireccionar hacia otra acción, dependiendo la funcionalidad de ésta. Si se desea conocer más a fondo el funcionamiento de los controladores en las páginas web MVC, Microsoft ofrece gran documentación al respecto[15].

Por defecto en las plantillas generadas de las páginas web MVC 6, la cual ha sido usada para la creación del portal D20, se crean tres controladores.

- HomeController: este controlador se encarga de gestionar las peticiones a la raíz de la página web y a las páginas About y Contact
- AccountController: es el encargado de las páginas referentes al registro y autentificación de usuarios en el portal web.
- ManageController: es el encargado de gestionar las paginas que permiten a los usuarios modificar sus datos en el portal web
-

# 7

# Servicios Web

Este capítulo expone las partes comunes de los servicios web. Por un lado se explica la arquitectura general de los tres servicios web desarrollados así como su funcionamiento. También se hace mención de los elementos comunes que componen estos tres servicios web.

## 7.1 Arquitectura de los servicios web

En la siguiente figura se muestra la arquitectura general de los servicios web de cuentas de usuario, Honor y ELO.

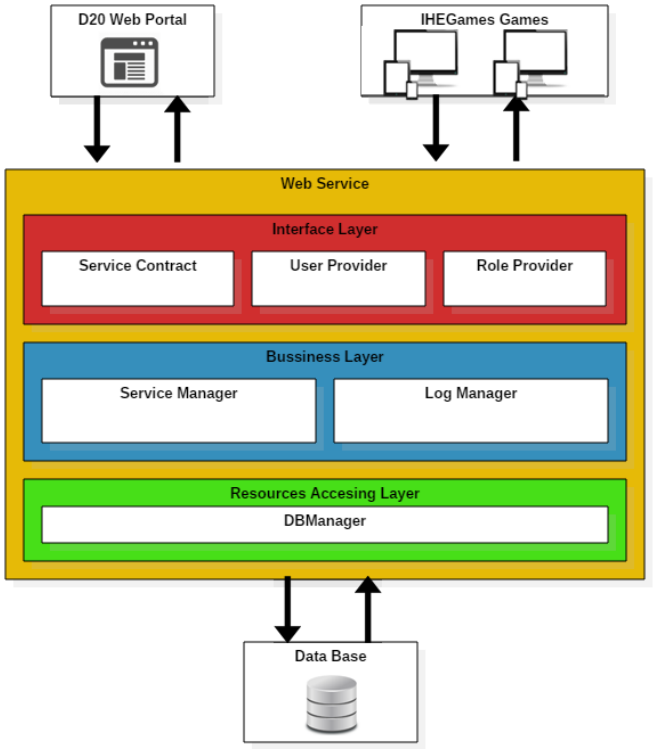


Figura 10: Arquitectura general de los servicios web

Se pueden distinguir 3 bloques generales en los servicios web, el bloque de interfaz del servicio (Interface Layer) el bloque de la lógica de negocio (Bussiness Layer) y el bloque de acceso a los datos (Resources Accesing Layer). A continuación se explica el contenido de cada bloque en los servicios web

### 7.1.1 Interface Layer.

Interface Layer está compuesto por cuatro apartados: Service Contract, Service Adapters, User Provider y Role Provider. Estos apartados definen el funcionamiento del servicio, los tipos de datos que se van a usar así como la autentificación y autorización de uso del servicio.

#### 7.1.1.1 Service Contracts

En la siguiente figura se muestran los contratos que tienen los servicios web.


Figura 11: Service contracts

- Service Contract: en el contrato del servicio en sí, se especifica los datos de la interfaz del servicio, dando a conocer a las aplicaciones cliente que funcionalidades ofrece el servicio. Las funcionalidades de cada servicio se especifican más adelante para cada servicio web mediante los casos de uso correspondientes.
- Data Contract: los contratos de datos especifican qué tipos de datos acepta recibir el servicio así como los que puede enviar. Estos datos pueden ser simples (string, int, etc.) o clases con datos más complejos. Cada servicio web desarrollado tiene sus propios contratos de datos, estos son especificados más adelante en el modelo de datos de cada servicio.
- Message Contract: el contrato de los mensajes especifica cómo se van a enviar y recibir los mensajes en los servicios web, en este caso los servicios web usan el método por defecto Request/Reply, el cual ofrece la oportunidad de realizar peticiones al servicio, ya sea de forma síncrona o asíncrona desde las aplicaciones cliente. El funcionamiento de los servicios web dada las necesidades de la aplicación web D20 y el juego K-Tan se ha decidido que utilice dicho método de funcionamiento (Request/Reply).
- Policy and binding: las políticas y configuraciones bindings, especifican el funcionamiento del servicio. Los tres servicios web funcionan de la misma manera, por un lado obligan a las aplicaciones web a utilizar Https y a utilizar certificados propios, y por otro lado tienen que enviar sus Credenciales para autentificarse en el servicio.

Para cumplir con uno de los requisitos, el cual obliga a diferenciar de alguna forma los clientes que hagan uso del servicio web y cifrar los mensajes que se reciben y envían, se ha decidió obligar a las aplicaciones cliente a hacer uso de certificados para que estos puedan ser validados en el servicio web, y a su vez, autentificar la autoridad de los mensajes entrantes. Para ello las aplicaciones cliente tienen que hacer uso del protocolo de comunicaciones WS-Security mediante el uso del binding wsHttpBinding.

La configuración de las políticas y las configuraciones Binding se encuentran en el fichero de configuración del servicio Web.Config, el cual se encuentra localizado en la raíz del servicio. En el siguiente apartado se explica con más detalle.

#### 7.1.1.2 Fichero de configuración Web.Config

Se ha fraccionado el fichero para dar una mejor explicación de cada apartado. Hay partes en las que se hace referencia al nombre servicio web y puede variar dependiendo éste, el siguiente ejemplo es del servicio de cuentas de usuario, con el nombre en la aplicación de AccountService.

| | |
|---|---|
| Descripción: | En este apartado de la configuración de los servicios se especifica el funcionamiento de servicio. El nombre que se le ha dado en la configuración al binding del servicio es ServiceWSBinding. Este nombre es necesario más adelante, para seleccionar el binding en la parte correspondiente a la definición del servicio. Y hace uso de wsHttpBinding.<br><br>En éste se especifica la seguridad del servicio web, en este caso la seguridad será mixta, por un lado a nivel de transporte mediante uso de certificados, y por otro lado a nivel de mensaje mediante las Credenciales de usuario. |

```
<bindings>
 <wsHttpBinding>
  <binding name="ServiceWSBinding">
   <security mode="TransportWithMessageCredential">
    <transport clientCredentialType="Certificate"/>
    <message clientCredentialType="UserName"/>
   </security>
  </binding>
 </wsHttpBinding>
</bindings>
```

| | |
|---|---|
| Descripción: | Aquí se define el RoleProvider encargado de la autorización del servicio.<br>Por un lado se define en providers, donde se almacena la clase encargada de los roles de los usuarios del servicio así como su nombre "customRoleProvider", Posteriormente se le dice a la aplicación que haga uso de dicho rol especificándolo en la etiqueta roleManager. |

```
<providers>
  <add name="customRoleProvider" type="Service.providers.ServiceRoleProvider,
AccountService"/>
 </providers>
<roleManager enabled="true" defaultProvider="customRoleProvider"/>
```

| | |
|---|---|
| Descripción: | En el siguiente apartado de configuración se define el comportamiento general del servicio. Por un lado se define como se puede acceder a los metadatos del servicio, bien por http o https, mediante la etiqueta serviceMetadata. Por otro el certificado que utiliza el servicio y donde se encuentra localizado mediante la etiqueta serviceCertificate. Y Finalmente como se va a comprobar las Credenciales de las aplicaciones cliente mediante la etiqueta userNameAuthentication.<br><br>El nombre que se le ha dado al comportamiento es Default. Este nombre es necesario más adelante, para seleccionar el comportamiento del servicio en la parte correspondiente a la definición del servicio. |

```
<behaviors>
 <serviceBehaviors>
  <behavior name="default">
   <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
   <serviceDebug includeExceptionDetailInFaults="false"/>
   <serviceCredentials>
    <serviceCertificate findValue="5ed77831bdd64c1fc9174dcce365525169b9e1d2"
storeLocation="LocalMachine" storeName="My" x509FindType="FindByThumbprint"/>
    <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="Service.providers.ClientAuthenticationProvide
r, AccountService"/>
   </serviceCredentials>
   <serviceAuthorization roleProviderName="customRoleProvider"
principalPermissionMode="UseAspNetRoles"/>
  </behavior>
 </serviceBehaviors>
</behaviors>
```

| | |
|---|---|
| Descripción: | En esta sección se define la configuración del servicio. Se hace uso de las configuraciones anteriormente mencionadas mediante el nombre de cada apartado. Cabe recordar que los servicios web WCF permiten que un servicio web pueda tener diferentes behavior y bindings, los cuales son definidos en la etiqueta endpoint.<br><br>En el caso de los servicios web se definen dos tipos, uno para acceder a los metadatos del servicio, es decir para que las aplicaciones cliente puedan ver que funcionalidades tiene el servicio y que clases de datos utiliza, y otro endpoint con la interfaz del servicio y el behavior y binding definido previamente. |

```
<services>
 <service behaviorConfiguration="default" name="AccountService.AccountService">
  <endpoint address="" binding="wsHttpBinding" bindingConfiguration="
ServiceWSBinding" contract="AccountService.interfaces.IAccountService"/>
  <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
 </service>
</services>
```

### 7.1.1.3 UserProvider

Los tres servicios web tienen implementados un User Provider, la funcionalidad del User Provider es comprobar las Credenciales de las aplicaciones cliente.

Dado que los datos de las Credenciales109 de los clientes son almacenados en un almacenamiento de datos Mysql, y no se hace uso de los disponibles por WCF, es necesaria la creación de un User Provider personalizado.

La clase creada para la comprobación de Credenciales se llama ClientAuthenticationProvider y hace uso de la interfaz UserNamePasswordValidator.

```
                                        ...
public class ClientAuthenticationProvider : UserNamePasswordValidator {
                                        ...
public override void Validate(string clientName, string password){
                                        ...
}
```

Las aplicaciones cliente mandan sus Credenciales y se comprueban, al igual que se comprueba si la cuenta está bloqueada. En caso de las Credenciales no sean validas o la cuenta de la aplicación cliente esté bloqueada se reporta error lanzado una excepción MessageSecurityException y se rechaza a la aplicación que intenta acceder al servicio.

A continuación se muestra el diagrama que se sigue para la comprobación de las Credenciales.

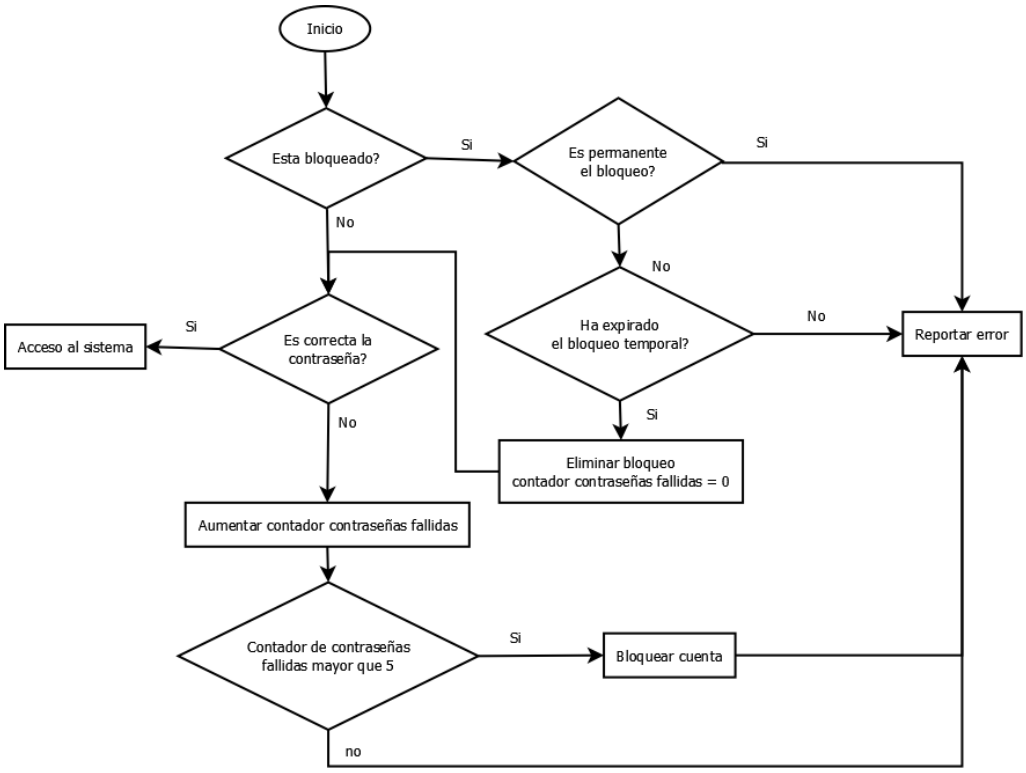


Figura 12: Diagrama de secuencias comprobación de credenciales

Para almacenar las Credenciales de las aplicaciones clientes de una forma segura, ha sido necesaria implementar una librería de cifrado propia, la cual es usada por todos los servicios web. Ahora mismo no es necesaria, pero más adelante si se quiere hacer cumplir con la LOPD, ley orgánica de protección de datos, va a ser necesaria con lo que me ha parecido oportuno el realizarla para evitar en un futuro el modificar las claves, la base de datos y el cómo se comprueban las Credenciales.

Esta librería hace uso de funciones hash que dada la contraseña del cliente y una ristra de caracteres alfanuméricos semi aleatorios, genera una nueva ristra alfanumérica ilegible que representa la contraseña. Mediante este método la contraseña modificada es irreversible, es decir, no se puede obtener la contraseña original. El método previamente descrito consiste en un cifrado Hash SHA-2.

Los detalles de cómo se ha implementado la librería de encriptación se muestra más adelante.

#### 7.1.1.4 RoleProvider

El único servicio que hace uso del Role Provider es el Servicio web de cuentas de usuario.

Esta decisión se toma a raíz de que en el alcance está establecido que algunas aplicaciones pueden editar datos de usuario y otras no. Por ello se hace uso de un Custom Role Provider para el servicio web de Cuentas de usuarios.

Al igual que el User Provider, al almacenarse los roles en un almacenamiento distinto a los predefinidos por WCF, se crea uno personalizado para las comprobaciones.

A pesar de que solo lo use el servicio de cuentas de usuario, se decide implementar en los servicios web de Elo y Honor, para tener la posibilidad de aplicar roles en caso de que más adelante sea necesario.

La clase creada es ServiceRoleProvider y hace uso de la interfaz de RoleProvider, aunque solo se implementa la función de IsUserInRole, que es la que hace uso el servicio web.

```
                                        ...
public class ServiceRoleProvider : RoleProvider {
                                        ...
public override bool IsUserInRole(string userName, string roleName) {
                                        ...
}
```

Y las funciones en el servicio que requieren un rol específico deben tener una etiqueta que denote que hacen uso de dicho rol. Por lo que las funciones del servicio que modifiquen o añadan datos se les añade al comienzo la siguiente etiqueta:

```
[PrincipalPermission(SecurityAction.Demand, Role = "edit")]
```

Y las que requieran de permisos de lectura

```
[PrincipalPermission(SecurityAction.Demand, Role = "read")]
```

Estas etiquetas solo se añaden en las funciones que son accesibles por las aplicaciones servicio, es decir, en la clase que implementa la interfaz del servicio.

### 7.1.2 Bussiness Layer

Los tres servicios web desarrollados utilizan la misma estructura de lógica de negocio.

Por un lado cada servicio tiene un controlador propio que analiza los datos entrantes y los trata en consecuencia, estos controladores se detallan en los apartados correspondientes a cada servicio.

Por otro lado hay un controlador común en los tres servicios encargado de gestionar los mensajes de auditoría (logs) generados por el servicio web. Todas las excepciones que puede generar el servicio son capturadas y tratadas por este controlador. También se encarga de almacenar los cambios que puedan llevar a cabo las aplicaciones clientes en el servicio. Las auditorias generadas son almacenadas en una base de datos de Mysql, y en caso de que ésta falle, se almacenan en el visor de registros del sistema operativo Windows.

### 7.1.3 Resources Accesing Layer

Para el acceso a los datos se ha decidido crear una interfaz IDBManager para cada servicio, con todas las operaciones necesarias para obtener y almacenar los datos que el servicio utiliza. Una vez realizada la interfaz, se ha creado un controlador que implementa la interfaz. Esta decisión se ha tomado para permitir un cambio de gestor de bases de datos en un futuro, de una forma más sencilla. De esta forma el único cambio que será necesario, es el implementar el controlador para el nuevo gestor de bases de datos.

## 7.2 Modelo de datos

En el modelo de datos del servicio se distinguen dos bloques, por un lado los que estarán incluidos en el contrato de la interfaz del servicio, haciendo públicos los datos de las clases para que las aplicaciones cliente puedan hacer uso de ellas, y por otro lado el modelo de datos propio del servicio. Estos modelos son detallados en el apartado correspondiente de cada servicio más adelante.

Algo común en los servicios web es la clase de datos ServiceDBException, la cual se ha creado para generalizar, los posibles errores que pueden surgir en un sistema de gestión de bases de datos. Esto completaría la generalidad en lo que al bloque de acceso a los datos se refiere junto con la interfaz IDBManager. Se crea esta clase debido a que cada gestor de bases de datos puede tener sus propias excepciones, con lo que los controladores que están por encima de DBManager, de alguna forma tienen que ser capaces de tratarlas. De esta forma se consigue que pese a que se cambie de gestor de base de datos, los controladores que están por encima, no tengan que cambiar los tipos de excepción capturados.

## 7.3 Estructura de los servicios web

En las siguientes figuras se muestra la estructura de los tres servicios web implementados, se explica aquí dada la similitud de estos.

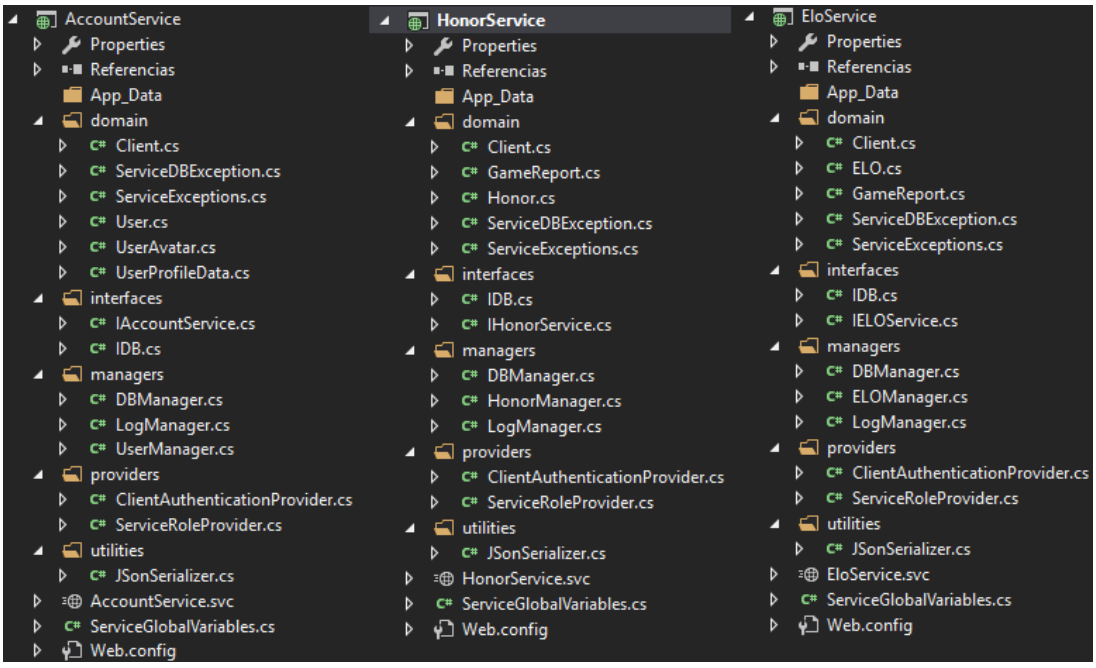


Figura 13: Estructura de los servicios web

Se ha intentado que la estructura sea lo más simple e intuitiva posible en los servicios web desarrollados. A continuación se explica la estructura:

- domain: las clases del modelo de datos son almacenados en esta carpeta.
- Interfaces: en esta carpeta se almacenan las interfaces del servicio.
- managers: los controladores del servicio son almacenados en esta carpeta.
- providers: dentro de esta carpeta se almacenan los controladores que comprueban la autorización y Credenciales de las aplicaciones que hacen uso del servicio, estos controladores se denominan providers.
- utilities: en esta carpeta se almacenan las clases que son de utilidad para el servicio.
- En la raíz del servicio se encuentra la fachada del servicio AccountService.svc, HonorService.svc o EloService.svc que implementa la interfaz del servicio. Así como el fichero web.conf mencionado en el capitulo anterior y una clase compuesta de las variables globales utilizadas en el servicio web.

