Propuesta metodológica para motivar a los estudiantes en el estudio de las Redes de Computadores

A. Zafra[1], E. Gibaja[1], M. Luque[1], and L. Toribio[2]

[1] Universidad de Córdoba, azafra@uco.es, egibaja@uco.es, mluque@uco.es
[2] Director de seguridad privada. Máster en Seguridad
ltoribiocastro@gmail.com

Resumen Computer networks is a course included in the curriculum of any Computer Engineering Degree. The main objective of this course is to initiate students in foundations of networks of computers. In order to achieve the competences related to this subject, carrying out practices in which students have to design and develop their own online application is crucial. This paper presents a methodological proposal to implement classic games where several users can play simultaneously from different locations. This proposal aims to engage students in the teaching/learning process to acquire and go in depth the specific skills in this subject.

Keywords: Computer network · client-server model · socket programming

Resumen Actualmente, el estudio de las redes de computadores se encuentra presente en cualquier plan de estudios del Grado en Ingeniería Informática. El objetivo principal de las asignaturas que introducen esta materia es iniciar a los estudiantes en el conocimiento de los fundamentos de las redes de computadores. Para alcanzar las competencias elementales relacionadas con esta asignatura, es importante acompañar su enseñanza con prácticas en la que los estudiantes diseñen y desarrollen su propia aplicación en red. Con la finalidad de motivar a los estudiantes para que se impliquen en el proceso de enseñanza/aprendizaje y sean capaces de adquirir y profundizar las competencias específicas en este tipo de asignaturas, se presenta una propuesta metodológica que implica la implementación de un juego clásico en red, donde varios usuarios pueden jugar simultáneamente desde cualquier localización.

Keywords: Redes de Computadores · modelo cliente-servidor· programación con sockets

1. Introducción

Cada vez son más los estudios que demuestran la relevancia de un aprendizaje que motive e incentive al estudiante, de forma que éste se convierta en una

parte importante en el proceso enseñanza/aprendizaje. Muestra de ello, es la gran cantidad de propuestas que actualmente se encuentran relacionadas con la gamificación [**?**,**?**]. Estas propuestas intentan que el alumno adquiera una serie de competencias, de forma divertida y amena, mientras está jugando. En esta línea, las prácticas y ejercicios que les planteamos a los estudiantes, deben motivarles para continuar aprendiendo, siendo necesario pensar en prácticas que les puedan resultar alentadoras y donde el alumno o alumna, pueda ver un resultado práctico y funcional de lo que está aprendiendo.

Es indudable que las redes de computadores están inmersas en nuestro día a día, utilizándose en casi cualquier ámbito de nuestra vida. La estructura interna en la que se basa todo su funcionamiento resulta transparente para los usuarios finales, quienes hacen uso de ellas a través de la mayoría de las aplicaciones que utilizan habitualmente, pero no conocen cómo se está produciendo realmente el envío y la recepción de la información. Este gran auge del uso de aplicaciones que emplean las redes de computadores, ha producido que asignaturas que enseñen los fundamentos de las mismas se impartan como asignaturas troncales en cualquier plan de estudios en un Grado en Ingeniería Informática. Su contenido suele englobar competencias tales como conocer y aplicar las características, estructura, organización y funcionalidades de los sistemas distribuidos, las redes de computadores e Internet, y diseñar e implementar aplicaciones basadas en ellas. La consecución de estas competencias implica que los estudiantes comprendan los diferentes protocolos, servicios, problemas, soluciones y los estándares más importantes que permiten que las computadores se comuniquen entre sí y ser capaces de programar aplicaciones en red utilizando los servicios orientados y no orientados a la conexión.

Estos contenidos, que suelen suponer la primera toma de contacto de los estudiantes con los modelos y arquitecturas que hacen uso de la red subyacente y de todas su conexiones, les suelen resultar complejos. Los estudiantes, como usuarios finales, están acostumbrados a usar aplicaciones en red, pero también son ajenos al funcionamiento interno y la programación de dichas aplicaciones. En este escenario, los estudiantes requieren conocer los modelos y/o protocolos con los que se va a trabajar, y conocer cómo se produce la comunicación para poder desarrollarla de la forma más eficiente posible.

