ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"COMUNICACIÓN EN TIEMPO REAL ENTRE DISPOSITIVOS USANDO TECNOLOGÍA DE SOCKETS EN UNA SIMULACIÓN DEL JUEGO CUARENTA"

INFORME DE PROYECTO INTEGRADOR

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

GIANNI ANDRÉS CARLO UNDA

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTO

Mis más sinceros agradecimientos a mi familia que siempre estuvieron presente en todo momento, y a mi prometida quien da balance a mi vida.

TRIBUNAL DE EVALUACIÓN

Rafael Bonilla MSc

PROFESOR EVALUADOR

Cristina Abad Ph.D.

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Gianni Carlo Unda

RESUMEN

Este proyecto consiste en una implementación del juego Cuarenta. Se analizaron las soluciones ya existentes en las plataformas móviles Android y iOS. Ninguna aplicación presenta mayor reto en cuanto a jugar contra la computadora, y solo una aplicación permite partidas entre dispositivos pero la comunicación es lenta. El objetivo de este proyecto es mostrar que se puede proveer una experiencia más rápida en la comunicación entre los dispositivos de los jugadores y que la inteligencia artificial presente un mayor desafío para partidas de un solo jugador. Esto se lo logró basando la comunicación entre dispositivos en sockets TCP y la inteligencia artificial en un algoritmo probabilístico Minimax. Se encontró que los sockets TCP son más rápidos que una solución que se basa en tecnología HTTP, y que Minimax provee un buen desafío para un jugador regular.

ÍNDICE GENERAL

CAPÍTULO 1

1. Análisis del problema

Se han analizado las implementaciones existentes del juego de Cuarenta para las plataformas móviles de iOS y Android. Se encontró que las partidas de un solo jugador, los algoritmos de inteligencia artificial que han implementado no presenta mayor reto. En multijugador, la única aplicación que permite partidas multijugador entre dispositivos, la comunicación entre los dispositivos es lenta y merma la experiencia de usuario, esto puede ser ocasionado por el protocolo usado en la implementación del canal de comunicación.

1.1 Causas

La comunicación en tiempo real entre dispositivos en la única aplicación que lo permite se demora.

El algoritmo de inteligencia artificial para la computadora no es el apropiado.

1.2 Efectos

La experiencia de usuario se ve afectada por la comunicación lenta entre dispositivos.

No presenta mayor reto la computadora.

1.3 Soluciones ya existentes

Las aplicaciones ya existentes son:

- “Cuarenta” (Android y iOS)
- “40 caída y limpia” (Android y iOS)
- “Cuarenta Classic” (Android)

Solo “40 caída y limpia” ofrece multijugador tanto para Android como iOS, pero hay mucho por mejorar en la implementación. Existen veces que no logra conectar contra otro dispositivo en busca de partida e inmediatamente empieza una partida contra la computadora.

1.3.1 Cuarenta

Figura 1.3.1.1: Partida de un jugador del App Cuarenta

Se encuentra disponible para Android [1] y iOS [2]. Solo presenta opción de un solo jugador

1.3.2 40 caída y limpia

Figura 1.3.2.1: Pantalla inicial del App 40 caída y limpia

Se encuentra disponible para Android [3] y iOS [4]. Presenta las opciones de un solo jugador y multijugador

1.3.3 Cuarenta Classic

Figura 1.3.3.1: Pantalla del menú de un solo jugador

Se encuentra disponible para Android [5]. Solo presenta opción de un solo jugador

1.4 Reglas del juego

Las reglas del cuarenta son las siguientes:

- Caída: si se juega una carta de igual valor a la última carta que jugó el adversario (otorga 2 puntos).
- Limpia: si con la carta jugada, se capturan todas las cartas de la mesa (otorga 2 puntos).

- Soplo: si el jugador lanza una carta que puede capturar cartas en la mesa pero no las hace, el oponente tiene la opción de capturarlas automáticamente.
- Ronda: si al iniciar una mano se tienen tres cartas con el mismo valor (otorga 2 puntos).
- Convertir cartas capturadas a puntos: al finalizar una ronda completa, se cuentan las cartas capturadas, y si estas superan las 18 cartas, comienza a sumar 6, 7, 8… puntos por cada carta extra capturada.
- Al inicio de cada ronda, siempre se debe cambiar el jugador que empieza, en sentido de las manecillas del reloj.