## 7.4 Librerías usadas en los servicios web

### 7.4.1 Librería Encriptación

La librería de encriptación consiste en una clase, la cual hay que instanciar, y que contiene unos métodos para encriptar textos en el servicio mediante SHA-2, el cual es un conjunto de funciones hash criptográficas. También se hace uso de una clave Salt también denominada VI o vector de inicialización, el cual establece los parámetros iniciales para cifrar el texto.

La librería, dentro del conjunto de funciones hash que contiene, solo permite encriptar en SHA256, SHA384 y SHA512.

### 7.4.2 Implementación de la librería

A continuación se exponen las funciones que componen la librería de encriptación

| Función | public HashEncrypt() { |
|---|---|
| Descripción | Constructora de la clase vacía. Crea una instancia con el tipo de encriptación por defecto SHA512. |

| Función | public HashEncrypt(EncryptMethods method) { |
|---|---|
| Descripción | Constructora con parámetros. Crea una instancia con el tipo de encriptación dada. encryptMethods consiste en un enum de valores que solo contiene SHA256, SHA384 y SHA512<br>public enum EncryptMethods {SHA256, SHA384 , SHA512 }; |

| Función | public void encryptWithoutSalt(string inputText,int saltLength, out string hashedText, out string salt) { |
|---|---|
| Descripción | Dado un texto y una longitud para la clave salt, devuelve el texto cifrado junto a la clave salt. |

| Función | public string encryptWithSalt(string inputText, string salt) { |
|---|---|
| Descripción | Dado un texto y una clave salt, devuelve el texto cifrado. |

### 7.4.3 Pruebas realizadas a la librería

Para comprobar la correcta encriptación de la librería se han creado una serie de batería de pruebas. Estas pruebas consisten en UnitTest que proporciona el propio Visual Studio para testear las clases creadas.

A continuación se muestra un ejemplo de una de las pruebas realizadas a la librería. En la prueba se asigna una clave ya cifrada junto su clave salt y se intenta comprobar que desde la contraseña y la clave salt, el resultado de la clave cifrada es el mismo.

```
[TestClass]
 public class HashEncryptTestV1 {
  [TestMethod]
  public void TestValidAsymetricEncrypt() {
   string originalPassword = "Mi Contraseña";
   string manualSalt =
"IFH8AaqgYZf0QLMxnM3NyXizagAMZXxqu9onMFY2noBo8Oyuq8NehHPdrLi63bzU7qx8KiygNyg6QMB3h
HRLtAb7ah7693fhcUm9W7S4QzpLkWcKZaEAZkZioXLIfl4A";
   string manualPasswordEncrypted =
"JenPsF4cv/h60dpfkzELIbEfQzUQVM7K3VITWkkmMWFcFEAsMjg9ukJLDrmjQfWkxJTsi5omgMkyX/1Av
4p0+Q==";

   string passTmp;

   HashEncrypt crypto = new HashEncrypt (HashEncrypt.encryptMethods.SHA512);
   passTmp=crypto.encryptWithSalt(originalPassword, manualSalt);

   Assert.IsTrue (manualPasswordEncrypted.Equals(passTmp));
  }
. . .
}
```

### 7.4.4 Librería Json.NET de Newtonsoft

La función de esta librería es poder serializar correctamente algunas clases, para que puedan ser almacenadas y recuperadas fácilmente en la base de datos.

Json.NET es una librería de código abierto la cual se puede encontrar en su página web[16]. Para hacer uso de ella se ha añadido la referencia de la librería descargada y se ha creado una clase llamada JSonSerializer, la cual hace uso de la librería de Json.NET de una forma genérica, para todas las clases de los servicios web que son almacenadas directamente en la base de datos.

## 7.5 Pruebas realizadas a los servicios

Para comprobar el correcto funcionamiento de los servicios se han creado una serie de pruebas de caja negra para comprobar el correcto funcionamiento de los servicios web. Dichas pruebas han sido documentadas en un informe de pruebas realizadas, las cuales serán utilizadas de referencia en futuras versiones de los servicios web para comprobar sus funcionalidades, pudiendo modificarse en un futuro, añadiendo, eliminando o modificando las pruebas que se estimen oportunas. Estas pruebas han sido realizadas mediante la programación de pruebas unitarias utilizando Visual Studio 2015 y c#.

Cada prueba realizada se documenta de la siguiente forma:

| | | |
|---|---|---|
| **Número de prueba** | RFX.X Nombre de la función testeada | |
| **Descripción** | Descripción de la prueba realizada | |
| **Métodos/Herramientas** | Métodos o herramientas utilizadas | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | Descripción del resultados esperados | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

Tabla 1: Tabla base para las pruebas de los servicios web

Dada la extensibilidad de las pruebas realizadas, se adjuntan en anexos al final del documento.

# 8

# Servicio Web de cuentas de usuario

En este capítulo se encuentra los apartados referentes al análisis, diseño e implementación del servicio web de cuentas de usuario. Las pruebas realizadas al servicio se encuentran en los anexos a este documento.

## 8.1 Análisis y diseño

### 8.1.1 Modelo de datos

Parte de las clases en el servicio web de cuentas de usuario son visibles desde las aplicaciones cliente, ya que en algunos casos de uso, se hace uso de dichas clases, tanto para enviar, como para recibir datos.

En la siguiente figura se ve el modelo de datos resumido del servicio web de cuentas de usuario.

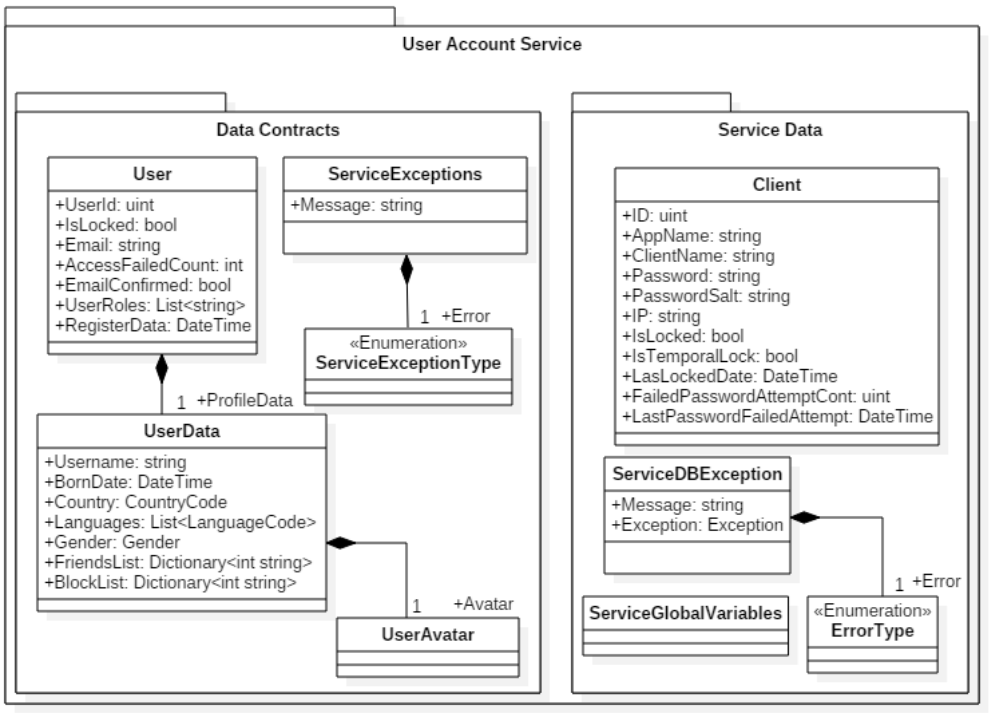


Figura 14: Modelo de datos servicio web cuentas de usuario

### 8.1.2 Esquema de la base de datos

En la siguiente figura se muestra el esquema de la base de datos usada para almacenar todos los datos del servicio de cuentas de usuario. Tanto los datos de los usuarios de las aplicaciones cliente, así como de los propios datos referentes a las aplicaciones clientes.

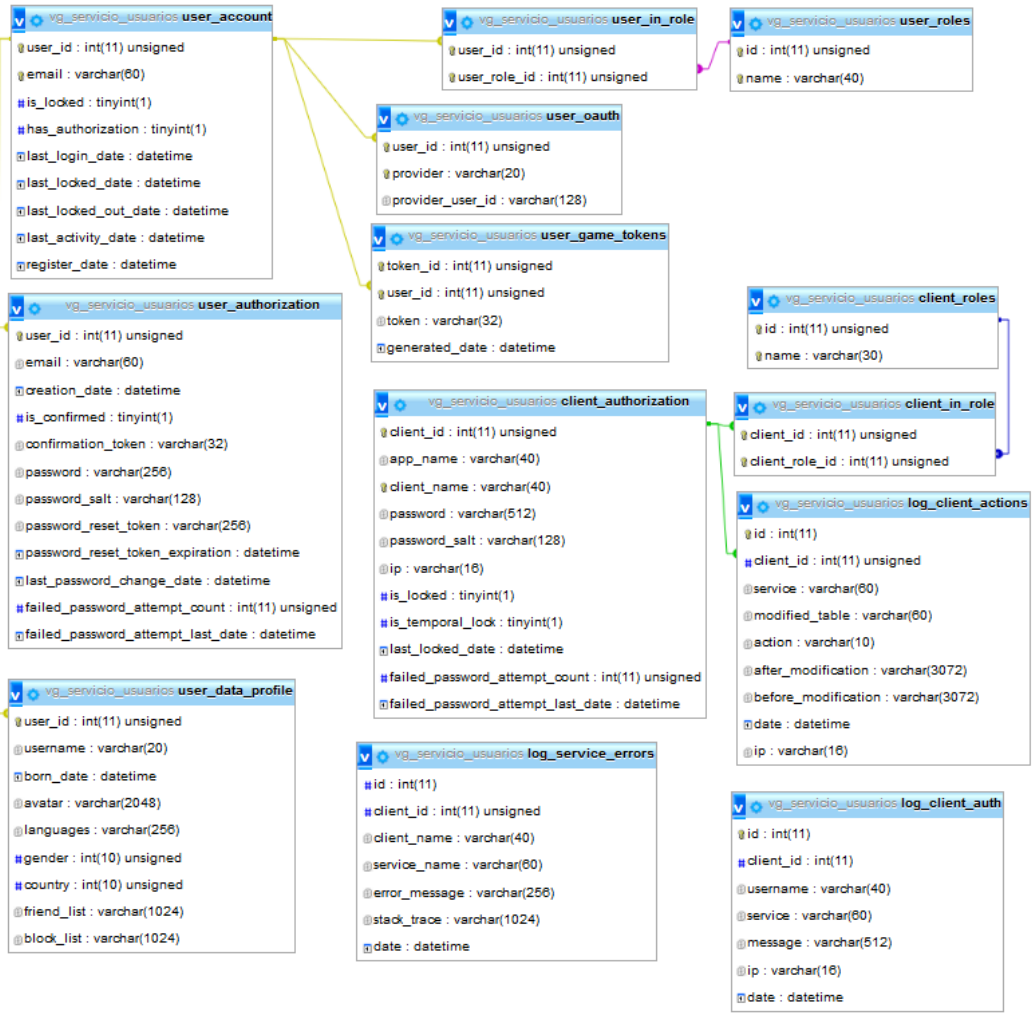


Figura 15: Esquema de la base de datos servicio cuentas de usuario

Por un lado se almacenan los datos de los usuarios partiendo de la tabla user_account y por otro lado los datos referentes a las aplicaciones que pueden acceder al servicio web partiendo de la tabla de client_authorization.

También hay varias tablas para registrar las auditorias, las cuales comienzan con el nombre de log_.

### 8.1.3 Casos de uso

Los casos de uso corresponden con las funcionalidades que ofrece la fachada del servicio. Cada uno de los casos de uso hace referencia a una función que ofrece el servicio. Se utiliza el nombre de cliente para hacer referencia a los juegos de IHEGames así como el portal D20 y a su vez servicio para hacer mención del servicio web de cuentas de usuario.

Todos los casos de uso comprueban los tipos de valores que le llegan, en caso de que sean incorrectos lanzan una excepción. En los casos de obtención, edición o eliminación de datos referentes a usuarios, se comprueba si existe el usuario y en caso de que no exista el servicio devuelve una excepción avisando de que no existe el usuario, excepto en los caso de uso de comprobación de Credenciales que devuelve False. Se omite el hacer mención de dichas comprobaciones en todos los casos de uso.

Dada la cantidad de casos de uso del servicio web de cuentas de usuario se han dividido en dos partes para facilitar su comprensión, se muestra los casos de uso mediante una imagen y a continuación las descripciones de los casos de uso.

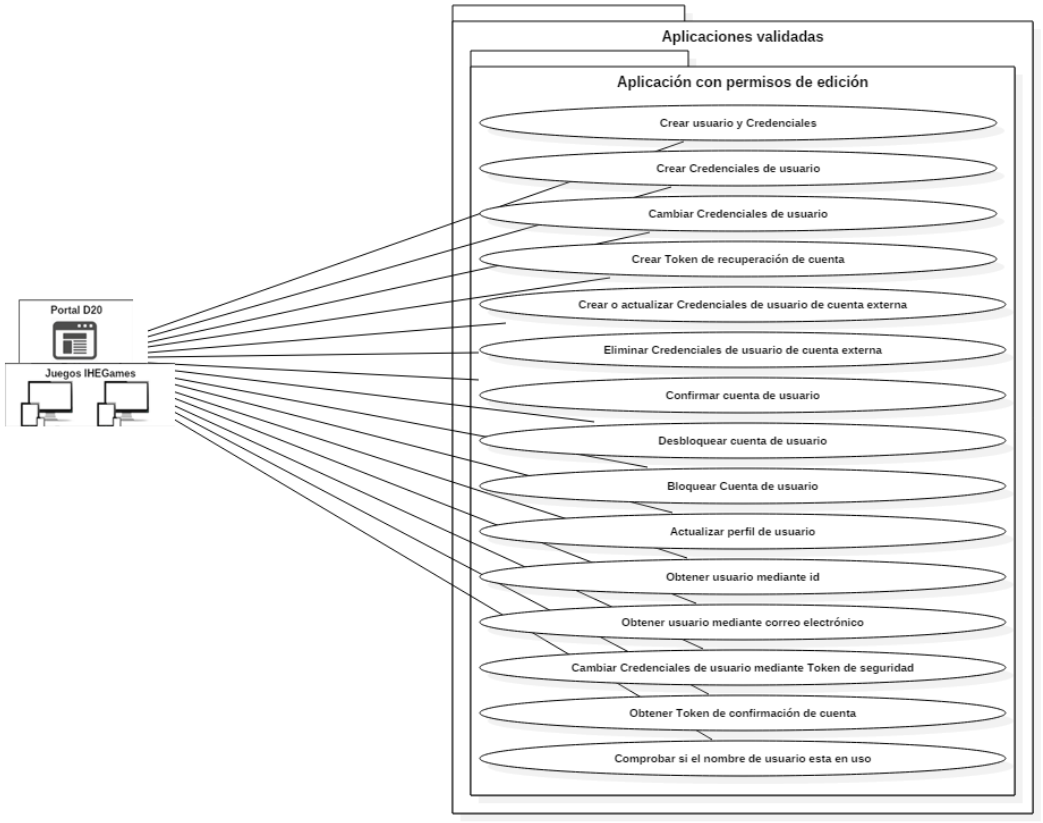


Figura 16: Primera parte de los casos de uso del servicio web de cuenta de usuarios

### Crear usuario y Credenciales

El cliente envía los datos de usuario, nombre de usuario, correo electrónico y contraseña y el servicio comprueba que el nombre de usuario y el correo electrónico no estén ya registrados en el servicio. En caso de que no lo estén genera el usuario y lo almacena en la base de datos, en caso contrario devuelve una excepción avisando de que el usuario ya existe.

### Crear Credenciales de usuario

El cliente envía los datos de correo electrónico y contraseña de un usuario. Si no existen Credenciales para dicho usuario se crean, en caso contrario devuelve una excepción avisando que ya tiene Credenciales dicho usuario.

### Cambiar Credenciales de usuario

El cliente envía las Credenciales de un usuario y una nueva contraseña. El servicio comprueba las Credenciales del usuario dado, si son correctas cambia la contraseña por la nueva contraseña y devuelve true, en caso contrario devuelve false.

### Crear Token de recuperación de cuenta

Dada una dirección de correo electrónico, el servicio crea un Token, para la recuperación de cuenta. Este código tiene una validez temporal. Una vez creado el Token se devuelve al cliente.

### Crear o actualizar Credenciales de usuario de cuenta externa

Dadas las Credenciales Oauth de un usuario compuestas por la dirección de correo electrónico del usuario y el código del usuario en el proveedor de cuentas externo, además del nombre de un proveedor de cuentas externo (Google, Facebook…). El servicio realiza unas comprobaciones que se detallan en la figura que se ve a continuación.

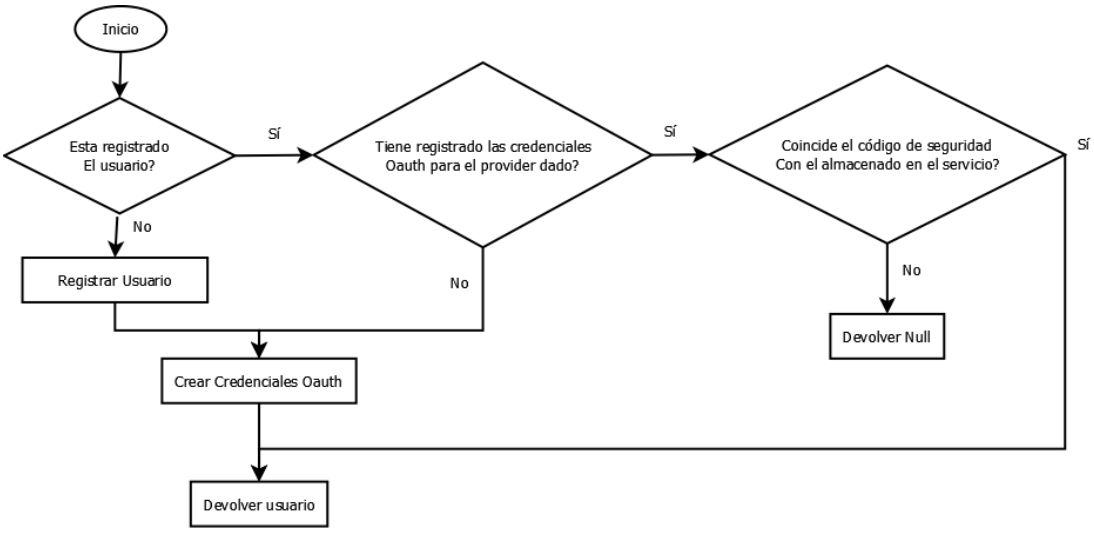


Figura 17: Crear o actualizar credenciales Oauth

### Eliminar Credenciales de usuario de cuenta externa

Dadas las Credenciales Oauth de un usuario y el nombre del proveedor de cuentas externo, el servicio las elimina de la base de datos. En caso de que el usuario no tenga Credenciales Oauth para el proveedor de cuentas externo dado, devuelve una excepción.