En este artículo se presentan dos propuestas que pueden ser utilizadas en los guiones de prácticas de las asignaturas de introducción a las Redes de Computadores. Concretamente, se basa en definir dos juegos clásicos, que permitan que el alumno pueda diseñar y desarrollar su propio protocolo, que debe ser compartido por los extremos de la comunicación, profundicen en los conceptos de servicio, protocolo e interfaces, y que hagan uso de los servicios ofrecidos por la capas inferiores. En este caso, de la capa de transporte del protocolo TCP/IP [**?**]. El desarrollo de esta aplicación resulta muy motivadora para los alumnos, ya que al finalizar, disponen de una aplicación que pueden compartir con sus compañeros en cualquier momento y desde cualquier lugar.

En la sección **??** se determinan las tecnologías que se deben utilizar, comentando el modelo TCP/IP, la arquitectura cliente-servidor, y la programación con

sockets, como herramientas que nos permiten diseñar y desarrollar nuestras propias aplicaciones. En la sección **??** se explica la base de los juegos propuestos, detallando el protocolo que debe seguirse para desarrollar el juego del bingo y del dominó, que son las dos propuestas planteadas, y que permiten establecer las reglas de comunicación que deben producirse para que diferentes usuarios, desde diferentes localizaciones puedan conectarse, y jugar una partida en red. Finalmente, la sección **??** dará la conclusiones obtenidas.

2. Tecnologías disponibles

En esta sección se describen las principales tecnologías que se pueden utilizar para llevar a cabo las propuestas que se plantean en la siguiente sección. Debido a que la mayoría de las aplicaciones de Internet están basadas en la arquitectura TCP/IP [**?**], y siguen el modelo cliente/servidor, será en estos sistemas en los que nos basemos, para facilitar la implementación de las aplicaciones en red.

2.1. Modelo TCP/IP

TCP/IP [**?**] cuyo nombre viene dado por los dos protocolos más conocidos que emplea su arquitectura: Protocolo de Control de Transmisión (TCP, *Transmission Control Protocol*) y el Protocolo de Internet (IP, *Internet Protocol*), representa un conjunto de protocolos que supone un estándar abierto que recibió el aval del sector de redes y fue ratificado, o aprobado, por una organización de estandarización. Esta suite de protocolos permiten la interconexión de los dispositivos en Internet.

El conocimiento del conjunto de protocolos TCP/IP resulta fundamental para las personas que desean administrar o brindar soporte técnico a una red TCP/IP. Se trata de un sistema en capas independientes (ver Figura **??**). Cada

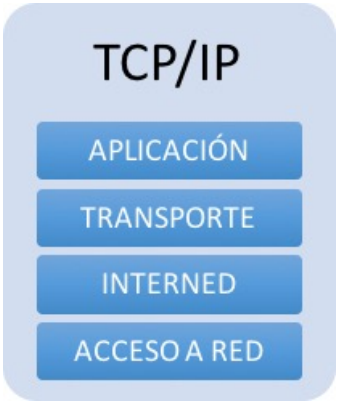

Figura 1. Modelo TCP/IP

capa del modelo se comunica con un nivel adyacente (superior o inferior), utilizando el servicio de la capa inferior y proporcionándoselos a la capa superior. El modelo TCP/IP contiene cuatro capas que de abajo a arriba son: **capa de acceso a la red:** especifica la forma en la que los datos deben enrutarse, sea cual sea el tipo de red utilizado; **capa de Internet:** es responsable de proporcionar el paquete de datos. Debe realizar una serie de acciones sobre la información que recibe del nivel anterior para luego acometer el envío al nivel inferior; **capa de transporte:** brinda los datos de enrutamiento, junto con los mecanismos que permiten conocer el estado de la transmisión. Es la encargada de ofrecer una comunicación entre extremos de programas de aplicación. Comprende a los protocolos TCP y UDP; y **capa de aplicación:** es el más alto dentro del protocolo que nos ocupa y en él se encuentran una serie de aplicaciones que tienen la capacidad de acceder a diversos servicios a los que se puede acceder vía Internet.