Condiciones de ganar:

- Un equipo llega a los cuarenta puntos.
- Un jugador obtiene cuatro cartas con el mismo valor.

CAPÍTULO 2

2. Análisis de la propuesta.

Se propone utilizar dos servidores, uno dedicado para guardar los datos de los jugadores y las listas de partidas en espera a que se unan más jugadores. El otro servidor será para nuestro canal de comunicación mediante el cual los jugadores de una partida podrán comunicar las jugadas. Este esquema permite que no seamos limitados por plataformas y da apertura a que podamos implementar el cliente en cualquier plataforma disponible.

2.1 Arquitectura de la solución propuesta

Nuestra solución será de tipo cliente-servidor, para el back-end, proponemos dos bases de datos:

1. Base de datos relacional – La cual está encargada de persistir los datos del jugador (nombre de usuario, id de sesión).
2. Base de datos no relacional – La cual está encargada de la creación y sincronización de las partidas y sus participantes. Contendrá dos listas, una para los juegos en espera de dos jugadores, y la otra para los juegos en espera de cuatro jugadores.

Las bases de datos van a residir en un servidor, el cual expondrá el acceso e interacción con estas bases como un servicio REST.

La sincronización de las partidas multijugadores seguirá el siguiente esquema:

1. A través del servicio REST se consultará a la base de datos no relacional si existen partidas en espera que se registren más jugadores.
2. Si no hay partidas en espera, se lo registra dentro de una partida en espera al jugador en la lista de dos o cuatro jugadores según sea el caso.
3. Si existe una partida en espera, se lo agrega a la lista de participantes de esa partida, caso contrario se crea un nuevo juego en espera dentro de la base de datos no relacional.

4. En caso que el número de jugadores es suficiente para empezar la partida, se elimina la partida de la lista de juegos en la base de datos no relacional, y se envía en la respuesta al último jugador en unirse para que cree un grupo en el servidor de socket con todos los integrantes de la partida.

Una vez creado el grupo, la comunicación entre los jugadores estará dada por el servidor de socket al cual todos los jugadores se conectan con su id de sesión. El servidor de socket se encarga de enviar los mensajes entre los jugadores con la información de las cartas seleccionadas de la mesa y que cartas se van jugando.

Esta arquitectura de servidor (Figura 2.1), permite que nuestra aplicación cliente pueda ser implementada para cualquier dispositivo, habilitando la posibilidad de partidas de multijugadores entre múltiples plataformas.

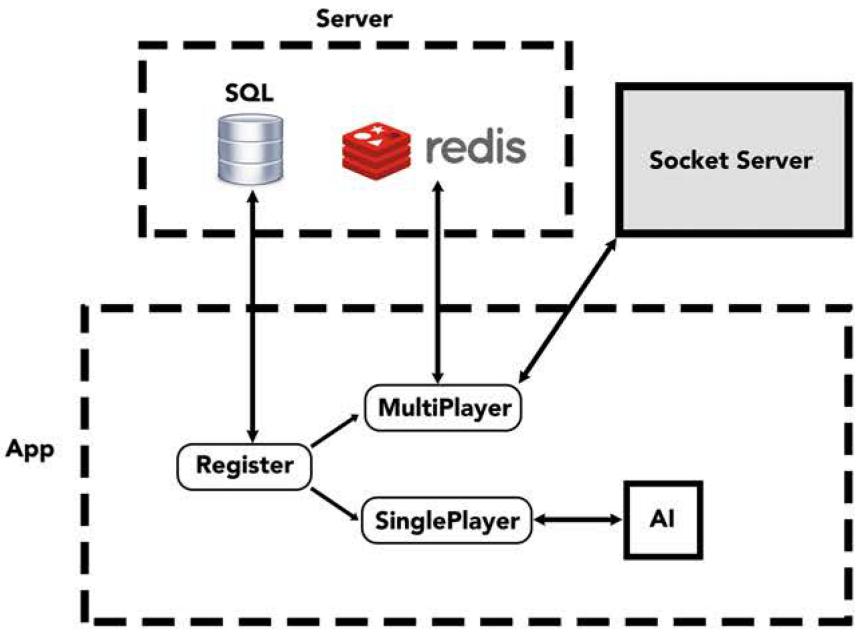

Figura 2.1: Arquitectura del sistema

CAPÍTULO 3

3. Implementación

3.1 Servidor

Para el servidor es necesario instalar MySQL y Redis [6]. MySQL tiene el propósito de guardar la información de los usuarios de la aplicación, mientras que Redis se encarga de llevar el registro de las partidas que aún no han comenzado y sus integrantes.