### Confirmar cuenta de usuario

Dado una dirección de correo electrónica y un Token de confirmación de cuenta. El servicio comprueba que el Token coincida con la cuenta dada, en caso satisfactorio cambia el estado de la cuenta a cuenta validada y devuelve true. En caso contrario devuelve false.

### Desbloquear cuenta de usuario

Dada una dirección de correo electrónica, el servicio cambia el estado de la cuenta a la que hace referencia al estado de bloqueada.

### Bloquear cuenta de usuario

Dada una dirección de correo electrónica, el servicio cambia el estado de la cuenta a la que hace referencia al estado de desbloqueada.

### Actualizar perfil de usuario

Dados los datos de perfil de un usuario y la dirección de correo electrónica del usuario, el servicio actualiza los datos de perfil del usuario en la base de datos.

### Obtener usuario mediante id

Dada un número de identificación id, el servicio devuelve un objeto User con los datos del usuario pedido.

### Obtener usuario mediante correo electrónico

Dada una dirección de correo electrónica, el servicio devuelve un objeto User con los datos del usuario pedido.

### Cambiar Credenciales de usuario mediante Token

El servicio recibe una dirección de correo electrónico, un Token de seguridad y una nueva contraseña. Si el Token de seguridad coincide con el de la cuenta a la que hace referencia el correo electrónico, y no ha expirado el tiempo de uso del Token, el servicio modifica las Credenciales de la cuenta con la nueva contraseña y devuelve true. En caso contrario devuelve false.

### Obtener Token de confirmación de cuenta

Dada una dirección de correo electrónico, el servicio devuelve el Token de confirmación de cuenta para la cuenta dada.

### Comprobar si el nombre de usuario está en uso

Dado un nombre de usuario, el servicio comprueba si algún usuario hace uso de dicho nombre, en caso de que así sea devuelve true, en caso contrario false.

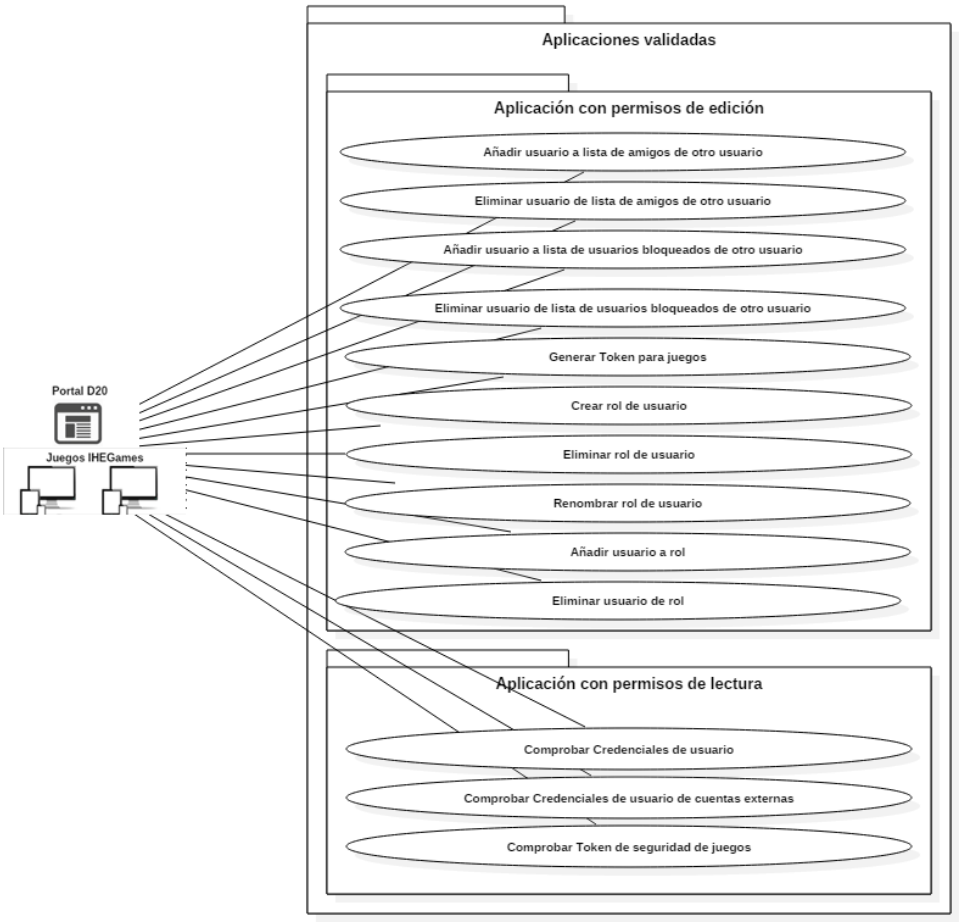


Figura 18: Segunda parte de los casos de uso del servicio web de cuenta de usuarios

### Añadir usuario a la lista de amigos de otro usuario

El servicio añade a la lista de amigos de un usuario dado otro usuario. En caso de que éste último se encuentre en su lista de usuarios bloqueados, el servicio quitara al usuario de dicha lista antes de añadirlo a la lista de amigos.

### Eliminar usuario de la lista de amigos de otro usuario

El servicio elimina de la lista de amigos del usuario dado, el otro usuario pasado como parámetro.

### Añadir usuario a la lista de usuarios bloqueados de otro usuario

El servicio añade a la lista de usuarios bloqueados de un usuario dado otro usuario. En caso de que éste último se encuentre en su lista de amigos, el servicio quitara al usuario de dicha lista antes de añadirlo a la lista de usuarios bloqueados.

### Eliminar usuario de lista de usuarios bloqueados de otro usuario

El servicio elimina de la lista de usuarios bloqueados del usuario dado, el otro usuario pasado como parámetro.

### Generar Token de seguridad para juegos

Dado una id de usuario que hace referencia a una cuenta de usuario, el servicio crea un Token de juego con una validez limitada, y lo devuelve al cliente.

### Crear rol de usuario

Dado un nombre de rol, el servicio crea dicho rol. En caso de que exista devolverá una excepción.

### Eliminar rol de usuario

Dado un nombre de rol, el servicio elimina dicho rol. En caso de que el rol pertenezca a una lista de roles protegidos, el rol no será eliminado. En caso de que el rol no exista, no se hace nada.

### Renombrar rol de usuario

Dado un nombre de rol y un nuevo nombre, el servicio cambia el nombre del rol. Si el nombre del actual rol o el nuevo nombre pertenecen a la lista de roles protegidos, el servicio no hace nada.

### Añadir usuario a rol

Dada una id de cuenta de usuario y un nombre de rol, el servicio añade a dicho usuario al rol dado como parámetro.

### Eliminar usuario de rol

Dada una id de cuenta de usuario y un nombre de rol, el servicio elimina a dicho usuario del rol pasado como parámetro. En caso de que el usuario no perteneciese al rol dado, no se hace nada.

#### Comprobar Credenciales de usuario

El cliente envía las Credenciales de usuario al servicio compuestas bien por el correo electrónico y contraseña del usuario, o el nombre de usuario y la contraseña. El servicio comprueba las Credenciales y en caso de que sean correctas se devuelve el usuario.

#### Comprobar Credenciales de usuario de cuentas externas

Dada las Credenciales de usuario de una cuenta externa, el servicio las comprueba y si coincide con las que tiene almacenadas en la base de datos, devuelve los datos del usuario, en caso de que no coincidan devuelve Null.

Si no existe el usuario, el servicio genera un nuevo usuario con las Credenciales dadas y lo devuelve.

#### Comprobar Token de seguridad de juegos

El servicio recibe una id de usuario y un Token de juego, en caso de que coincidan devuelve un objeto User con los datos del usuario, en caso contrario devuelve false.

## 8.2 Implementación

### 8.2.1 Controladores del servicio

El servicio web de cuentas de usuario contiene dos controladores principales para tratar los datos que llegan al servicio, y otro controlador encargado de gestionar la obtención y almacenamiento de los datos.

- UserManager: es el controlador del servicio, el cual es una clase estática.
    - Una vez el servicio ha validado que los datos que llegan son correctos, hace uso de las funciones implementadas. Dependiendo la funcionalidad a la que se ha accedido, lanzara una petición al controlador de la base de datos para pedir los datos pertinentes, o para actualizarlos.
    - En caso de que algún proceso falle, lanza una excepción o devuelve un error. A su vez, dependiendo la gravedad del error, hace una llamada al controlador de LogManager para que registre dicho error
    - Las funciones que impliquen cambios en la base de datos, como podrían ser modificaciones en los datos de los perfiles de los usuarios, en las credenciales etc. también hacen uso de la clase LogManager para registrar los cambios realizados por parte del cliente.
- LogManager: es el controlador encargado de registrar los fallos que se pueden generar en el servicio y los cambios realizados por las aplicaciones cliente, es una clase estática, la cual hace uso de DBManager para almacenar los reportes que

genera. En caso de que falle la base de datos, crea un error en el registro del sistema operativo Windows.

- DBManager: Es una clase que contiene los métodos para acceder al sistema de gestión de base de datos Mysql, para obtener, modificar o eliminar los datos que haga falta.

# 9

# Servicio Web de Honor

En este capítulo se encuentran los apartados referentes al análisis, diseño e implementación del servicio web de Honor. La principal característica de este servicio es el cálculo del valor de Honor de los usuarios, el cual simboliza la conducta y comportamiento general de los usuarios y es calculado mediante formulas propias de IHEGames.

Las pruebas realizadas al servicio se encuentran en los anexos a este documento.

## 9.1 Análisis y diseño

### 9.1.1 Modelo de datos

Parte de las clases en el servicio web de Honor son visibles desde las aplicaciones cliente, ya que en algunos casos de uso, se hace uso de dichas clases, tanto para enviar como para recibir datos.

En la siguiente figura se ve el modelo de datos del servicio web de Honor.

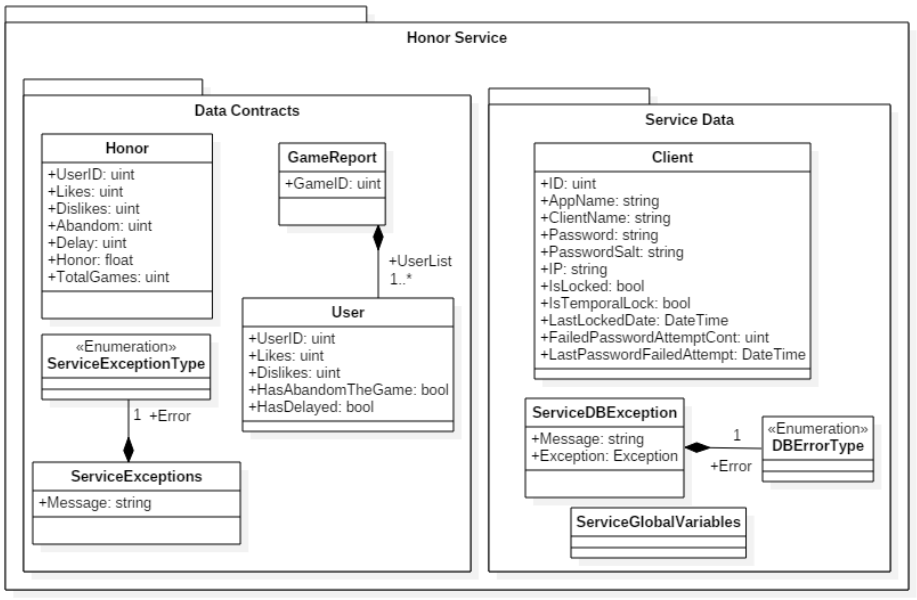


Figura 19: Modelo de datos servicio de Honor

El servicio web de Honor, en algunas de sus funciones recibe un objeto de tipo GameReport. En otras de sus funciones envía objetos de tipo Honor de los usuarios pedidos. Y al igual que el resto de servicios web hace uso de la clase ServiceExceptions para reportar errores en el servicio o en la petición de los datos.

El servicio web de Honor también hace uso de la clase Client y de la clase de ServiceDBException para capturar las excepciones que pueda lanzar el gestor de bases de datos. Y finalmente una clase que contienen las variables globales usadas en el servicio web de Honor.

### 9.1.2 Esquema de la base de datos

En la siguiente figura se muestra el esquema de la base de datos usada para almacenar todos los datos del servicio web de Honor. Tanto los datos de los usuarios de las aplicaciones cliente así como de las aplicaciones clientes.

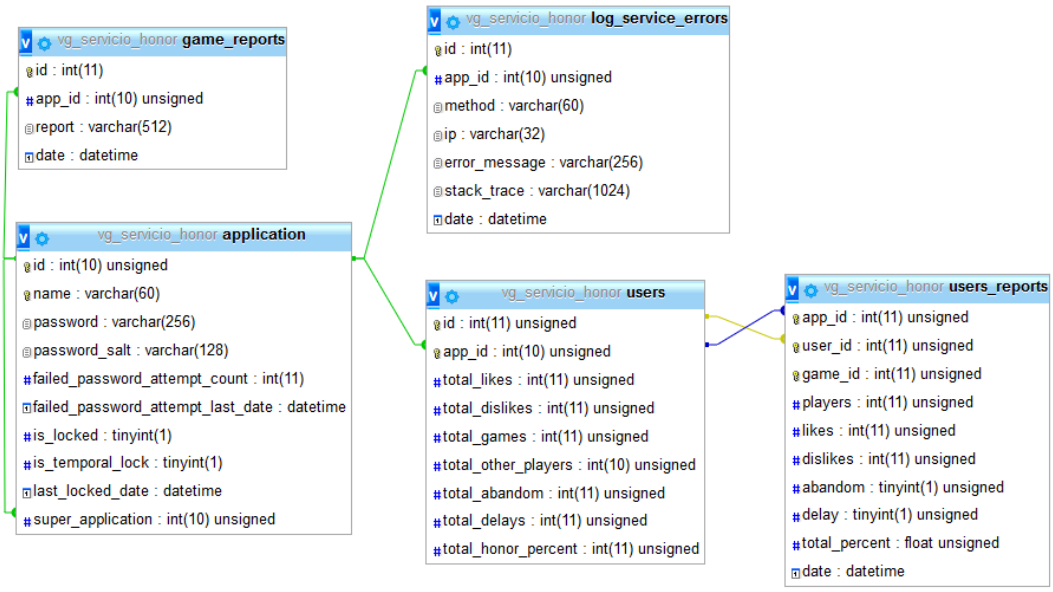


Figura 20: Esquema de la base de datos del servicio web de Honor

En el esquema superior se puede ver como cada aplicación tiene sus propios usuarios, estos usuarios a su vez pueden estar en otras aplicaciones. Para cálculos de medias ponderadas, se decide establecer una relación entre aplicaciones padre e hijo, es decir. Una aplicación puede tener más de una aplicación hijo, pero solo una aplicación padre.

En el esquema de la base de datos también se puede ver cómo están relacionados los usuarios de la aplicación con los reportes de los juegos.

Como en los demás servicios web, hay un log para los errores que puedan surgir en el servicio, pero en este caso dado a que no es un servicio critico, no se registran los intentos de acceso en un log.

### 9.1.3 Casos de uso

A continuación se muestran los casos de uso implementados en el servicio web de Honor.

Estos casos de uso son de la fachada del servicio, cada uno de los casos de uso hace referencia a una función que ofrece el servicio.

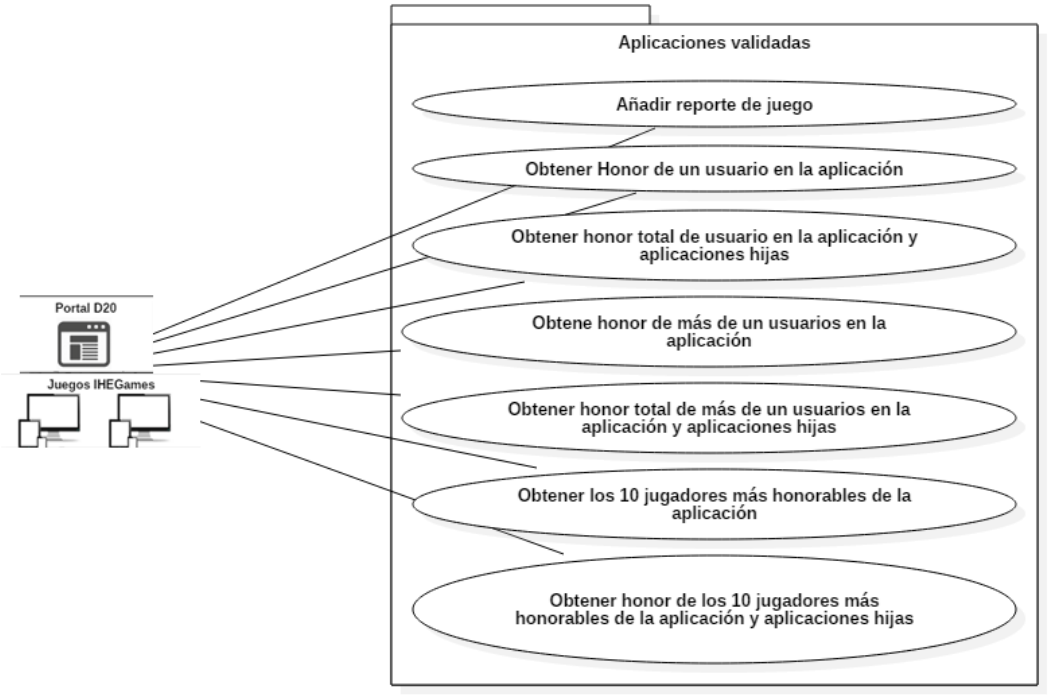


Figura 21: Casos de uso del servicio web de Honor

Para implementar dichos casos de uso se ha creado la interfaz del servicio y posteriormente se ha implementado.

A continuación se explican los casos de uso, se utiliza el nombre de cliente para hacer referencia a los juegos de IHEGames así como el portal D20 y a su vez servicio para hacer mención del servicio web de Honor.

#### Añadir reporte de juego

El servicio recalcula el valor de Honor de los usuarios que se encuentren en el reporte dado y guarda los cambios en la base de datos. En caso de que no existan los usuarios los crea.

Una vez modificados los cambios de los usuarios, el servicio serializa el reporte mediante el uso de la clase JSonSerializer y lo almacena en la base de datos.

### Obtener Honor de un usuario en la aplicación

El servicio devuelve un objeto Honor del usuario pedido. En caso de que el usuario no exista devuelve un objeto Honor con unos valores preestablecidos.

### Obtener Honor total de usuario en la aplicación y aplicaciones hijas

El servicio devuelve un objeto Honor con la media de los valores que componen el objeto Honor en dicha aplicación, y todas las aplicaciones hijas a ésta.