Las aplicaciones que se proponen, serán implementadas en la capa de aplicación, haciendo uso de los servicios de la capa de transporte, que nos permite el diseño de aplicaciones que emplea un servicio no orientado a conexión (usando el protocolo *UDP*) o bien usando un servicio orientado a conexión (haciendo uso del protocolo *TCP*). Dependiendo del tipo de aplicación que se diseñe, puede resultar conveniente utilizar un servicio u otro.

2.2. Arquitectura Cliente-Servidor

A la hora de comunicar diferentes dispositivos podemos encontrar varias arquitecturas, las que son más ampliamente utilizadas son la arquitectura cliente-servidor, y la arquitectura peer-to-peer (P2P). La primera consiste en una aplicación principal que actúa como servidor manteniéndose a la espera de ofrecer un servicio, mientras que el cliente o clientes, son los que realizarán peticiones del servicio ofrecido. En este tipo de arquitectura es común tener un único programa servidor y cientos de aplicaciones clientes que realizan peticiones del servicio ofrecido (ver figura **??**). En el caso de la arquitectura P2P, está basada en la unión de ordenadores en una red donde cada ordenador realiza la labor de cliente y servidor al mismo tiempo. Existen redes P2P híbridas que utilizan un servidor central para mantener la información sobre las redes o nodos. Entre las principales ventajas con las que cuenta la arquitectura cliente-servidor, podemos citar: *centralización del control*, los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no puede dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos; *escalabilidad*, se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores); y *fácil mantenimiento*, las diferentes funciones y responsabilidad se encuentra en los servidores, que pueden ser reemplazados, reparados o actualizados, sin que los clientes se verán afectados por ese cambio.

En esta sección, nos centraremos en la arquitectura cliente-servidor, que es la que proponemos para al implementación de las aplicaciones que se plantean.

Figura 2. Aplicación Cliente-Servidor

2.3. Socket de Berkeley

Existen diversas APIs que se pueden utilizar para TCP/IP, para esta propuesta se sugiere el uso de sockets BSD (Berkeley sockets) [**?**]. Con la interfaz de los sockets disponemos de un mecanismo estándar para utilizar la pila de protocolos TCP/IP y poder intercambiar información de forma eficiente a través de Internet, independientemente del sistema operativo y de la red local a la cual se encuentre conectado el host que las ejecuta.

La interfaz de los sockets añade una nueva abstracción para la comunicación a través de la red, el socket. Cada socket activo se identifica por un entero denominado su descriptor de socket. Desde el punto de vista funcional, el socket se define como un punto terminal al que pueden *conectarse* dos procesos para comunicarse entre sí. Dado que en un mismo dispositivo/ordenador podemos estar ejecutando de forma simultánea diferentes aplicaciones que utilizan Internet para comunicarse, resulta imprescindible identificar cada socket con una dirección diferente. Así, un socket se va a identificar por la dirección IP del dispositivo, más un número de puerto.

La especificación de un socket requiere:

- Familia de Protocolo. Representa el tipo de protocolo se va a utilizar para realizar las distintas comunicaciones. Los protocolos TCP/IP constituyen una única familia representada por la constante PF_INET (*Protocol Family Internet*).
- Tipo de servicio. Representa el tipo de servicio que se desea: orientado a la conexión (*SOCK_STREAM*) o no orientado a conexión (*SOCK_DGRAM*).
- Familia de direcciones finales. Representa la dirección final de una comunicación de una manera distinta. En el protocolo IP, una dirección final se especifica usando la dirección IP de la máquina y el número de puerto de protocolo que usará el programa. direcciones.

Los sockets permiten implementar una arquitectura cliente/servidor. Un socket queda definido por una dirección IP, un protocolo y un número de puerto. Estos parámetros configuran las condiciones necesarias para que el programa servidor y el programa cliente puedan leer y escribir información coordinadamente. Las primitivas de las que consta este API y que nos permit trabajar

tanto con sistemas orientados a conexión, como no orientadas a conexión, se puede consultar en el manual de Miller et al. [**?**].

3. Propuestas metodológicas

En esta sección, se describen dos propuestas metodológicas que pueden ser empleadas para que los estudiantes del Grado en Ingeniería Informática adquieran las competencias necesarias a los estudiantes en la asignatura relacionada con los fundamentos de las Redes de Computadores. Se basan en el diseño y desarrollo de dos juegos clásicos que permite que se puedan jugar múltiples partidas, con múltiples jugadores, de forma simultánea.