### Obtener Honor de más de un usuario en la aplicación

El servicio devuelve una lista de objetos de Honor con los valores de Honor de los usuarios pedidos. En caso de que alguno de los usuarios no exista, el valor de Honor de dicho usuario será el preestablecido.

### Obtener Honor total de más de un usuario en la aplicación y aplicaciones hijas

El servicio devuelve una lista de objetos Honor con la media de los valores que componen el objeto Honor en dicha aplicación, y todas las aplicaciones hijas a ésta de cada uno de los usuarios pedidos.

### Obtener los 10 jugadores más honorables de la aplicación

El servicio devuelve una lista de objetos Honor, con los 10 jugadores con un valor de Honor más elevado.

### Obtener Honor de los 10 jugadores más honorables de la aplicación y aplicaciones hijas.

El servicio devuelve una lista de objetos Honor, con los 10 jugadores con un nivel de Honor más elevado calculado mediante la media aritmética de estos en la aplicación y aplicaciones hijas.

## 9.2 Implementación

### 9.2.1 Controladores del servicio

El servicio web de Honor contiene dos controladores principales para tratar los datos que llegan al servicio, y otro controlador encargado de gestionar la obtención y almacenamiento de los datos.

- HonorManager: es el controlador del servicio, el cual es una clase estática.
    - Una vez el servicio ha validado que los datos que llegan son correctos, hace uso de las funciones implementadas. Dependiendo la funcionalidad a la que se ha accedido, lanzara una petición al controlador de la base de datos para pedir los datos pertinentes, o para actualizarlos.
    - En caso de que algún proceso falle, lanza una excepción o devuelve un error. A su vez, dependiendo la gravedad del error, hace una llamada al controlador de LogManager para que registre dicho error
- LogManager: es el controlador encargado de registrar los fallos que se pueden generar en el servicio y los cambios realizados por las aplicaciones cliente, es una clase estática, la cual hace uso de DBManager para almacenar los reportes que genera. En caso de que falle la base de datos, crea un error en el registro del sistema operativo Windows.
- DBManager: Es una clase que contiene los métodos para acceder al sistema de gestión de base de datos Mysql, para obtener, modificar o eliminar los datos que haga falta.

# 10

# Servicio Web de Elo

En este capítulo se encuentran los apartados referentes al análisis, diseño e implementación del servicio web de Elo. El servicio web de Elo se caracteriza por el cálculo Elo, el nivel de Experiencia de los usuarios. Para el cálculo del nivel Elo se utiliza la misma fórmula utilizada en el cálculo de nivel de los jugadores en el ajedrez[4], sin embargo el cálculo del nivel de Experiencia se realiza mediante formulas propias de IHEGames.

Las pruebas realizadas al servicio se encuentran en los anexos a este documento.

## 10.1 Análisis y diseño

### 10.1.1 Modelo de datos

Parte de las clases en el servicio web de Elo son visibles desde las aplicaciones cliente, ya que en algunos casos de uso, se hace uso de dichas clases, tanto para enviar, como para recibir datos.

En la siguiente figura se ve el modelo de datos resumido del servicio web de Elo.

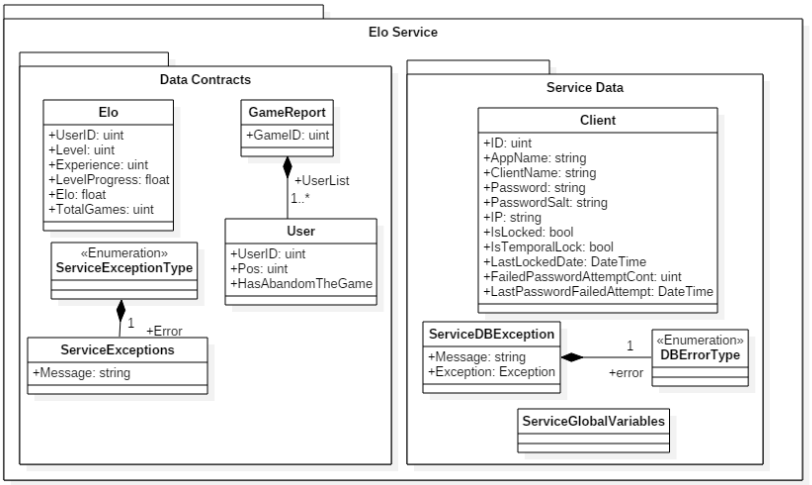


Figura 22: Modelo de datos servicio Elo

El servicio web de Elo, en algunas de sus funciones recibe un objeto de tipo GameReport. En otras de sus funciones envía objetos de tipo Elo de los usuarios pedidos. Y al igual que el resto de servicios web hace uso de la clase ServiceExceptions para reportar errores en el servicio o en la petición de los datos.

El servicio web de Elo también hace uso de la clase client y de la clase de ServiceDBException para capturar las excepciones que pueda lanzar el gestor de bases de datos. Y finalmente una clase que contienen las variables globales usadas en el servicio de Elo.

### 10.1.2 Esquema de la base de datos

En la siguiente figura se muestra el esquema de la base de datos usada para almacenar todos los datos del servicio web de Elo. Tanto los datos de los usuarios de las aplicaciones cliente así como de las aplicaciones clientes.

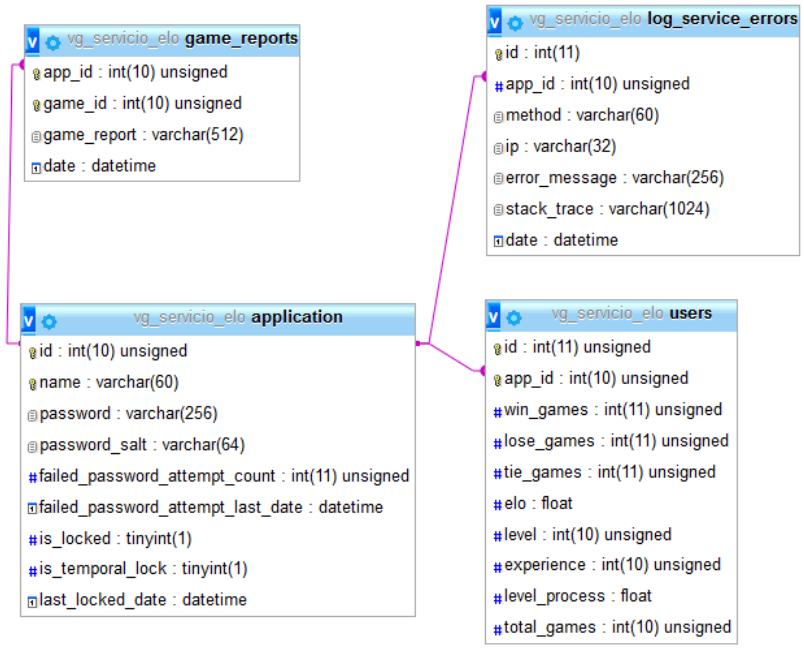


Figura 23: Esquema de la base de datos servicio web Elo

En este caso, al contrario que en el servicio web de Honor, una aplicación no tiene aplicaciones hijas. Los reportes que mandan las aplicaciones son guardados, para que más adelante, en futuras versiones, puedan ser consultados, o incluso si en algún momento se manda un reporte erróneo, se puedan recuperar los datos de los usuarios hasta un estado no erróneo.

### 10.1.3 Casos de uso

A continuación se muestran los casos de uso implementados en el servicio web de Elo.

Estos casos de uso son la fachada del servicio, cada uno de los casos de uso hace referencia a una función que ofrece el servicio.

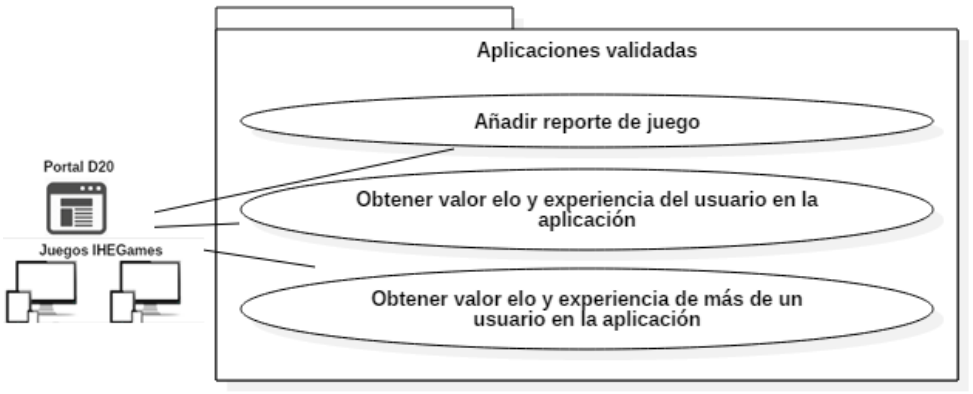


Figura 24: Casos de uso servicio web Elo

Para implementar dichos casos de uso se ha creado la interfaz del servicio y posteriormente se ha implementado.

A continuación se explican los casos de uso, se utiliza el nombre de cliente para hacer referencia a los juegos de IHEGames así como el portal D20 y a su vez servicio para hacer mención del servicio web de Elo.

### Añadir reporte de juego

El servicio actualiza los valores de Experiencia, nivel y Elo para los usuarios que se encuentren en el reporte del juego. Si los usuarios no existen en el servicio, es decir, no estaban previamente creados, el servicio los crea con unos valores por defecto, y actualiza dichos valores acorde al reporte de juego obtenido.

A su vez el servicio guarda el reporte de juego serializado mediante el uso clase JSonSerializer.

### Obtener valor Elo y Experiencia del usuario en la aplicación

El servicio devuelve un objeto Elo con los valores de Elo, Experiencia y nivel del usuario pedido en la aplicación cliente que manda la petición

### Obtener valor Elo y Experiencia de más de un usuario en la aplicación

El servicio devuelve una lista de objetos Elo con los valores de Elo, Experiencia y nivel de los usuarios pedidos en la aplicación cliente que manda la petición

## 10.2 Controlador del servicio

El servicio web de Elo contiene dos controladores principales para tratar los datos que llegan al servicio, y otro controlador encargado de gestionar la obtención y almacenamiento de los datos.

- EloManager: es el controlador del servicio, el cual es una clase estática.
    - Una vez el servicio ha validado que los datos que llegan son correctos, hace uso de las funciones implementadas. Dependiendo la funcionalidad a la que se ha accedido, lanzara una petición al controlador de la base de datos para pedir los datos pertinentes, o para actualizarlos.
    - En caso de que algún proceso falle, lanza una excepción o devuelve un error. A su vez, dependiendo la gravedad del error, hace una llamada al controlador de LogManager para que registre dicho error
- LogManager: es el controlador encargado de registrar los fallos que se pueden generar en el servicio y los cambios realizados por las aplicaciones cliente, es una clase estática, la cual hace uso de DBManager para almacenar los reportes que genera. En caso de que falle la base de datos, crea un error en el registro del sistema operativo Windows.
- DBManager: Es una clase que contiene los métodos para acceder al sistema de gestión de base de datos Mysql, para obtener, modificar o eliminar los datos que haga falta.

# 11

# Portal Web D20

En este capítulo se muestran las herramientas, librerías y los cambios necesarios en la plantilla web de ASP.NET MVC6 proporcionada en Visual Studio 2015, para cumplir con el alcance establecido en el proyecto.

## 11.1 Librerías y Herramientas utilizadas

### 11.1.1 WCF Connected Service

En Asp.Net 5, en el entorno de desarrollo de Visual Studio 2015, no está disponible la opción de añadir la referencia de un servicio WCF. Con lo que había dos opciones, crear las referencias manualmente, lo cual es poco productivo, ya que cualquier cambio en los servicios obliga a tener que cambiar el código de las referencias con todo lo que ello conlleva. Y la otra opción era instalar una utilidad que sea capaz de gestionar las referencias de los servicios WCF en el proyecto. Esta herramienta es WCF Connected Service[17].

Para poder hacer uso de ella, hay que instalarla en Visual Studio 2015 mediante la herramienta que ofrece Visual Studio 2015 para la gestor de paquetes llamada Nuget. Para más detalles consultar la documentación expuesta en la página de la herramienta [17].

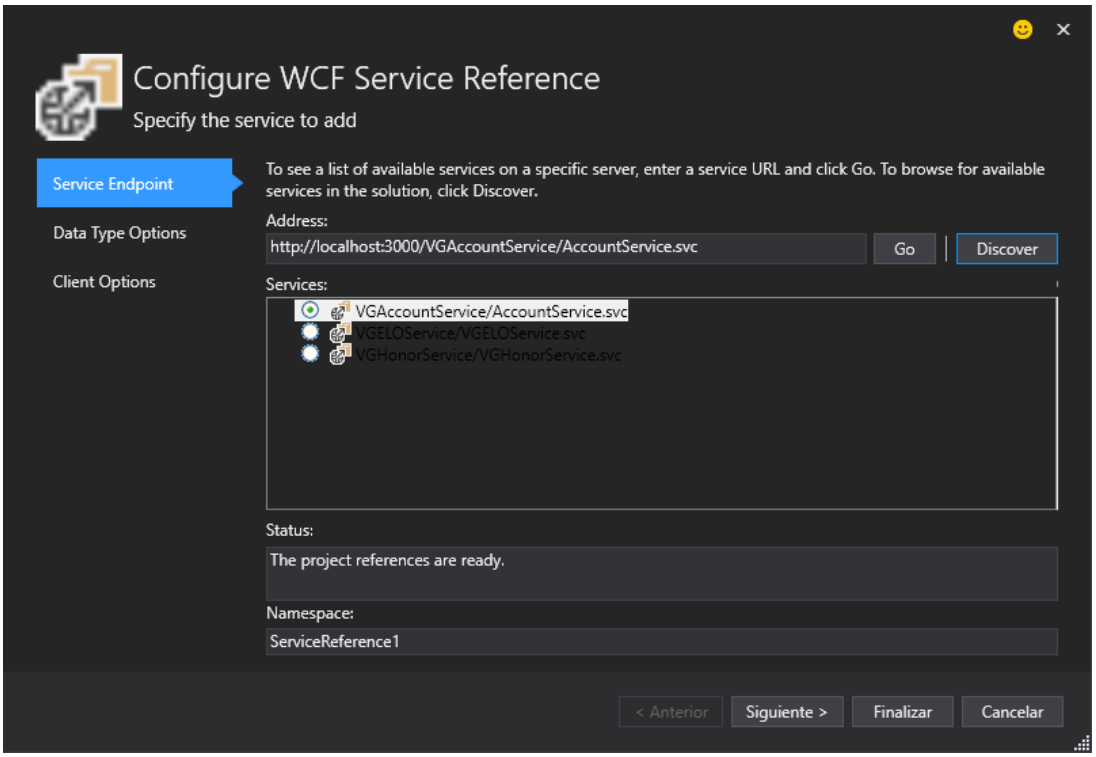


Figura 25: Imagen añadir servicio WCF en ASP.NET MVC

### 11.1.2 Librerías de los proveedores de cuentas externas

Para que la página web de ASP.NET MVC 6 haga uso de proveedores de cuentas externas hay que instalar mediante Nuget las librerías necesarias para hacer uso de las cuentas externas. Como se da servicio a cuentas de usuario de Google, Facebook y Twitter se instalan las tres librerías siguientes:

- Microsoft.AspNet.Authentication.Facebook
- Microsoft.AspNet.Authentication.Google
- Microsoft.AspNet.Authentication.Twitter

El uso y configuración de las librerías se explica más adelante en el apartado de configuración de la página web.

### 11.1.3 Librerías necesarias para la implementación del portal multilenguaje

Una de las decisiones tomadas a la hora de implementar la web, era obtener las traducciones a través de ficheros .resource almacenados en el portal web.

Para poder hacer uso de los ficheros .resource, es necesario instalar la librería System.Resources.ResourceManager la cual está disponible desde el gestor de paquetes Nuget.

## 11.2 Análisis y diseño

### 11.2.1 Casos de uso

A continuación se muestran los casos de uso implementados en el portal web D20.

En este caso, el actor, a diferencia de los servicios web es el usuario que hace uso de los servicios proporcionados por el portal web D20. Se distinguen dos tipos de usuarios. El primero, usuario anónimo, hace referencia a los usuarios que acceden al portal y no se han autentificado. El segundo hace referencia a los usuarios que previamente se han autentificado.

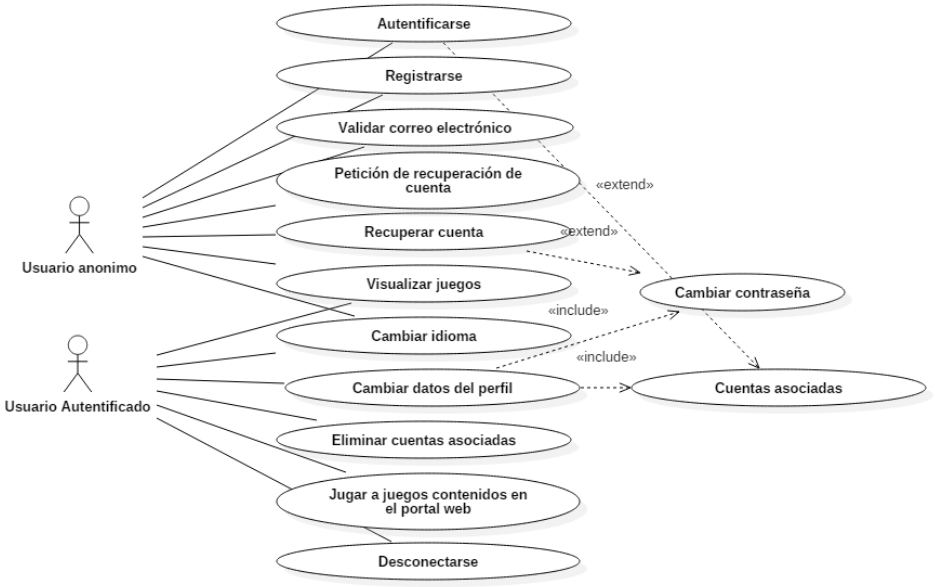


Figura 26: Casos de uso portal web D20

A continuación se explican los casos de uso implementados:

### Autentificarse

El usuario introduce sus Credenciales o hace uso de una cuenta asociada. Tras ello el portal web D20 haciendo uso del servicio de cuentas de usuario las comprueba. En caso de que sean correctas, el portal web D20 cambia el estado del usuario a usuario autentificado. En caso contrario devuelve un aviso de “usuario o contraseña incorrectas”.

### Registrarse

El usuario introduce un nombre de usuario, dirección de correo electrónico y una contraseña. Tras introducir los datos y enviarlos al portal web D20, éste haciendo uso del servicio web de cuentas de usuario comprueba que la dirección de correo electrónico y el nombre de usuario no estén registrados.

En caso de que no exista ningún usuario con el correo electrónico dado ni el nombre de usuario. El portal web D20 envía un email con los pasos de confirmación de cuenta al usuario, y devuelve un aviso de que la cuenta ha sido satisfactoriamente creada.

En caso de que exista un usuario con la dirección de correo dada, o el nombre de usuario dado, el portal web D20 envía un mensaje al usuario avisando de que ya está en uso el correo electrónico, el nombre de usuario o ambos dependiendo el caso.

### Validar correo electrónico

Dada una dirección de correo electrónico y un Token de validación del correo electrónico, el portal web D20 reenvía los datos recibidos al servicio web de cuentas de usuario a la función correspondiente de validación de correo electrónico. Si éste devuelve true, el portal web D20 envía un aviso al usuario de que la cuenta ya está registrada y ya puede conectarse, en caso contrario avisa de que el Token o la dirección de correo electrónica proporcionada no es válida.

### Petición de recuperación de cuenta

El usuario introduce su correo electrónico, y el portal web D20 lo comprueba haciendo uso del servicio web de cuentas de usuario. Si es correcto, el servicio web de cuentas de usuario devuelve un Token de recuperación de cuenta. Tras recibir el Token de recuperación de cuenta, el portal web D20 envía un correo electrónico al usuario con el Token y las instrucciones para recuperar la cuenta, y a su vez, avisa al usuario de que se le han mandado las instrucciones para recuperar la cuenta a su correo electrónico.

En caso de que la dirección de correo electrónico no sea válida, el portal web D20 avisa al usuario de que el correo electrónico no es válido.

### Recuperar cuenta

El usuario introduce el Token de recuperación de cuenta y su dirección de correo electrónico. El portal web D20 las compruebas haciendo uso del servicio web de cuentas de usuario. Si son válidos, el portal web D20 permite al usuario modificar la contraseña para dicha cuenta.

En caso de que no sean correctas, el portal web D20 muestra un mensaje avisando de que los datos introducidos no son correctos.

### Visualizar juegos

El portal web D20 muestra el juego junto con los datos referentes al juego seleccionado por el usuario.

### Cambiar idioma

El usuario selecciona un idioma distinto al que tiene seleccionado, el portal web D20 cambia el idioma del usuario en el portal Web D20.

### Cambiar datos del perfil

El usuario modifica los datos de su perfil y envía los datos al portal web D20. El portal Web D20 comprueba que los datos introducidos son validos, en tal caso los envía al servicio web de cuentas de usuario Si el servicio web de cuentas de usuario no devuelve ninguna excepción, se avisa al usuario de que los datos de usuario han sido modificados satisfactoriamente. En caso contrario avisa de que alguno de los datos es incorrecto.

### Cuentas asociadas

El usuario selecciona uno de los proveedores de cuentas externas y se le redirecciona a éste para que se conecte en el. Tras ello el portal web D20 con los datos recibidos, los reenvía al servicio web de cuentas de usuario. Si los datos son correctos, se cambia el estado del usuario a usuario autentificado, en caso contrario se avisa al usuario de que ha ocurrido un error.

### Eliminar cuentas asociadas

El usuario selecciona una de las cuentas que tiene asociadas a su cuenta de usuario. Tras ello el portal web D20 manda la petición al servicio web de cuentas de usuario. Si éste no devuelve ninguna excepción, el portal web D20 elimina la cuenta asociada y se avisa al usuario de que la cuenta asociada ha sido eliminada. En caso de error se avisa al usuario del error ocurrido.

### Jugar a juegos contenidos en el portal web D20

El usuario accede a la sección de jugar del juego seleccionado. El portal web D20 comprueba que el usuario está autentificado, en caso de que lo esté, muestra al usuario el juego seleccionado y le permite jugar. En caso contrario redirecciona al usuario a la ventana de autentificación del portal web D20.

### Desconectarse

El usuario pulsa en el botón de desconectarse, el portal web D20 cambia el estado del usuario de conectado a desconectado.

## 11.3 Implementación

### 11.3.1 Cambios en ASP.NET Identity del portal web D20

Para conseguir que el portal web interactúe con los servicios web, hay que realizar diversas modificaciones es ASP.NET Identity, ya que por defecto viene configurado para hacer uso de un gestor de base de datos de Microsoft SQL Server.

Para ello se han creado tres controladores, CustomSignInManager y CustomUserManager que sobrescriben a las clases abstractas de ASP.NET Identity SignInManager y UserManager.

- CustomsignInManager: es el encargado de comprobar las Credenciales de los usuarios.
- CustomUserManager es el encargado de obtener y modificar los datos de los usuarios, y de crear y borrar los usuarios.

La implementación de los controladores se muestra más adelante.

También ha sido necesario modificar las clases que realizan operaciones con los datos de obtención y modificación de usuarios. Por ello se han creado la clase CustomUserStore.

Una vez creadas las clases anteriormente mencionadas se ha configurado la clase Startup.cs para sobrescribir las clases propias de ASP.NET Identity. Para ello, en la función ConfigureServices se ha añadido lo siguiente:

```
...
public void ConfigureServices(IServiceCollection services) {
...
 services.AddIdentity<ApplicationUser, ApplicationRole>()
  .AddEntityFrameworkStores<ApplicationDbContext>()
  .AddDefaultTokenProviders()
  .AddUserStore<CustomUserStore>()
  .AddUserManager<CustomUserManager>();
 services.AddScoped<SignInManager<ApplicationUser>, CustomSignInManager>();
...
```

#### 11.3.1.1 CustomSignInManager

Se ha creado una clase que hace uso de la clase abstracta SignInManager la cual provee ASP.Net Identity. Esta nueva clase con las modificaciones realizadas, hace uso del servicio de cuentas de usuarios, para ello ha sido necesario sobrescribir los métodos encargados de comprobar las Credenciales de los usuarios, tanto con cuentas locales como externas.

A continuación se muestra un ejemplo de cómo se ha sobrescrito el método encargado de la comprobación de las credenciales locales haciendo uso del servicio de cuentas de usuario.

```
public class CustomSignInManager : SignInManager<ApplicationUser> {
 AccountService.AccountClient accounts = new
AccountService.AccountClient(AccountService.
```

```
AccountClient.EndpointConfiguration.Account);
...
public override Task<SignInResult> PasswordSignInAsync(string userName, string
password, bool isPersistent, bool lockoutOnFailure) {
 try {
  Task<AccountService.User> taskA = accounts.checkUserPasswordAsync(userName,
password);
  AccountService.User serviceUser = taskA.Result;
  if (serviceUser == null) {
   return Task.FromResult<SignInResult>(SignInResult.Failed);
  }
  if (serviceUser.IsLocked) {
   return Task.FromResult<SignInResult>(SignInResult.LockedOut);
  }
  if (!serviceUser.EmailConfirmed) {
   return Task.FromResult<SignInResult>(SignInResult.NotAllowed);
  }
  ApplicationUser user = new ApplicationUser(serviceUser);
  base.SignInAsync(user, isPersistent);
  return Task.FromResult<SignInResult>(SignInResult.Success);
 }catch (Exception e){
  return Task.FromResult<SignInResult>(SignInResult.Failed);
 }
}
...
```

#### 11.3.1.2 CustomUserManager

Como ya se ha mencionado previamente la clase CustomUserManager se encarga de obtener y modificar los datos de los usuarios, así como de la creación y eliminación de estos.

Para interactuar con el servicio web de cuentas de usuario, ha sido necesario sobrescribir los métodos encargados de crear y eliminar cuentas de usuario locales, añadir o eliminar cuentas de usuario externas así como las funciones encargadas de obtener y modificar los datos de los usuarios.

A continuación se muestra un ejemplo de la función sobrescrita encargada de crear los usuarios.

```
public class CustomUserManager : UserManager<ApplicationUser> {
AccountService.AccountClient accounts = new
AccountService.AccountClient(AccountService.
AccountClient.EndpointConfiguration.Account);
                                        ...
public override async Task<IdentityResult> CreateAsync(ApplicationUser user,
string password) {
 try {
  await accounts.createUserAndAuthorizationAsync(user.UserName, password,
user.Email, true);
  return IdentityResult.Success;
 } catch (Exception e) {

 }
 IdentityError errorResult = new IdentityError() { Code = "Error", Description =
"Unknown" };
  return IdentityResult.Failed(errorResult);
 }
                                        ...
```

#### 11.3.1.3 CustomUserStore

La clase CustomUserStore es la encargada de obtener y guardar datos de usuarios.

Dado que muchas de las funcionalidades de las que hace uso, no se utilizan ya que son delegadas en el servicio web de cuentas de usuario como podría ser el cifrado de las contraseñas etc., solo se han vuelto a implementar algunas de las funciones.

Las funciones que se han vuelto a implementar son las encargadas de obtener el usuario y comprobar los roles de éste.

A continuación se muestra un ejemplo de cómo se obtienen los usuarios en la clase CustomUserStore haciendo uso del servicio web de cuentas de usuario.

```
public class VGUserStore : IUserStore<ApplicationUser> ,
IUserPasswordStore<ApplicationUser> , IUserLoginStore <ApplicationUser>,
IUserRoleStore<ApplicationUser>{
AccountService.AccountClient accounts = new
AccountService.AccountClient(AccountService.
AccountClient.EndpointConfiguration.Account);
...
 public Task<ApplicationUser> FindByIdAsync(string userID, CancellationToken
cancellationToken) {
 try {
  uint realUserID = (uint)Convert.ToInt32(userID);
  Task<AccountService.User> userTask = accounts.getUserByIDAsync(realUserID);
  AccountService.User serviceUser = userTask.Result;
  ApplicationUser user = new ApplicationUser(serviceUser);

  return Task.FromResult<ApplicationUser>(user);
 }catch (Exception e) {
  return null;
 }
}
...
```

### 11.3.2 Login Oauth con Google, Facebook y Twitter

Uno de los requisitos para el portal D20 es el de permitir la conexión al portal web D20 mediante cuentas externas como son las de Google, Facebook y Twitter. Esto se consigue mediante las librerías correspondientes a dichos proveedores de cuentas previamente instaladas.

Para implementar en el portal web D20 el acceso con cuentas externas, lo primero que se ha hecho es instalar las librerías de cada uno de los proveedores de cuentas tal y como se ha mencionado en el apartado de Librerías y herramientas utilizadas.

También es necesario registrarse, tanto en Google, Facebook como Twitter y entrar en las secciones de desarrolladores para registrar el portal web D20 y obtener las credenciales de los proveedores de cuentas externas para el portal web D20.

Una vez instaladas las librerías pertinentes y creadas las cuentas, hay que decirle al portal web que haga uso de dichos proveedores de cuentas. Esto se hace editando la clase Startup.cs y añadiendo en la función Configure los datos de los proveedores de cuentas.

Se han omitido las credenciales por asteriscos.

```
                                        ...
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)

//Google AUTH
  string googleClientID = "*********.apps.googleusercontent.com";
  string googleClientSecret = "*********

  app.UseGoogleAuthentication(options =>
   {
    options.ClientId = googleClientID;
    options.ClientSecret = googleClientSecret;
   });

  //Facebook Auth
  string facebookID = "*********";
  string facebookSecret = "*********";

   app.UseFacebookAuthentication(options => {
    options.ClientId = facebookID;
    options.ClientSecret = facebookSecret;
   });

  //Twitter Auth
  string TwitterConsumerKey = "*********";
  string TwitterConsumerSecret = "*********";

  app.UseTwitterAuthentication(options => {
   options.ConsumerKey = TwitterConsumerKey;
   options.ConsumerSecret = TwitterConsumerSecret;
  });
```

Una vez realizados los cambios, a la hora de acceder a la ventana de login en el portal web D20, aparece la opción de los tres proveedores de cuentas de usuario Google Facebook y Twitter.

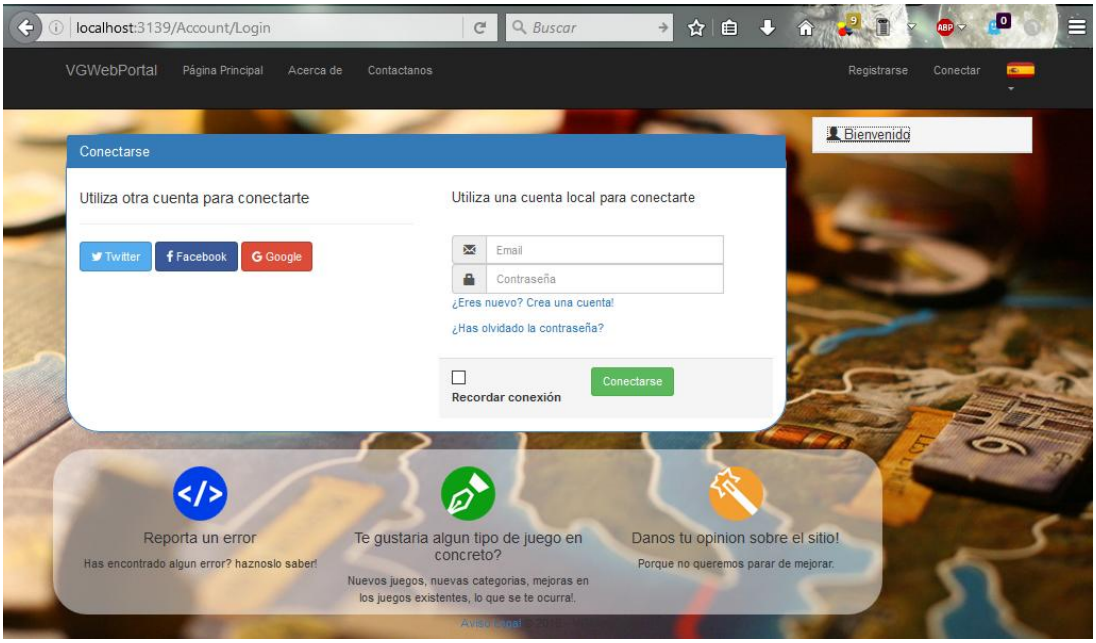


Figura 27: Ventana login portal web D20 con proveedores de cuentas externas

Pese a los cambios realizados, para que interactúe con el servicio web de cuentas de usuario es necesaria unas modificaciones en el controlador de UserAccount en la acción de ExternalLoginCallback, tal y como se muestra a continuación.

```
public class AccountController : Controller{
// GET: /Account/ExternalLoginCallback
 [HttpGet]
 [AllowAnonymous]
 public async Task<IActionResult> ExternalLoginCallback(string returnUrl = null){
  ExternalLoginInfo info = await
_CustomSignInManager.GetExternalLoginInfoAsync();
   if (info == null){
    return RedirectToAction(nameof(Login));
   }

   SignInResult result = _ CustomSignInManager.externalLogin(info, isPersistent:
false);

   if (result.Succeeded){
...
```

Una vez que el proveedor de la cuenta externa devuelve los datos del usuario, estos son recuperados mediante la función de GetExternalLoginInfoAsync de _CustomSignInManager.

Posteriormente se ha implementado la función de externaLoginInfo para comprobar los datos obtenidos y almacenarlos en el servicio web de cuentas de usuario.

### 11.3.3 Soporte múltiples idiomas

El soporte de idiomas se incluye entre las librerías de ASP.NET en la librería de Localization, solo falta importarla donde se utilice y configurarla en la web para su uso.

Se ha decido cambiar la forma en la que obtiene los idiomas y de donde los obtiene, para facilitar cambios futuros. Para ello se ha implementado dos clases, las cuales son usadas por la librería de Localization. LanguageStringLocalizer que implementa la interfaz de IStringLocalizer y LanguageStringLocalizerFactory que implementa la interfaz de IStringLocalizerFactory.

Para habilitarlo, hay que editar la clase Startup.cs y añadir en la función ConfigureServices lo siguiente

```
                                        ...
public void ConfigureServices(IServiceCollection services) {
                                        ...
 services.AddLocalization(options => options.ResourcesPath = "Resources");
services.AddMvc()
 .AddViewLocalization()
.AddDataAnnotationsLocalization();services.AddSingleton<IStringLocalizerFactor
y, LanguageStringLocalizerFactory>();
                                        ...
```

Esto habilita la utilización de diferentes lenguajes en las vistas de la página web. ResourcesPath es la localización física en la página web, en donde se van a encontrar los ficheros que contentan los diferentes idiomas.

También es necesario especificar que idiomas estarán disponibles en el portal web, así como el idioma por defecto. En la misma clase Startup.cs, en la función configure, hay que añadir lo siguiente:

```
                                        ...
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)...
RequestLocalizationOptions localizationOptions = new
RequestLocalizationOptions {
 SupportedCultures = new List<CultureInfo> { new CultureInfo("en-US"), new
CultureInfo("es-ES") },
 SupportedUICultures = new List<CultureInfo> { new CultureInfo("en-US"), new
CultureInfo("es-ES") }
 };
 app.UseRequestLocalization(localizationOptions, new RequestCulture("es-
ES"));...
```

Dentro de la raíz de la web, tal y como se ha configurado el resources Path, en la carpeta resources hay que añadir los ficheros .resx con los diferentes idiomas para cada vista que se quiere tener en diferentes idiomas. Los ficheros creados tienen que seguir el formato de nombre de View.NombreControlador.NombreAccion.idioma.resx

También es necesario importar las librerías en la vista de Microsoft.AspNet.Mvc.Localization y añadir IViewLocalizer. Localization utilizar la etiqueta *Localization* en la vista para

especificar el texto que se va a utilizar. A continuación se muestra un ejemplo para el apartado de login.

Por un lado habrá dos ficheros View.Account.Login.cshtml.en-US.resx y View.Account.Login.cshtml.es-ES.resx, en los cuales estará los textos en castellano y en ingles. Por otro se modifica la vista de login como se muestra a continuación:

```
                                        ...
@using Microsoft.AspNet.Mvc.Localization
@inject IViewLocalizer Localization
                                        ...
<section>
 <form asp-controller="Account" asp-action="Login" asp-route-
returnurl="@ViewData["ReturnUrl"]" method="post" class="form-horizontal"
role="form">
  <div class="panel-heading">
   <h3 class="panel-title">
    @Localization["UseLocalAccount"]
   </h3>
  </div>
  <div class="panel-body">
   <div asp-validation-summary="ValidationSummary.All" class="text-danger bg-
danger"></div>
    <div class="input-group">
     <span class="input-group-addon"><span class="glyphicon glyphicon-
envelope"></span></span>
     <input asp-for="Email" class="form-control" placeholder="Email" required
autofocus />
     <span asp-validation-for="Email" class="text-danger bg-danger"></span>
    </div>
    <div class="input-group">
     <span class="input-group-addon"><span class="glyphicon glyphicon-
lock"></span></span>
     <input asp-for="Password" class="form-control"
placeholder="@Localization["Password"]" required autofocus />
     <span asp-validation-for="Password" class="text-danger bg-danger"></span>
    </div>
    <p>
     <a asp-action="Register">@Localization["RegisterNewUSer"]</a>
    </p>
    <p>
     <a asp-action="ForgotPassword">@Localization["ForgotPassword"]</a>
    </p>
   </div>
   <div class="panel-footer">
    <div class="row">
     <div class="col-xs-4 col-sm-4 col-md-5">
      <input asp-for="RememberMe" />
      <label asp-for="RememberMe">@Localization["RememberMe"]</label>
     </div>
     <div class="col-xs-6 col-sm-6 col-md-7">
      <button type="submit" class="btn btn-labeled btn-
success">@Localization["Connect"]</button>
     </div>
    </div>
   </div>
  </form>
 </section>
                                        ...
```

Tras los cambios realizados, al acceder a la vista, dependiendo el idioma que tenga seleccionado el usuario en el portal web D20, los textos que son obtenidos mediante la etiqueta @Localization, serán en ingles o castellano.

### 11.3.4 Envío de correos electrónicos

Los correos electrónicos que envía el portal web, son a través de Gmail. Para poder enviar dichos corres, se ha creado una cuenta de correo en Gmail para el envío de correos electrónicos

Para implementar el envío de correos electrónicos se hace uso de la clase provista en ASP.NET MVC6 AuthMessageSender y se ha implementado la función de SendEmailAsync.

Para el envío de correos electrónicos a través de Gmail, se hace uso de la librería de .Net Mail, la aplicación se conecta mediante SMTP a Gmail para el envío de correos electrónicos.

Por un lado se crea una instancia del objeto MailMessage con el contenido del mensaje y los datos del receptor, y por otro lado se hace uso de SmtpClient para el envío de dicho mensaje a través del SMTP de Gmail.