3.1. El juego del bingo

Especificación del juego. El juego del bingo está formado por: 90 bolas numeradas del 1 al 90, y por cartones de bingo. Cada cartón (ver figura **??**) está formado por tres filas y nueve columnas. Cada fila tiene cinco valores y cuatro espacios se dejan en blanco, con lo que cada cartón tiene finalmente 15 números elegidos al azar, y completados con el siguiente procedimiento:

- La primera columna contiene números del 1 al 9,
- La segunda del 10 al 19,
- La tercera, del 20 al 29,
- La cuarta del 30 al 39,
- La quinta del 40 al 49,
- La sexta del 50 al 59,
- La séptima del 60 al 69,
- La octava del 70 al 79 y
- La novena del 80 al 90

Cada jugador recibe uno o más cartones, y paulatinamente se van sacando una de las 90 bolas disponibles. El objetivo del juego consiste en marcar los números que se encuentran en nuestro cartón conformen vayan saliendo, y lograr ser el primero en completar una línea, dos líneas y todo el cartón, que sería *hacer un bingo*.

	16	23			54	66		82
		20		44	52		72	80
7			32	40		61	75	

Figura 3. Cartón del juego del bingo

Implementación del juego. La comunicación entre los clientes del juego del bingo se realizará bajo el protocolo de transporte TCP, siguiendo la arquitectura cliente/servidor, donde los jugadores (clientes) se conectan al servidor para jugar. El protocolo de comunicación que se estable es el siguiente:

1. Un cliente se conecta al servicio y si la conexión ha sido correcta el servidor devuelve *+0k. Usuario conectado.*
2. Para poder acceder a los servicios es necesario identificarse mediante el envío del usuario y clave para que el sistema lo valide. Los mensajes que deben indicarse son: *USUARIO* <*usuario*> para indicar el usuario, tras el cual el servidor enviará *+Ok. Usuario correcto* o *−ERR. Usuario incorrecto.* En caso de ser correcto el siguiente mensaje que se espera recibir de dicho usuario es *PASSWORD* <*password*>, donde el servidor responderá con el mensaje: *+Ok. Usuario validado* o *−ERR. Error en la validación.*
3. Un usuario nuevo podrá registrarse mediante el mensaje *REGISTRO −u* <*usuario*> *−p* <*password*>. El servidor llevará un control para evitar colisiones con los nombre de usuarios ya existentes, y responderá con el mensaje: *+Ok. Usuario registrado* o *−ERR. Error en el registro.*
4. Una vez conectado y validado, el cliente podrá solicitar jugar una partida indicando el mensaje *INICIAR-PARTIDA*. Recibido este mensaje en el servidor, éste comprobará:
   - Si con esta petición, ya se forma un grupo de cuatro jugadores. En cuyo caso, mandará un mensaje a cada uno de ellos, para indicarle que la partida va a comenzar: *+Ok. Empieza la partida.*
   - Si todavía falta algún jugador para iniciar la partida, mandará un mensaje al nuevo usuario, especificando que tiene su petición y que está a la espera de la conexión de otros jugadores: *+Ok. Petición Recibida. Quedamos a la espera de más jugadores.*
5. Una vez que están todos los jugadores, la partida comenzará. Para ello, se repartirá un cartón por jugador, el servidor enviará a cada jugador un mensaje, con el siguiente formato *CARTON* |Num1, Num2, Num3, X, X, X, Num4, X, Num5; Num6, X, X, X, Num7, Num8, X, Num9, Num10; Num11, X, Num12, Num13, X, Num14, X, X, Num15| y preparará las 90 bolas para iniciar el juego. El cartón deberá cumplir con las restricciones que se han especificado anteriormente.
6. Una vez comenzado el juego, el servidor irá cantando los números, de forma paulatina, enviando el mensaje *NUMERO-OBTENIDO* <Número de dos dígitos> a todos los jugadores de dicha partida.
7. En el momento que un jugador complete una línea, le mandará el mensaje *UNA-LINEA* al servidor, el cual informará a todos los jugadores de dicha partida con el mensaje: *+Ok. Jugador* <Nombre> *ha cantado una línea.* Si no es el primer jugador en obtener línea, el servidor enviará un mensaje al jugador para indicarle que no es un comando válido *−Err. El comando UNA-LINEA no procede.*
8. En el momento que un jugador complete dos líneas, le mandará el mensaje *DOS-LINEAS* al servidor, el cual informará a todos los jugadores de dicha