## 11.4 Integración K-Tan en el portal D20

Al jugar a K-Tan desde el portal D20, para evitar que el usuario tenga que identificarse tanto en el portal como en K-Tan, se establece un protocolo para que el juego pueda obtener los datos necesarios para identificar al usuario sin necesidad de que éste proporcione ningún tipo de credencial.

Una vez se carga y se inicia el juego en el navegador, el propio juego utiliza una función de javascript que hace una petición AJAX al portal D20. Éste, que ya ha tenido que identificar al usuario previamente y tiene los permisos necesarios, invoca el servicio de cuentas de usuario para solicitar un Token. El portal D20 devuelve el Token solicitado junto con la id del usuario. Mediante estos datos, el juego es capaz de identificar al usuario y obtener su información a través del servicio de cuentas.

Este Token, es de un solo uso y además de tiempo limitado, tras el cual se caduca y queda invalidado.

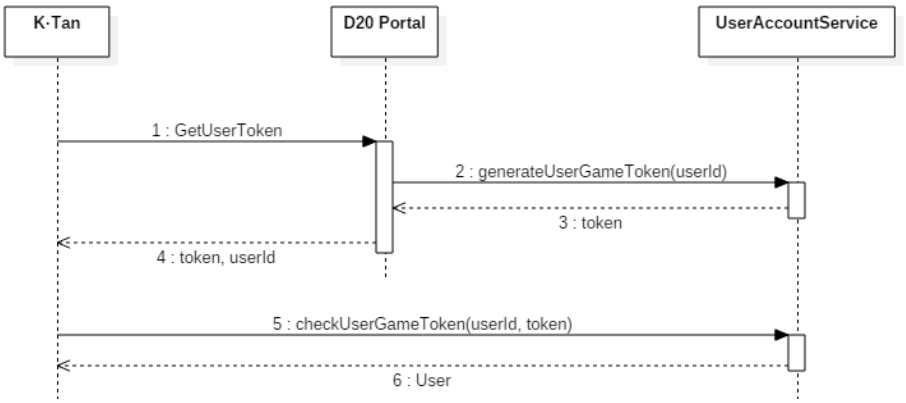


Figura 28: Diagrama de secuencia de la identificación de usuarios mediante Token

## 11.5 Pruebas con usuarios

En la última etapa del ciclo de este proyecto se han desarrollado unas pruebas con usuarios para validar la usabilidad de las aplicaciones desarrolladas para IHEGames y verificar la satisfacción de los usuarios.

### 11.5.1 Tipos de usuarios

IHEGames establece una serie de usuarios tipo para la realización de las pruebas con usuarios, en base al espectro de dedicación de dicho usuarios en los juegos. Se distinguen las siguientes cuatro categorías de usuarios.

Casual: Se denomina así a los jugadores que no suelen emplear mucho tiempo en un juego dado, caracterizándose, en muchos casos, por jugar a muchos juegos, pero durante poco tiempo o en intervalos irregulares. Además, estos jugadores no suelen estar comprometidos propiamente a conseguir todos los objetivos posibles en un juego. Un gran grupo de ellos no tienen interés en mejorar sus habilidades ya que lo consideran solo un pasatiempo.

Regular: Muchas personas no consiguen encajar en la categoría de jugador *Hardcore*, pero a su vez tampoco en jugadores casuales, para estos se asigna el término de jugadores regulares el cual se refiere a un jugador que juega de manera habitual, tiene ciertos conocimientos de los videojuegos, pero no busca un gran reto como los jugadores *Hardcore*; son competitivos pero no tienen interés en ser los mejores.

Hardcore: Se caracteriza por ser un jugador que dedica muchas horas al día a jugar videojuegos y que busca mejorar constantemente y tener puntuaciones máximas. Estos jugadores no quedan satisfechos con terminar un videojuego de manera habitual, y buscan siempre conseguir todo lo que se pueda en la mayor dificultad posible. Estos jugadores son verdaderamente aficionados a los videojuegos, no buscan solo un medio de entretenimiento sino un reto o una aventura.

Profesional: Se caracteriza por ser un jugador con habilidades extraordinarias para jugar, y por ello, es considerado un jugador experto por la comunidad de los videojuegos. Un gran grupo de estos jugadores logran ganar dinero por sus habilidades, integrarse a un equipo y participar en torneos.

Para las pruebas realizadas, se han buscado jugadores de las 3 primeras categorías, ya que IHEGames tiene estimado que con estos tres tipos de usuario se cubre más del 90% de los consumidores de videojuegos.

### 11.5.2 Tipos de dispositivos

Para las pruebas realizadas se han utilizado dispositivos con diferentes tamaños y resoluciones de pantalla para comprobar la usabilidad de las aplicaciones y sus interfaces. Por ello, las pruebas se han llevado a cabo tanto en PCs y portátiles, como en tabletas y móviles.

Tal y como está definido en su alcance, el juego K-Tan se ha desarrollado para su ejecución en dispositivos Android y en la web a través del Portal D20; debido a esto, los dispositivos móviles y tabletas utilizadas en estas pruebas disponen del sistema operativo Android.

A continuación se listan los dispositivos utilizados en las pruebas:

| Núm. | Tipo de dispositivo | Pantalla | |
|---|---|---|---|
| | | Tamaño | Resolución |
| D1 | Móvil Android | 5“ | 480x854 |
| D2 | Móvil Android | 5“ | 1280 x 720 |
| D3 | Tableta Android | 7“ | 1024x600 |
| D4 | Tableta Android | 10.1” | 1280x800 |
| D5 | Portátil | 14” | 1366x768 |
| D6 | Portátil | 15.6” | 1920x1080 |
| D7 | PC | 24” | 1680x1050 |
| D8 | PC | 24” | 1920x1080 |

Tabla 2: Dispositivos utilizados en las pruebas con usuarios

### 11.5.3 Pruebas realizadas

Para estas pruebas se ha reunido a 8 usuarios, distribuidos entre las 3 categorías de usuario descritas en el apartado *Tipos de usuario*. Concretamente han sido 3 jugadores casuales, 3 jugadores regulares y 2 jugadores *hardcore*, entre los cuales 1 de los jugadores casuales, 2 de los regulares y uno de los *hardcore* conocían de antemano las reglas de *Los colonos de Catan* en el cual está basado el juego K-Tan. Dado que las pruebas se han realizado simultáneamente con los 8 usuarios, cada uno ha hecho uso de uno de los 8 dispositivos descritos en el apartado anterior *Tipos de dispositivos*.

A lo largo de estas pruebas cerradas, los usuarios han tenido libertad de explorar las distintas funcionalidades a su gusto, pero han sido guiados a través de una serie de pasos que se han dividido en dos etapas. En la primera etapa, los usuarios han podido probar algunas de las funcionalidades del Portal D20:

- Los usuarios han realizado los pasos necesarios para completar su registro en el Portal D20 de IHEGames. Los registros de las cuentas se han realizado tanto creando una cuenta mediante usuario y contraseña como a través de otro proveedor de cuentas de usuario externo como Google, Facebook o Twitter.
- Los usuarios han podido completar sus perfiles y modificar los datos de sus cuentas.
- Los usuarios han explorado las secciones y funcionalidades del Portal D20.
- Finalmente, los usuarios han rellenado una encuesta sobre su experiencia en el Portal D20.

En la segunda etapa los usuarios han podido probar la aplicación de K-Tan:

- Los usuarios que han accedido a través de la App de Android de K-Tan, se han identificado.
- Los jugadores han explorado las funcionalidades de la interfaz general de K-Tan.
- Se han creado dos partidas simultáneas de 4 jugadores .Los 8 jugadores se han distribuido en estas dos partidas en base a si conocían de antemano las reglas del juego.
- Se han explicado las reglas de juego a los jugadores que no las conocían.
- Los jugadores han explorado las funcionalidades de la interfaz de partida.
- Los jugadores de la primera partida en concluir, se han unido a la otra partida como espectadores.
- Tras la finalización de las dos partidas, los usuarios han rellenado una segunda encuesta sobre su experiencia con K-Tan.

### 11.5.4 Resultados

Las encuestas realizadas a los usuarios se han realizado a mediante la aplicación online de Google Forms. Esta aplicación web permite generar formularios de una forma fácil y sencilla mediante las herramientas de que dispone. Y ofrece la posibilidad de responder a los formularios a través de un enlace. Por otro lado, almacena todas las respuestas enviadas y genera gráficas a partir de dichas respuestas, facilitando la compresión y análisis de los resultados.

A continuación se muestran algunas de las gráficas obtenidas en las encuestas y referentes a las características de los usuarios comentadas a lo largo de este capítulo (se reservan el resto de los resultados obtenidos):

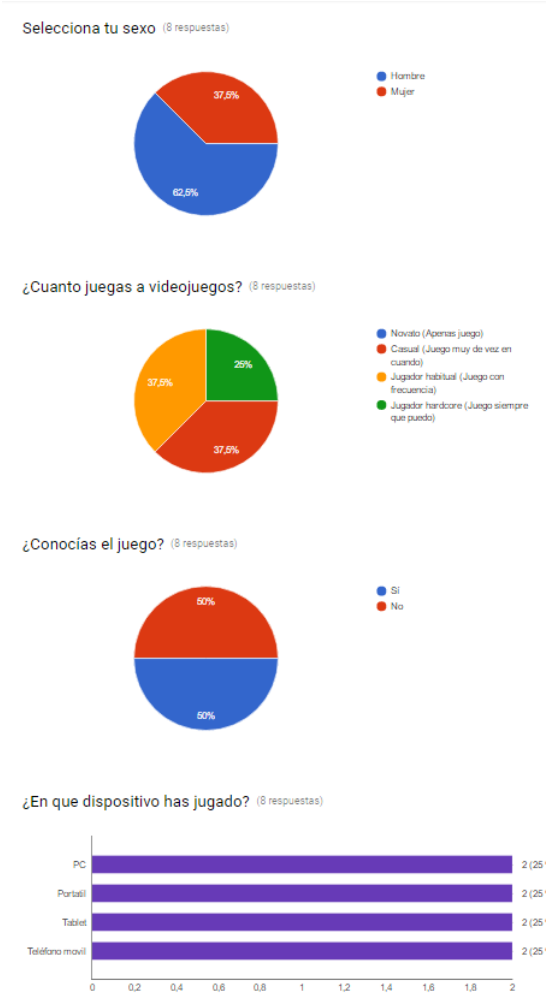


Figura 29: Gráficas de las respuestas de las encuestas

A partir de las valoraciones recogidas en estas encuestas, IHEGames inferirá las nuevas especificaciones para las siguientes versiones del Portal D20 y K-Tan.

# 12

# Gestión

En este capítulo se describen, las áreas más relevantes en lo referente a la gestión llevada cabo a lo largo de todo este proyecto.

Dada las dependencias directas que existen entre el Portal D20 y los servicios de IHEGames, desarrollados en este proyecto, y el juego K-Tan, desarrollado en otro proyecto en paralelo, la gestión del tiempo y de las comunicaciones es fundamental a lo largo de este proyecto.

Además de la gestión de las comunicaciones y del tiempo, en este capítulo, también se desarrolla la gestión de costes y del alcance, puesto que son también de gran relevancia para este proyecto.

## 12.1 Gestión del alcance

Debido a las dependencias que existen entre este proyecto y el proyecto en el que se desarrolla K-Tan, se establece al inicio del proyecto que todo cambio en el alcance del proyecto se debe de realizar con la aprobación de IHEGames y de los responsables de ambos proyectos.

El alcance del proyecto no ha sufrido grandes cambios a lo largo del desarrollo de éste, sin embargo dado que estos cambios no han sido significativos, no ha hecho falta consultarlos con la directiva de IHEGames.

Al hacer uso de tecnologías actuales, las cuales disponen de poca documentación a día de hoy e interconectarlas entre sí, ha conllevado en desvíos significativos en la realización de algunas partes del proyecto, lo que ha implicado que tareas que previamente había planificadas, fueran trasladadas a una fase posterior a la realización de este proyecto.

Tras finalizar el prototipado inicial con unas funcionales mínimas, tanto de los servicios web, como del servidor de chat y la pagina web, y la intercomunicación de estos, se comenzó con el desarrollo. Pero debido a los fallos surgidos en las comunicaciones, entre los servicios web y la página web, se decide priorizar la realización de los servicios por prioridades.

Al inicio de cada proyecto se definen unas especificaciones iniciales para la primera versión del producto a desarrollar. Una vez desarrollada la primera versión se procede a la validación del producto mediante pruebas con usuarios reales. De las pruebas realizadas se recogen las valoraciones de los usuarios y se definen unas nuevas especificaciones para la siguiente versión. Este proceso se continúa hasta obtener una versión final.

## 12.2 Gestión de costes

Una gran cantidad de tiempo es aplicada en entender el funcionamiento de las tecnologías utilizadas, esto se puede apreciar en el tiempo invertido en el primer servicio web y el portal web D20, y como en el resto de servicios el tiempo previsto disminuye considerablemente. El tiempo invertido en el aprendizaje se ha introducido en el diseño y desarrollo del módulo del servicio de cuentas de usuario.

A pesar de ello, se ha conseguido ajustar el tiempo de una forma más precisa de la esperada, ya que la desviación total de tiempo entre la planificación inicial y la final apenas supone un 10%.

En los siguientes apartados se refleja de una forma más concisa el coste de tiempo real en la elaboración del proyecto.

| Bloque de trabajo | Fase | Dedicaciones | | |
|---|---|---|---|---|
| | | Estimadas | Reales | Desvíos |
| Servicio Cuentas de usuarios | Análisis diseño y desarrollo | 97h | 116h | +19h |
| | Pruebas | 20h | 24h | +4h |
| | Puesta en producción | 4h | 2h | -2h |
| | **Subtotales** | **121h** | **142h** | **+21h** |
| Servicio Honor | Análisis diseño y desarrollo | 35h | 27h | -8h |
| | Pruebas | 10h | 10h | 0h |
| | Puesta en producción | 2h | 2h | 0h |
| | **Subtotales** | **47h** | **39h** | **-8h** |
| Servicio ELO | Análisis diseño y desarrollo | 15h | 17h | +2h |
| | Pruebas | 8h | 8h | 0h |
| | Puesta en producción | 2h | 2h | 0h |
| | **Subtotales** | **25h** | **27h** | **+2h** |
| Portal Web | Análisis diseño y desarrollo | 90h | 105h | +15h |
| | Pruebas | 15h | 12h | -3h |
| | Puesta en producción | 10h | 8h | -2h |
| | **Subtotales** | **115h** | **125h** | **+10h** |
| Gestión | Planificación | 8h | 8h | 0 |
| | Seguimiento y control | 4h | 4h[2] | 0[2] |
| | Reuniones | 12h | 14h[2] | +2h[2] |
| | **Subtotales** | **24h** | **26h** | **+2** |
| Trabajo Académico | Memoria | 60h | 80h[2] | +20h[2] |
| | Presentación/Defensa | 10h | 10h[2] | 0h[2] |
| | **Subtotales** | **70h** | **90h** | **+20h** |
| **Totales** | | **402h** | **449 h** | **+47h** |

Tabla 3: Costes de tiempo en el proyecto

Dado que el material necesario para la elaboración del proyecto ha sido proporcionado de forma gratuita, el único coste económico reconocido es el coste en horas trabajadas en éste.

Dado que los certificados utilizados en esta etapa del proyecto no han sido emitidos por ninguna entidad certificadora reconocida sino que han sido creados y autofirmados, su coste es nulo, y no se valora en la gestión de costes.

Al ser un proyecto colaborativo se computan también las horas de consulta, las cuales han sido incluidas en la categoría de gestión en el apartado reuniones.

*2 Valores estimados dado que la tarea no ha terminado*

## 12.3 Gestión de las comunicaciones

Se observa al inicio, tanto en del proyecto de K-Tan como en del de los servicios de IHEGames y el Portal D20, que ambos proyectos comparten los mismos interesados. En la siguiente tabla se detallan todos los interesados identificados:

| INTERESADOS | |
|---|---|
| Responsable del proyecto de K-Tan | Iker Boyra Sarachaga |
| Responsable del proyecto de los servicios IHEGames y del portal D20 | Héctor Antruejo Escalante |
| Promotores de IHEGames | Héctor Antruejo Escalante<br>Iker Boyra Sarachaga |
| Tutor del Proyecto de Fin de Grado | José Miguel Blanco |

Tabla 4: Interesados del proyecto

A continuación se indica la autoridad ("Poder": Bajo/Medio/Alto) y el nivel de preocupación ("Interés": Bajo/Medio/Alto) con respecto a los resultados de este proyecto de cada uno de los interesados identificados para cada uno de los bloques principales de este proyecto.

| | PRODUCTO | | ACADÉMICO | | GESTIÓN | |
|---|---|---|---|---|---|---|
| | INTERÉS | PODER | INTERÉS | PODER | INTERÉS | PODER |
| Responsable del proyecto de K-Tan | Medio | Medio | Bajo | Bajo | Alto | Medio |
| Responsable del proyecto de los servicios IHEGames y del portal D20 | Alto | Alto | Alto | Alto | Alto | Alto |
| Promotores de IHEGames | Alto | Alto | Bajo | Bajo | Alto | Alto |
| Tutor del Proyecto de Fin de Grado | Medio | Bajo | Alto | Alto | Medio | Bajo |

Tabla 5: Interés y poder de los interesados del proyecto

Dada las dependencias que existen entre ambos proyectos, y el interés y el poder que se deriva de dicha dependencia, es importante mantener una buena comunicación entre los responsables de ambos proyectos, e indirectamente con los promotores de IHEGames.

Para mantener una buena comunicación y visibilidad de los proyectos, se definen dos líneas de comunicación principales.

Mediante la herramienta online de Mingle, ambos responsables registran diariamente el progreso y el estado de sus respectivos proyectos. De esta manera, cada responsable es capaz de visionar en cualquier momento el avance y características de las fases, iteraciones o historias ("Story") de cualquiera de los dos proyectos, entre las que se destaca:

- Estado completado/en curso/pendiente
- Fecha de inicio y de finalización
- Estimación de horas inicial
- Estimación de horas hasta su final
- Coste de horas dedicadas

Por otro lado, se establecen reuniones periódicas entre ambos responsables, tanto presenciales como vía Skype, para compartir los aspectos relevantes de ambos proyectos entre ambos responsables e indirectamente a los promotores de IHEGames.

Adicionalmente, se establece un sistema de información compartido mediante Google Drive para poder compartir la documentación relevante entre ambos responsables de proyecto.

Respecto a la comunicación con el tutor del Proyecto de Fin de Grado, ésta se realiza mediante reuniones acordadas previamente mediante mensajes de correo electrónico. Gestión de adquisiciones

## 12.4 Gestión del tiempo

El tiempo a lo largo de este proyecto se ha gestionado principalmente en base al ciclo de vida del proyecto, teniendo en cuenta el calendario académico y las fechas en las que se han previsto causas externas que limiten el tiempo de dedicación disponible para la realización del proyecto.

En la siguiente figura se muestra la distribución preestablecida de los distintos paquetes de trabajo a lo largo del tiempo así como las transiciones entre las fases del ciclo de vida del proyecto.

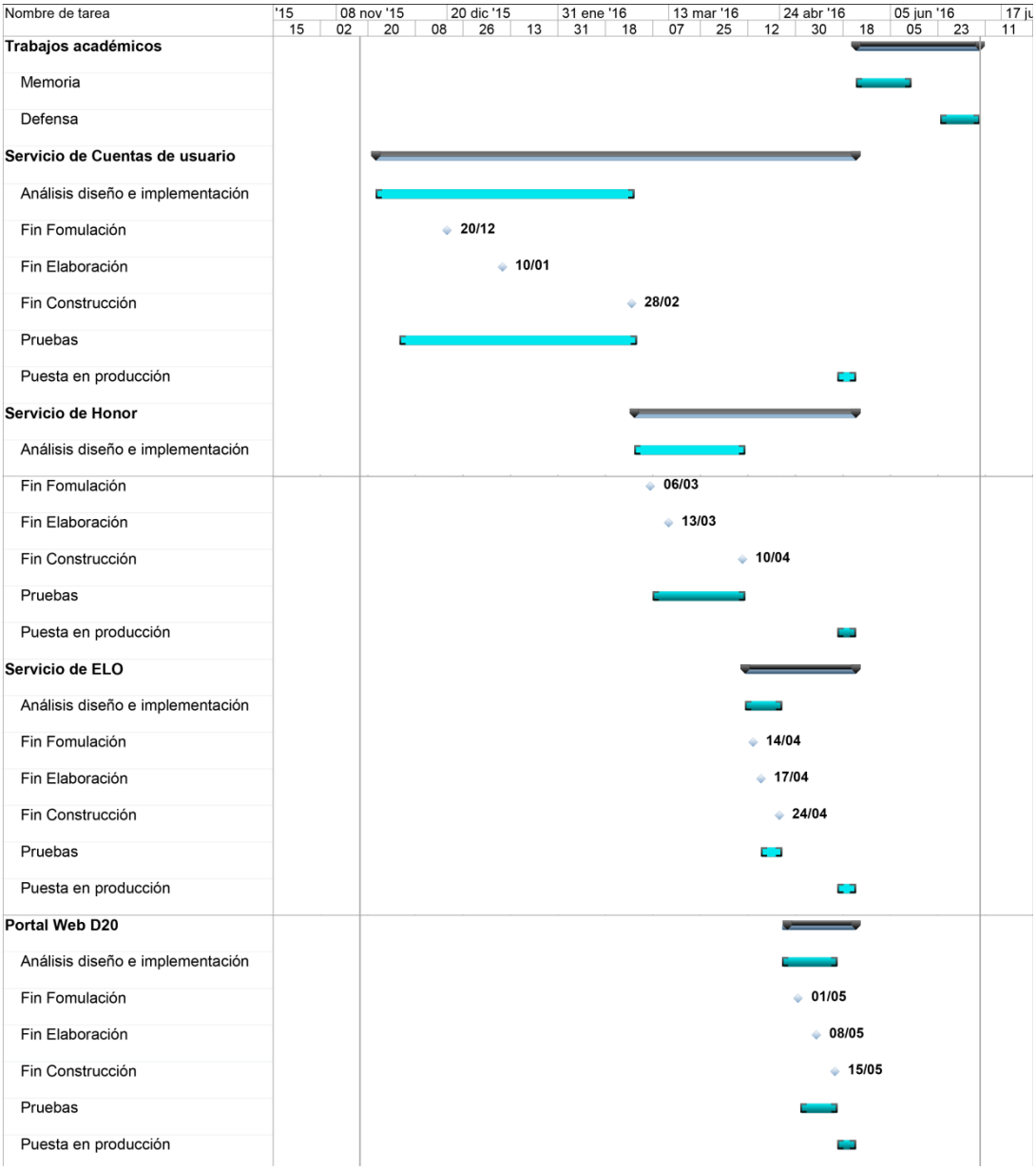


Figura 30: Diagrama de barras del proyecto

Dado que el juego K-Tan depende directamente, tanto de los servicios de IHEGames como del propio portal D20, ha sido necesario gestionar el tiempo conjuntamente con el proyecto paralelo responsable de este juego.

En la siguiente figura, se muestra la distribución de los distintos paquetes de trabajo pertenecientes al proyecto paralelo en el que se desarrolla el juego de K-Tan.

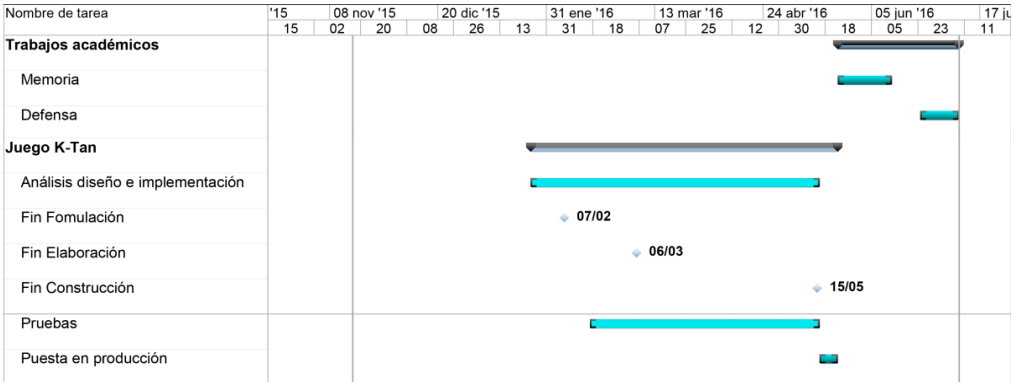


Figura 31: Diagrama de barras del proyecto de K-Tan

Junto a la distribución de las actividades, se establecen los hitos más significativos de ambos proyectos. Estos hitos se muestran en la siguiente figura.

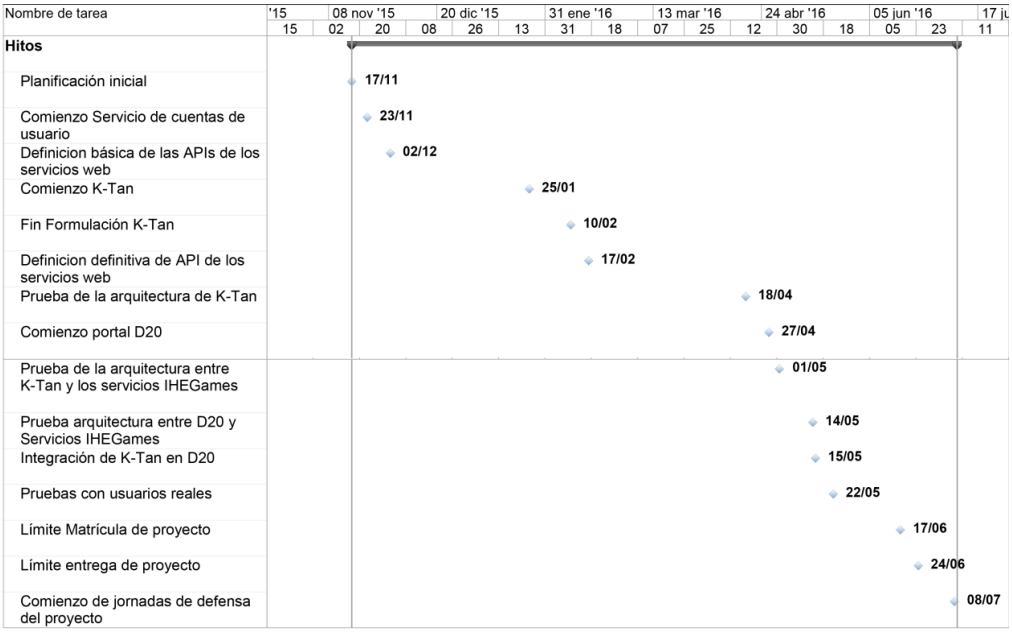


Figura 32: Hitos del proyecto

Tanto la distribución de las actividades como los hitos, se preestablecen teniendo en cuenta las dependencias que existen entre los sistemas desarrollados en cada proyecto y los costes de tiempo estimados de cada paquete de trabajo. A pesar de ello se establecen las fechas finales con márgenes de tiempo suficientes para intentar garantizar la finalización de las actividades y evitar retrasar las actividades dependientes en caso de sobrecostes o eventos inesperados.

Como se muestra en el apartado de Gestión de Costes, los paquetes de trabajo de este proyecto han sufrido, en general, un sobrecoste con respecto a las estimaciones realizadas. A pesar de ello, los márgenes de las fechas han permitido respetar los hitos del proyecto.

# 13

# Conclusiones

Este capítulo contiene las lecciones aprendidas o conclusiones obtenidas a lo largo del desarrollo de este proyecto. Se recogen las conclusiones a nivel de organización o trabajo en grupo, las conclusiones propias obtenidas y finalmente las líneas futuras del proyecto.

## 13.1 Conclusiones organizativas

Creo que como las personas, la gestión no es perfecta, pero que con esfuerzo y trabajo se puede mejorar. Esto se ha visto a lo largo del desarrollo del proyecto. Antes de iniciar este proyecto comenzamos gestionando pequeños proyectos de una forma, y poco a poco hemos ido refinándolo hasta la forma de hacerlo en este proyecto, la cual se describe en el ciclo de vida. Pese a ello se han visto algunos problemas los cuales serán mejorados para posteriores ciclos. Con esto quiero decir, que no hay que acomodarse en lo que ha gestión corresponde, y que siempre se puede encontrar algo que haga de ésta una gestión más eficaz.

Puede parecer a simple vista que la elaboración del proyecto con partes conjuntas puede ahorrar tiempo en la elaboración de éste. Esta impresión dista mucho de la realidad debido a las discrepancias que pueden surgir a lo largo del desarrollo del proyecto, así como de tener que estar de acuerdo en cada cambio que se realiza, debido a que estos cambios pueden afectar a ambos proyectos. Todos estos inconvenientes pueden suponer un coste en horas mucho mayor del que se puede pensar inicialmente.

Pero no todo son desventajas, ya que el desarrollo del proyecto con partes conjuntas ofrece la oportunidad de consultar información técnica sobre temas relevantes al desarrollo del proyecto, pudiendo así despejar dudas en el diseño o en el desarrollo. También permite compartir conocimiento o experiencias adquiridas durante el proyecto que pueden ayudar a lo largo del proyecto o en futuros proyectos.

## 13.2 Conclusiones personales

El mundo de los servicios web parece sencillo a simple vista, pero entrama una cantidad de protocolos y funcionalidades dignas de varios proyectos de fin de carrera. Solo en el ámbito de la seguridad junto con las comunicaciones pueden llegar a ser muy complejos. Pese a ello plataformas como las que ofrece Amazon y Azure para el desarrollo de servicios web pueden facilitar o ser un buen punto de partida para aquellos que se inicien en el mundo de los servicios web a nivel profesional.

También he de decir respecto a los servicios web, que aunque no haya quedado reflejado en el proyecto realizado, he realizado pruebas en el servidor web de Microsoft IIS desplegando los servicios web sin ningún tipo de dificultad. Con esto quiero decir, que junto con lo anteriormente mencionado, si uno decide realizar los servicios web por su cuenta sin hacer uso de las plataformas que proporciona Amazon o Azure, el desarrollo de los servicios web puede ser complejo y confuso al inicio debido a la cantidad de funcionalidades que pueden llegar a ofrecer, pero su despliegue es relativamente sencillo.

Sobre el desarrollo de la página web, he de decir que me ha costado más de lo que esperaba debido a ASP.NET MVC6 es relativamente nuevo y la documentación de Microsoft puede ser un poco confusa, pero el mayor inconveniente fue encontrarme con la sorpresa de que no ofreciese la oportunidad de hacer uso de los servicios WCF. Sorpresa debido a que versiones anteriores como MVC5 si que ofrecía dicha opción, de no haber sido posible la integración mediante la herramienta *WCF Connected Service*, se habría replanteado el uso de ASP.NET MVC6 y se habría hecho uso de la versión anterior, lo que habría supuesto un pequeño contratiempo en el desarrollo del proyecto. Con esto quiero decir, que no se puede dar nada por sentado, ni en temas que pueden parecer obvios, ya que en este caso, a pesar de estar haciendo uso de tecnologías de la misma empresa, las cuales no están obsoletas y que en versiones anteriores sí que estaban soportadas entre sí, se puede dar el caso de que en las nuevas no. Con lo que antes de lanzarse a utilizar algunas tecnologías, sobre todo en tecnologías que tienen que hacer uso unas de otras, hay que realizar pequeñas pruebas entre ellas, antes de embarcarse en su desarrollo.

Finalmente puedo decir que estoy muy contento con el trabajo realizado pese a algunos contratiempos que han ido surgiendo a lo largo del desarrollo del proyecto, ya que el conocimiento y experiencia que he adquirido en el proceso no tiene precio.

## 13.3 Líneas futuras

A lo largo del desarrollo del proyecto, han ido surgiendo nuevas ideas, las cuales serán implementadas en líneas futuras, pero a corto plazo se realizarán tareas que a la hora de realizar el proyecto quedaron pospuestas, dado que excedían con creces el tiempo de desarrollo de éste. Una de las tareas pospuestas corresponde al rendimiento de los servicios web y a conseguir una disponibilidad en todo momento de éstos.

Para comprobar el rendimiento de los servicios web, se realizarán pruebas de estrés, lanzando miles de peticiones simultaneas para ver el rendimiento de éstos y analizar cómo se comportan. Tras los resultados obtenidos se buscará como mejorar el rendimiento de los servicios web, así como comprobar cuántos clientes puede mantener sin que el servicio de indicios de fallos o problemas de rendimiento. Dependiendo los resultados obtenidos, se modificarán los servicios para mejorarlos en lo referente al rendimiento o se pasará a una nueva etapa.

Para mantener la disponibilidad de los servicios web en todo momento, es necesario que estos sean escalables y es necesario crear servicios web de apoyo, para que en caso de que uno de ellos falle o se sature, se pueda desviar el tráfico. Esto implica crear métodos para monitorizar el estado de los servicios y detectar cuando se están saturando, servicios que sean capaces de desviar el tráfico creando así un balance de carga entre servicios principales y secundarios.

Y finalmente en lo referente al portal web D20 queda mucho trabajo por realizar. Pero una de las tareas primordiales consiste en el desarrollo de los diseños de las interfaces. Estas deben cumplir la iniciativa de Mobile Friendly[7] así como todos los requerimientos y estándares de accesibilidad establecidos por Web Accessibility Iniciative[10] para conseguir así que la web obtenga una puntuación en Google lo más elevada posible.

Y tras la realización de todas las tareas previamente descritas, para finalizar, faltaría desplegar todos los servicios web y el portal web D20 en un alojamiento web. Registrar los servicios en la agencia española de protección de datos[11] y añadir los avisos legales pertinentes.

# Bibliografía


Electronic Frontier Foundation. [Online] 26 de Mayo de 2016.
https://www.eff.org/

*WCF Bindings*. [Online] 24 de Mayo de 2016.
https://msdn.microsoft.com/en-us/library/ms730879%28v=vs.110%29.aspx

OASIS, [Online] 24 de Mayo de 2016.
https://www.oasis-open.org/

W3C, [Online] 24 de Mayo de 2016.
http://www.w3c.es/

Documentación WCF Microsoft [Online] 24 de Mayo de 2016
https://msdn.microsoft.com/es-es/library/dd456779(v=vs.110).aspx

ASP.NET MVC [Online] 24 de Mayo de 2016
http://www.asp.net/mvc

ASP.Net Core Identity [Online] 24 de Mayo de 2016
https://github.com/aspnet/Identity/

Mingle [Online] 24 de Mayo de 2016
https://www.thoughtworks.com/mingle/

MVC Recommended Resources [Online] 24 de Mayo de 2016
http://www.asp.net/mvc/overview/performance/recommended-resources-for-mvc

# Referencias


[1] Desarrollo para Comunidad Web de Videojuegos: Videojuego 3D Online Multijugador Multiplataforma. *Iker Boyra Sarachaga*. Junio 2016.
(Vid. Pág.2)

[2] Asociación Española de Videojuegos. [Online] 19 de septiembre de 2015.
http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/
(Vid. Pág. 6)

[3] Noticia el mundo juegos de mesa. [Online] 26 de Mayo de 2016.
http://www.elmundo.es/economia/2015/06/19/5583de0946163fc21e8b4572.html
(Vid. Pág. 6)

[4] Sistema de puntuación ELO. Wikipedia. [Online] 29 de abril de 2016.
https://es.wikipedia.org/wiki/Sistema_de_puntuaci%C3%B3n_Elo.
(Vid. Pág.8, 67)

[5] Https Everywhere. [Online] 26 de Mayo de 2016.
https://www.eff.org/HTTPS-EVERYWHERE
(Vid. Pág. 14)

[6] Google Security Blog. [Online] 19 de septiembre de 2015.
https://security.googleblog.com/2015/12/indexing-https-pages-by-default.html
(Vid. Pág. 14)

[7] Iniciativa mobile friendly. [Online] 26 de Mayo de 2016.
https://support.google.com/adsense/answer/6196932?hl=en
(Vid. Pág. 14, 104)

[8] DB Engines Ranking. [Online] 26 de Mayo de 2016.
http://db-engines.com/en/ranking
(Vid. Pág.15)

[9] Usando Oauth con Google [Online] 23 de mayo de 2016.
https://developers.google.com/identity/protocols/OAuth2
(Vid. Pág. 16

[10] Web Accessibility Iniciative (WAI). [Online] 23 de mayo de 2016.
https://www.w3.org/WAI/
(Vid. Pág. 21, 104)

[11] Agencia Española de Protección de datos. [Online] 23 de mayo de 2016. https://www.agpd.es/portalwebAGPD/index-ides-idphp.php (Vid. Pág.21, 104)

[12] Recomendaciones Bindings. [Online] 26 de Mayo de 2016. http://www.c-sharpcorner.com/UploadFile/746765/wcf-services-choosing-the-appropriate-wcf-binding/ (Vid. Pág.31)

[13] ASP.NET MVC. [Online] 26 de Mayo de 2016. http://www.w3schools.com/aspnet/mvc_intro.asp (Vid. Pág. 36

[14] Repositorio GitHub.NET Core. [Online] 29 de abril de 2016. https://github.com/dotnet/corefx. (Vid. Pág.37)

[15] Documentación Microsoft Controladores [Online] 23 de mayo de 2016. http://www.asp.net/mvc/overview/older-versions-1/controllers-and-routing/aspnet-mvc-controllers-overview-cs (Vid. Pág. 38

[16] Página web oficial de la librería JsonSerializer [Online] 23 de mayo de 2016. http://www.newtonsoft.com/json (Vid. Pág. 49

[17] Herramienta WCF Connected Service. [Online] 23 de mayo de 2016. https://blogs.msdn.microsoft.com/webdev/2015/12/15/wcf-connected-service-visual-studio-extension-preview-for-asp-net-5-projects/ (Vid. Pág.74)

# Glosario

Credenciales: Consisten en el uso de nombre de usuario o dirección de correo y contraseña para comprobar que un usuario es quien dice ser. (Vid. Pág.7, 13, 15, 19, 20, 32, 41, 42, 43, 44, 45, 48, 54, 55, 56, 59, 76)

Credenciales Oauth: Consisten en un identificador de usuario y una dirección de correo electrónico. (Vid. Pág. 55, 56)

Elo: Es un método matemático, basado en cálculo estadístico, para calcular la habilidad relativa de los jugadores de juegos como el ajedrez. (Vid. Pág. 8, 9, 10, 18, 20, 67, 70)

Experiencia: La experiencia de un usuario simboliza su experiencia de juego y la cantidad de partidas en las que ha participado. (Vid. Pág.8, 9, 18, 20, 67, 70)

Honor: El Honor de un usuario simboliza su conducta o comportamiento general que ha demostrado en partidas anteriores y permite al resto de jugadores prever el tipo de jugador que es. (Vid. Pág. 9, 10, 18, 20, 61, 64, 65)