partida con el siguiente mensaje: *+Ok. Jugador* <Nombre> *ha cantado dos líneas*. Si no es el primer jugador en obtener dos líneas, el servidor enviará un mensaje al jugador para indicarle que no es un comando válido *−Err. El comando DOS-LINEAS no procede*.

9. En el momento que un jugador complete todo el cartón, le mandará el mensaje *BINGO* al servidor, el cual informará a todos los jugadores de dicha partida con el mensaje: *+Ok. Jugador* <Nombre> *ha cantado bingo*, y terminará la partida enviando el mensaje *+Ok. Partida finalizada*.
10. Si con la misma bola, varios usuarios obtienen línea, dos líneas o bingo y ambos jugadores mandan el mensaje en ese momento, a ambos jugadores se le considerará la misma condición de haber cantado línea, dos líneas o bingo, con independencia del mensaje que llegó primero.
11. Es responsabilidad del servidor comprobar que ningún jugador se ha equivocado a la hora de determinar la línea, dos líneas o el bingo. En caso de error por parte del jugador, el servidor le enviará al usuario el mensaje *-Err. Su cartón no satisface* <UNA-LINEA/DOS-LINEAS/BINGO> *con los números actuales*.
12. Un jugador siempre podrá salir de la aplicación en cualquier momento. De este modo, el comando *SALIR* al servidor, implicará:
    - Si estaba esperando para comenzar una partida, debe eliminarse de la espera de dicha partida y se le mandará un mensaje *+Ok. Desconexión procesada*.
    - Si estaba jugando una partida se eliminará el jugador de la partida y el resto de jugadores continuará con la misma, dejando un mensaje a todos los jugadores *+Ok. Usuario* <Nombre> *ha abandonado la partida*.
13. Cualquier mensaje que no use uno de los especificadores detallados, generará un mensaje de *−ERR* por parte del servidor.

Restricciones. Algunas restricciones a la hora de desarrollar la aplicación son:

- El protocolo deberá permitir mandar mensajes de tamaño arbitrario. Teniendo como tamaño máximo de envío una cadena de longitud 250 caracteres.
- El servidor aceptará servicios en el puerto 2050.
- El servidor debe permitir la conexión de varios clientes simultáneamente.
- Solamente se admitirán partidas con cuatro jugadores, y cada jugador jugará con un único cartón.
- Los jugadores que se conecten y soliciten jugar una partida se quedarán en espera, y en el momento que existan cuatro jugadores esperando comenzarán una partida.
- Se admiten hasta 10 partidas simultáneas, y un máximo de 40 jugadores podrán estar conectados simultáneamente en el servidor.
- Todos los mensajes mandados al servidor con respecto a la conexión y validación o el desarrollo del juego recibirán una respuesta indicando que ha sido correcto *+OK. Texto informativo*, o que ha habido algún error *−ERR. Texto informativo*.

- Debe tenerse presente, que en la implementación no se considerará encriptación de la información, algo que sería necesario en el envío de información segura.

3.2. El juego del dominó

Especificación del juego. El juego de dominó requiere de 28 fichas rectangulares. Cada ficha está dividida en 2 espacios iguales en los que aparece una cifra de 0 hasta 6. Las fichas cubren todas las combinaciones posibles con estos números. Pudiéndose jugar con 2, 3 o 4 jugadores, o por parejas.

El objetivo del juego es que cada jugador coloque todas sus fichas en la mesa antes que los contrarios y sumar puntos. El jugador que gana una partida, suma puntos según las fichas que no han podido colocar los oponentes. La partida termina cuando un jugador o pareja alcanza la cantidad de puntos indicada en las opciones de mesa.

Implementación del juego. La comunicación entre los clientes del juego del dominó se realizará bajo el protocolo de transporte TCP. La aplicación será implementada según una arquitectura cliente/servidor, donde los jugadores (los clientes) se conectan al servicio (el servidor).

El protocolo de comunicación establecido determina las siguientes reglas de comunicación:

1. Los puntos del 1) al 4) serían similares a los especificados en la sección **??**.
2. Una vez conectado y validado, el cliente podrá llevar a cabo una partida en el juego indicando un mensaje: *INICIAR-PARTIDA*. Recibido este mensaje en el servidor, éste se encargará de comprobar las personas que tiene pendiente para comenzar una partida
   - Si con esta petición, ya se forma un grupo de cuatro jugadores, el servidor mandará un mensaje a cada uno de los jugadores, para indicarle que la partida va a comenzar: *+Ok. Empieza la partida.* El orden de los usuarios en el juego será el mismo orden en el que se ha realizado la conexión.
   - Si todavía falta algún jugador para iniciar la partida, el servidor mandará un mensaje al nuevo usuario, especificando que tiene su petición y que está a la espera de la conexión de otros jugadores: *+Ok. Petición Recibida. Quedamos a la espera de más jugadores.*
3. Una vez comenzada la partida, las 28 fichas del dominó serán repartidas al azar (7 fichas a cada jugador). El jugador que comience será el que tenga el seis doble.
4. En su turno, cada jugador colocará una de las fichas que tiene disponible, con la restricción de que dos piezas solamente pueden colocarse juntas cuando los cuadrados adyacentes son del mismo valor. La sintaxis para colocar una ficha será *COLOCAR-FICHA* $|valor1|valor2|, extremo$. Siendo valor1 y valor2, los valores que tiene la ficha que se quiere colocar y extremo indicará si la va a colocar a la derecha o a la izquierda.

5. Cada vez que un jugador envíe una ficha, el servidor reenviará la información del nuevo tablero a cada uno de los jugadores. El tablero será una cadena de texto formada por la secuencia de fichas que representan el estado de la partida. Por ejemplo, el mensaje |2|1||1|1||1|4||4|4| representa que hay cuatro fichas colocadas, concretamente, las fichas |2|1|, |1|1|, |1|4| y |4|4|. Por tanto, el jugador deberá tratar de poner una ficha que tenga como valor un 2 (por la izquierda) o un 4 (por la derecha).
6. El servidor será el encargado de comprobar que la ficha realmente pertenece a dicho jugador, y de que se puede colocar. En caso de que la ficha no pueda colocarse, el servidor devolverá: *-Err. La ficha no puede ser colocada*. En caso de que esa ficha no pertenezca a dicho jugador, el servidor enviará el mensaje: *-Err. La ficha no se encuentra entre sus fichas*.
7. El mensaje para indicar que es el turno de un jugador será enviado por el servidor al jugador específico mediante el mensaje: *+Ok. Turno de partida*.
8. Si un jugador intenta colocar una ficha cuando no es su turno, el servidor enviará el mensaje: *-Err. No es su turno*.
9. Si un jugador no puede colocar ninguna ficha en su turno, tendrá que pasar el turno al siguiente jugador, con el mensaje *PASO-TURNO*.
10. La partida continúa con los jugadores colocando sus fichas hasta que se presenta alguna de las situaciones siguientes:
    - Cuando un jugador coloca su última ficha en la mesa, el jugador gana la partida. El servidor mandará un mensaje indicando: *+Ok. Partida Finalizada.* <Jugador> *ha ganado la partida*, siendo <jugador> el identificador del jugador que ha ganado.
    - Todos los jugadores tienen fichas, pero ninguno de ellos puede continuar la partida. Esto ocurre cuando los números de los extremos ya han sido jugados 7 veces. En ese momento la partida está cerrada. Los jugadores contarán los puntos de las fichas que les quede; el jugador con menos puntos será el ganador. El servidor mandará el mensaje: *+Ok. Partida Finalizada.* <Jugador> *ha ganado la partida*, siendo <jugador> el identificador del jugador que ha ganado.
11. Un jugador siempre podrá salir de la aplicación en cualquier momento. De este modo, el comando *SALIR* implicará:
    - Si estaba esperando para comenzar una partida, debe eliminarse de la espera de dicha partida y se le mandará un mensaje *+Ok. Desconexión procesada*.
    - Si estaba jugando una partida, se anulará dicha partida, dejando un mensaje a todos los jugadores *+Ok. La partida ha sido anulada* y toda la información relativa a dicha partida será eliminada, y el resto de jugadores deberán indicar que quieren iniciar una nueva partida, para ponerse a la espera de que les sea asignada una.
    - En el momento que la partida es finalizada, los jugados continúan conectados, y si desean jugar una nueva partida deberán indicarlo con el mensaje: *INICIAR-PARTIDA*.
12. Cualquier mensaje que no use uno de los especificados, generará un mensaje: *-ERR. Comando no reconocido* por parte del servidor.