OASIS: Organization for the Advancement of Structured Information Standards (Organización para el Avance de Estándares de Información Estructurada, en idioma castellano), es un consorcio internacional sin fines de lucro que se orienta al desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y servicios web. (Vid. Pág. 28)

Token: Consiste en una secuencia alfanumérica, generado de forma aleatoria, que puede ser de una validez temporal. (Vid. Pág. 19 , 32, 55, 56, 56, 58, 59, 77, 77, 86 )

W3C. World Wide Web Consortium, es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a largo plazo. (Vid. Pág.28)

WS-Security: Es un protocolo que contiene especificaciones sobre cómo debe garantizarse la integridad y seguridad en mensajería de Servicios Web. (Vid. Pág. 28, 31, 41 )

# Anexo A: Pruebas Servicio web Cuentas de Usuario

| | | |
|---|---|---|
| **Número de prueba** | RF01.1 `createUserAndAuthorization` | |
| **Descripción** | Ya existe un usuario con la dirección de email dada. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción, avisando de que el usuario ya se encuentra registrado. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF01.2 `createUserAndAuthorization` | |
| **Descripción** | No existe un usuario con la dirección email dado, pero el nombre de usuario ya existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción avisando de que el nombre de usuario está en uso. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF01.3 `createUserAndAuthorization` | |
| **Descripción** | No existe un usuario con la dirección de email dada, y el nombre de usuario no está en uso y se requiere confirmación de cuenta | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio crea la cuenta de usuario pendiente de validación junto con su perfil y devuelve un código de confirmación de cuenta. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF01.4 `createUserAndAuthorization` | |
| **Descripción** | No existe un usuario con la dirección de email dada, y el nombre de usuario no está en uso y no se requiere confirmación de cuenta | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio crea la cuenta de usuario validada junto con su perfil y devuelve un string vacio | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF02.1 `createAuthorization` | |
| **Descripción** | Email de usuario existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que ya existe un usuario con la dirección de email dada. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF02.2 `createAuthorization` | |
| **Descripción** | Email de usuario no existe, pero la contraseña tiene una longitud inferior a la establecida. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción avisando de la contraseña es incorrecta. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF02.3 `createAuthorization` | |
| **Descripción** | Email de usuario no existe, y la contraseña es correcta | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio crea la autorización para el usuario y lo guarda en la base de datos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.1 addUserToFriendList | |
| **Descripción** | Id de Usuario no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario al que se le intenta añadir a la lista de amigos no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.2 addUserToFriendList | |
| **Descripción** | ID de Usuario existe, pero la id del usuario amigo no. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción avisando de que el usuario que se intenta añadir a la lista de amigos, no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.3 addUserToFriendList | |
| **Descripción** | Usuario se intenta añadir así mismo a la lista de amigos | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanzara una excepción avisando de que un usuario no puede añadirse así mismo a la lista de amigos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.4 addUserToFriendList | |
| **Descripción** | Usuario existe, e intenta añadir a alguien que ya está en la lista de amigos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio no duplica el amigo. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.5 addUserToFriendList | |
| **Descripción** | Usuario existe, e intenta añadir a un usuario existente que no está en la lista de amigos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | Añade el usuario a la lista de amigos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF03.6 addUserToFriendList | |
| **Descripción** | Usuario existe, y intenta añadir a un usuario existente que está en la lista de usuarios bloqueados. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | Añade el usuario a la lista de amigos y lo elimina de la lista de usuarios bloqueados | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.1 addUserToBlockList | |
| **Descripción** | Id de Usuario no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario al que se le intenta añadir a la lista de amigos no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.2 addUserToBlockList | |
| **Descripción** | ID de Usuario existe, pero la id del usuario a bloquear no. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción avisando de que el usuario que se intenta añadir a la lista de amigos, no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.3 addUserToBlockList | |
| **Descripción** | Usuario se intenta añadirse así mismo a la lista de usuarios bloqueados | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanzara una excepción avisando de que un usuario no puede añadirse así mismo a la lista de usuarios bloqueados. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.4 addUserToBlockList | |
| **Descripción** | Usuario existe, e intenta añadir a alguien que ya está en la lista de usuarios bloqueados. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio no actualiza la lista de usuarios bloqueados. (no duplica el usuario en la lista). | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.5 addUserToBlockList | |
| **Descripción** | Usuario existe, e intenta añadir a un usuario existente que no está en la lista de usuarios bloqueados. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá actualizar la lista de usuarios bloqueados añadiendo al usuario a bloquear. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF04.6 addUserToBlockList | |
| **Descripción** | Usuario existe, e intenta añadir a un usuario existente que está en la lista de amigos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio elimina al usuario de la lista de amigos y lo añade a la lista de usuarios bloqueados. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF05.1 updateProfileData | |
|---|---|---|
| **Descripción** | Modificación de datos para un usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF05.2 updateProfileData | |
|---|---|---|
| **Descripción** | Modificación de datos para un usuario existente. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá guardar los cambios para el usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF06.1 confirmAccount | |
|---|---|---|
| **Descripción** | Confirmación de cuenta para un usuario no existente. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF06.2 confirmAccount | |
|---|---|---|
| **Descripción** | La cuenta existe, pero el código de confirmación no coincide con el de la cuenta del usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servidor deberá devolver False. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF06.3 confirmAccount | |
|---|---|---|
| **Descripción** | La cuenta existe y el código de validación es correcto. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio valida el usuario, y elimina el token de validación. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF07.1 checkUserPassword | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando que el usuario o la contraseña no son validos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF07.2 checkUserPassword | |
| **Descripción** | Usuario existente, contraseña invalida | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando que el usuario o la contraseña no son validos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF07.3 checkUserPassword | |
| **Descripción** | Usuario y contraseña correctas | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá devolver un objeto User con los datos del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF08.1 generateUserPasswordRecoveryToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF08.2 generateUserPasswordRecoveryToken | |
| **Descripción** | Usuario existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá generar, almacenar en la base de datos y devolver un token de recuperación de cuenta de usuario. Y actualizar la fecha de password_reset_token_expiration a la fecha actual más un día. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF09.1 checkRecoveryPasswordToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF09.2 checkRecoveryPasswordToken | |
| **Descripción** | Usuario existente pero el token es incorrecto | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá devolver False | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF09.3 checkRecoveryPasswordToken | |
| **Descripción** | Usuario existe y el token es válido, pero ha expirado la fecha de validación. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá False. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF09.4 checkRecoveryPasswordToken | |
| **Descripción** | Usuario existe y el token es válido. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá True. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF10.1 changeUserPasswordByToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF10.2 changeUserPasswordByToken | |
| **Descripción** | El usuario existe, pero el código de recuperación de contraseña es inválido. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una excepción avisando de que el usuario o la respuesta secreta es inválida. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF10.3 changeUserPasswordByToken | |
| **Descripción** | El usuario existe, el código de recuperación de contraseña es válido, pero la fecha de validación del código ha expirado. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una excepción avisando de que el usuario o la respuesta secreta es inválida. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF10.4 changeUserPasswordByToken | |
| **Descripción** | Usuario existe, el código de recuperación de contraseña es válido y la fecha de validación del código no ha expirado. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve true y actualiza la contraseña del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF11.1 changeUserPassword | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF11.2 changeUserPassword | |
| **Descripción** | El usuario existe, pero la contraseña es incorrecta | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve False y suma uno al campo failed_password_attempt_count de la base de datos y actualiza la fecha del campo failed_password_attempt_last_date. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF11.3 changeUserPassword | |
| **Descripción** | Usuario existe y la contraseña es correcta. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá true y actualizara la contraseña del usuario. También reinicia el campo failed_password_attempt_count y lo deja a 0. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF12.1 createOrUpdateOauthAccount | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF12.2 createOrUpdateOauthAccount | |
| **Descripción** | El usuario existe pero no tiene creado el campo oauth para dicho Oauth Provider en la base de datos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio insertara una nueva tupla en la base de datos con los datos de oauth para el usuario en la correspondiente tabla. | |
| **Resultados obtenido** | ☐Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF12.3 createOrUpdateOauthAccount | |
| **Descripción** | El usuario existe y tiene creado el campo del provider oauth en la base de datos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio actualiza los datos de Oauth para el usuario. | |
| **Resultados obtenido** | ☐Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF13.1 getAccountConfirmationToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF13.2 getAccountConfirmationToken | |
| **Descripción** | El usuario existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá el valor del token de confirmación del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF14.1 LockUserAccount | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF14.2 LockUserAccount | |
| **Descripción** | El usuario existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá cambiar el valor de bloqueo de el usuario ha true, y actualizar la fecha de last_locked_date a la fecha actual. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF15.1 UnlockUserAccount | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF15.2 UnlockUserAccount | |
| **Descripción** | El usuario existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá cambiar el valor de bloqueo de el usuario a False. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF16.1 generateUserGameToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF16.2 generateUserGameToken | |
| **Descripción** | El usuario existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá generar un token de juego para el usuario, y devolverlo. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF17.1 checkUserGameToken | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF17.2 checkUserGameToken | |
| **Descripción** | El usuario existe y el token de el juego es incorrecto | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá devolver False. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF17.3 checkUserGameToken | |
| **Descripción** | Usuario existe y el token del juego es correcto. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá true y elimina el token del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF18.1 removeUserFromFriendList | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF18.2 removeUserFromFriendList | |
| **Descripción** | El usuario existe y pero el usuario que se intenta eliminar no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario que se intenta eliminar no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF18.3 removeUserFromFriendList | |
| **Descripción** | Usuario existe y el usuario que se intenta eliminar de la lista también, pero no está en la lista. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio mantiene la lista de amigos del usuario como estaba. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF18.4 removeUserFromFriendList | |
| **Descripción** | Usuario existe y el usuario que se intenta eliminar de la lista también y está en la lista. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio elimina de la lista de amigos del usuario, el usuario que se desea eliminar. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF19.1 removeUserFromBlockList | |
| **Descripción** | Usuario no existente | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá lanzar una excepción, avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF19.2 removeUserFromBlockList | |
| **Descripción** | El usuario existe y pero el usuario que se intenta eliminar no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción, avisando de que el usuario que se intenta bloquear no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF19.3 removeUserFromBlockList | |
| **Descripción** | El usuario existe y el usuario que se intenta eliminar también, pero el usuario que se intenta eliminar no está en la lista | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio mantiene la lista de usuarios bloqueados del usuario tal y como está. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF19.4 removeUserFromBlockList | |
| **Descripción** | Usuario existe y el usuario que se intenta eliminar de la lista también, y el usuario que se intenta eliminar está en la lista. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio elimina de la lista de usuarios bloqueados del usuario, el usuario que se desea eliminar. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF20.1 isUsernameInUse | |
| **Descripción** | No está en uso el nombre de usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve false | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF20.2 isUsernameInUse | |
| **Descripción** | El nombre de usuario está en uso. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve true. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF21.1 removeUserOauth | |
| **Descripción** | No existe el usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF21.2 removeUserOauth | |
| **Descripción** | El usuario existe, pero no tiene creado el provider que se quiere eliminar | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio no lanza excepción ni hace nada. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF21.3 removeUserOauth | |
| **Descripción** | El usuario existe, y también existe el provider que se desea eliminar | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio elimina de la base de datos las credenciales Oauth del usuario para dicho provider. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF22.1 getUserByID | |
| **Descripción** | No existe la id de usuario. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF22.2 getUserByID | |
| **Descripción** | Existe la id del usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto User con los datos del usuario | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF23.1 getUserByEmail | |
| **Descripción** | No existe el correo del usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el usuario no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF23.2 getUserByEmail | |
| **Descripción** | Existe el correo del usuario | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto User con los datos del usuario | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF24.1 createUserRole | |
| **Descripción** | Existe el rol que se quiere crear | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el rol ya existe | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF24.2 createUserRole | |
| **Descripción** | El rol no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio crea el rol en la base de datos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF25.1 removeUserRole | |
| **Descripción** | No existe el rol que se quiere eliminar | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el rol no existe | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF25.2 removeUserRole | |
| **Descripción** | El rol existe, pero está en la lista de roles protegidos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio no elimina el rol y lanza una excepción. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF25.3 removeUserRole | |
| **Descripción** | El rol existe, y no está en la lista de roles protegidos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio elimina el rol de la base de datos, así como todas las relaciones entre el rol y los usuarios que perteneciesen a dicho rol. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF26.1 renameUserRole | |
| **Descripción** | No existe el rol que se quiere renombrar. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el rol no existe | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF26.2 renameUserRole | |
| **Descripción** | Existe el rol que se quiere renombrar, pero está en la lista de roles protegidos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción de operación no valida. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF26.3 renameUserRole | |
| **Descripción** | Existe el rol que se quiere renombrar y no está en la lista de roles protegidos. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio renombra el rol y guarda los cambios en la base de datos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF27.1 addUserToRole | |
| **Descripción** | El usuario no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el usuario no existe | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF27.2 addUserToRole | |
| **Descripción** | El usuario existe pero no existe el rol | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el rol no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF27.3 addUserToRole | |
| **Descripción** | El usuario existe así como el rol al que se le quiere añadir | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio añade al usuario al rol dado, y guarda los cambios en la base de datos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF28.1 removeUserFromRole | |
| **Descripción** | El usuario no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que el usuario no existe | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF28.2 removeUserFromRole | |
| **Descripción** | El usuario existe pero no pertenece al rol dado | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio no realiza ninguna acción. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF28.3 removeUserFromRole | |
| **Descripción** | El usuario existe y pertenece al rol dado | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio elimina al usuario del rol y guarda los cambios en la base de datos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.1 checkUserOauth | |
| **Descripción** | El usuario no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio guarda en la base de datos un nuevo usuario y credenciales con los datos recibidos, y devuelve el usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.2 checkUserOauth | |
| **Descripción** | El usuario existe pero no tiene credenciales del Oauth dado | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio crea unas credenciales para el Oauth dado para dicho usuario, lo guarda en la base de datos y devuelve los datos del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.3 checkUserOauth | |
| **Descripción** | El usuario existe, tiene credenciales del Oauth dado, pero el string del provider no coincide. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que las credenciales de Oauth son incorrectas. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.4 checkUserOauth | |
| **Descripción** | El usuario existe, tiene credenciales del Oauth dado, y el string del provider coincide. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve los datos del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.5 addUserReport | |
| **Descripción** | Alguno de los usuarios no existe, y no existe ningún reporte de juego con la id de juego dada. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio deberá<br>• Generar los usuarios que no existan<br>• Almacenar los datos del reporte<br>• Actualizar los porcentajes de los usuarios que contenga el reporte. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.6 addUserReport | |
| **Descripción** | Los usuarios existen, y no existe ningún reporte de juego con la id de juego dada. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio realiza las siguientes acciones:<br>• Almacenar los datos del reporte<br>• Actualizar los porcentajes de los usuarios que contenga el reporte. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF29.7 addUserReport | |
| **Descripción** | Los usuarios existen, pero existe un reporte para dicho id de juego | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción de reporte repetido. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF30.1 getUserHonor | |
| **Descripción** | Id de usuario no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto nulo. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF30.2 getUserHonor | |
| **Descripción** | Existe ID de usuario. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devolverá un objeto tipo honor con los datos del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF31.1 getUserWeightedHonor | |
| **Descripción** | Id de Usuario no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto nulo. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF32.1 getUserWeightedHonor | |
| **Descripción** | Id de Usuario no existe en la aplicación padre pero si en una o más aplicaciones hijas. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto nulo. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF32.2 getUserWeightedHonor | |
| **Descripción** | ID de Usuario existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve la media aritmética del valor de honor en un objeto Honor, de la aplicación que lanza la petición junto con las aplicaciones hijas del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF33.1 getUsersHonor | |
| **Descripción** | Alguna ID de usuario no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanzara una excepción avisando de que alguna id no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF33.2 getUsersHonor | |
| **Descripción** | Las IDS de los usuarios existen. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos de honor de cada usuario pedido. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF34.1 getUsersWeightedHonor | |
| **Descripción** | Alguna ID de usuario no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanzara una excepción avisando de que alguna id no existe. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF34.2 getUsersWeightedHonor | |
|---|---|---|
| **Descripción** | Las IDS de los usuarios existen. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos honor de los usuarios pedidos, los cuales son la media aritmética del valor de honor en la aplicación que lanza la petición junto con las aplicaciones hijas de éste. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF35.1 getTopTenHonorableUsers | |
|---|---|---|
| **Descripción** | El servicio devuelve una lista de objetos honor con los 10 usuarios con un valor de honor más elevado. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | Devuelve una lista de objetos de honor ordenada por el valor de honor de los usuarios.<br>Estos serán los que tienen el mayor número de honor. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF36.1 getTopTenWeightedHonorableUsers | |
|---|---|---|
| **Descripción** | Uno de los usuarios no está en la aplicación que lanzo la petición. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servidor deberá devolver la lista de honor ponderado de los 10 usuarios más honoríficos de la aplicación y aplicaciones hijas. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| **Número de prueba** | RF37.1 getTopTenWeightedHonorableUsers | |
|---|---|---|
| **Descripción** | El servicio devuelve una lista de objetos honor con los 10 usuarios con un valor de honor ponderado más elevado. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servidor deberá devolver la lista de honor ponderado de los 10 usuarios más honoríficos de la aplicación y aplicaciones hijas. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

# Anexo C: Pruebas Servicio web Elo

| | | |
|---|---|---|
| **Número de prueba** | RF38.1 addGameReport | |
| **Descripción** | El reporte del juego ya existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio lanza una excepción avisando de que ya existe un reporte de juego | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF38.2 addGameReport | |
| **Descripción** | El reporte del juego no existe, pero alguno de los usuarios incluidos en el reporte no existe | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio realiza las siguientes acciones:<br>• Crea los usuarios que no estén creados en el servicio con los valores elo, nivel y experiencia por defecto.<br>• El servicio almacena el reporte del juego.<br>• El servicio calcula los nuevos niveles de honor elo y experiencia de los usuarios incluidos en el reporte y los actualiza. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF38.3 addGameReport | |
| **Descripción** | El reporte del juego no existe, y los usuarios incluidos en el reporte si existen. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio realiza las siguientes acciones:<br>• El servicio almacena el reporte del juego.<br>• El servicio calcula los nuevos niveles de honor elo y experiencia de los usuarios incluidos en el reporte y los actualiza. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF39.1 getUserELO | |
| **Descripción** | El usuario no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto Elo con los valores de elo, nivel y experiencia por defecto.. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF39.2 getUserELO | |
| **Descripción** | El usuario existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve un objeto Elo con los valores elo, nivel y experiencia del usuario. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF40.1 getUsersELO | |
| **Descripción** | Uno de los usuarios pedidos no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos ELO de los usuarios pedidos, los usuarios que no existen tendrán los valores por defecto de elo, nivel y experiencia. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF40.2 getUsersELO | |
| **Descripción** | Los usuarios existen | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos Elo con los valores elo, nivel y experiencia de los usuarios pedidos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF41.1 getUsersELO | |
| **Descripción** | Uno de los usuarios pedidos no existe. | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☐Alta ☒Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos ELO de los usuarios pedidos, los usuarios que no existen tendrán los valores por defecto de elo, nivel y experiencia. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |

| | | |
|---|---|---|
| **Número de prueba** | RF41.2 getUsersELO | |
| **Descripción** | Los usuarios existen | |
| **Métodos/Herramientas** | Pruebas de unidad en Visual Studio | |
| **Prioridad** | ☒Alta ☐Media ☐Baja | |
| **Resultado esperado** | El servicio devuelve una lista de objetos Elo con los valores elo, nivel y experiencia de los usuarios pedidos. | |
| **Resultados obtenido** | ☒Satisfactorio | ☐Insatisfactorio |