Restricciones a tener en cuenta en el juego. Algunas restricciones a la hora de desarrollar la aplicación son:

- El protocolo deberá permitir mandar mensajes de tamaño arbitrario. Teniendo como tamaño máximo de envío una cadena de longitud 250 caracteres.
- El servidor aceptará servicios en el puerto 2051.
- El servidor debe permitir la conexión de varios clientes simultáneamente.
- Solamente se admitirán partidas con cuatro jugadores (no se tendrá en cuenta partidas con un número menor de jugadores inicialmente, ni la consideración del juego por parejas).
- En el momento que existan cuatro jugadores conectados podrán comenzar una partida.
- Se admiten hasta 10 partidas simultáneas y hasta 40 jugadores conectados simultáneamente.
- Una partida finaliza cada vez que un jugador coloque todas sus fichas, y dicho jugador será designado el ganador de dicha partida.
- No se considerará la suma acumulativa de puntos en diferentes partidas para alcanzar un total de puntos que determine al ganador.
- Todos los mensajes mandados al servidor con respecto a la conexión y validación o la creación de canales recibirán una respuesta indicando que ha sido correcto *+OK. Texto informativo* o que ha habido algún error *-ERR. Texto informativo.*
- Debe tenerse presente, que en la implementación no se considerará encriptación de la información, algo que sería necesario en el envío de información segura.

4. Conclusiones

Este trabajo presenta dos recursos didácticos para la adquisión de las competencias en las asignaturas que intentan mostrar los fundamentos de las Redes de Computadores. Entre las principales ventajas que se han encontrado en su puesta en práctica en la asignatura de Redes del Grado en Ingeniería Informática de la Universidad de Córdoba, se encuentran:

- Los estudiantes han adquirido mediante el diseño y desarrollo de las aplicaciones afianzar conceptos esenciales en estos contextos, como son los protocolos, los servicios y las interfaces, que se encuentran definidos en las diferentes capas. El desarrollo de aplicaciones cliente/servidor usando sockets conlleva el diseño de un protocolo consistente en un lenguaje común entre el cliente y el servidor.
- La consecución de una aplicación final, operativa y disponible, que les ha permitido compartirlas con sus compañeros a través de la red, ha sido muy bien aceptada por los estudiantes. El hecho de estar motivados, ha conseguido una mayor implicación de los estudiantes, que en muchos casos, han mejorado la funcionalidad de forma voluntaria.

Referencias

1. A.M. Pisabarro, C.E. Vivaracho. Gamificación en el aula: gincana de programación. Actas de las XXIII Jornadas sobre Enseñanza Universitaria de la Informática, 39-46, 2017.
2. S. Luján, E. Saquete. Mejora en el aprendizaje a través de la combinación de la clase invertida y la gamificación. Actas de las XXIII Jornadas sobre Enseñanza Universitaria de la Informática, 205-212, 2017.
3. Charles Kozierok. 2005. The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference. No Starch Press, San Francisco, CA, USA.
4. Charles Kozierok. 2005. The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference. No Starch Press, San Francisco, CA, USA.
5. Miller, F.P. and Vandome, A.F. and McBrewster, J. 2009. Berkeley Sockets: Computer Network, Internet Socket, Unix Domain Socket, Application Programming Interface, Microsoft Windows, Library (computing), C (programming Language), Inter-process Communication. ISBN 9786130255060, Alphascript Publishing.