Para la implementación del back-end, se decidió seguir una arquitectura REST, bajo la cual se exponen las funcionalidades:

- Registrar usuario a la base de datos MySQL
- Buscar una partida en Redis
- Solicitar los participantes de una partida a Redis
- Borrar participantes de una partida en Redis

Las respuestas de requerimientos mal formados se las puede ver en la tabla 1

“error”	**“descriptionerror”**	**Descripción**
100	“No ‘data’ in web request”	El requerimiento no tiene ningún parámetro
102	“No request”	El parámetro “request” está vacío
103	“DB not connected”	No se pudo establecer la conexión con MySQL
104	“Schema not connected”	No se encuentro el esquema de la base de datos
105	“No parameters in request.”	El parámetro “request” contiene un valor no válido.

Tabla 1: Respuestas de requerimientos al servidor mal formados

El esquema de los sets de Redis se pueden observar en la tabla 2.

Key	Set	Descripción
“games_2players”	Identificador del juego	Este set contiene una lista de los identificadores de los juegos de 2 jugadores que todavía no empiezan
“game_4players”	Identificador del juego	Este set contiene una lista de los identificadores de los juegos de 4 jugadores que todavía no empiezan
Identificador del juego	Indentificador de participante	Este set contiene una lista de los identificadores de los participantes de la partida

Tabla 2: Esquema de Redis

Tanto los requerimientos como las respuestas del servidor se sirven en formato JSON. Las respuestas tienen los campos que se pueden observar en la tabla 3.

“error”	“descriptionerror”	“data“
Número entero	Descripción del error	Diccionario que contiene la data requerida

Tabla 2: Formato de las respuestas del servidor REST

3.1.1 Registrar usuario

Para registrar un usuario, el servidor primero consulta la base de datos MySQL para verificar si dicho nombre de usuario ya está en uso, si no está en uso, el usuario es registrado, caso contrario se retorna un error a la aplicación.

El requerimiento debe ser de tipo POST, con los siguientes parámetros:

- “request” = “register”
- “nickname” = nombre de usuario
- “sessionid” = identificador único del usuario

La respuesta retorna los parámetros en la tabla 4.

“error”	**“descriptionerror”**	**“data“**
0	No hay error	• “sessionid” = identificador único del usuario • “nickname” = nombre de usuario
3	“Invalid query request when inserting user”	No hay data
5	“This nickname is already in use”	No hay data
10	“Invalid query request”	

Tabla 4: Respuesta del servidor para registro de usuario

3.1.2 Buscar una partida

Para buscar una partida, el servidor consulta en Redis con el identificador del usuario para buscar juegos con cupo libre para el participante; en caso que encuentre un juego disponible, el servidor lo registra al usuario y retorna el identificador del juego junto a los integrantes. Caso contrario, el servidor crea un juego con el identificador del jugador y retorna a la aplicación este identificador y los participantes.

El requerimiento debe ser de tipo POST, con los siguientes parámetros:

- “request” = “search_game”
- “player_id” = identificador único del usuario

- “max_players” = número máximo de participantes del juego

La respuesta retorna los parámetros que se observan en la tabla 5.

“error”	**“all_participants”**	**“host_player “**
0	Lista de los identificadores de los participantes	Identificador del anfitrión del partida

Tabla 5: Respuesta del servidor para búsqueda de una partida

3.1.3 Eliminar una partida

El servidor borra una partida en dos ocasiones:

- El juego va a comenzar
- El anfitrión cancela la busqueda de una partida

Se elimina el juego de redis cuando va a comenzar para evitar que nuevos jugadores intenten unirse a dicha partida.

El requerimiento debe ser de tipo POST, con los siguientes parámetros:

- “request” = “delete_game”
- “game_identifier” = identificador único de la partida
- “max_players” = número máximo de participantes del juego

La respuesta retorna los parámetros en la tabla 6.

“error”	**“data”**
0	OK

Tabla 6: Respuesta del servidor para eliminar una partida

3.1.4 Salir de una partida

El servidor elimina al participante en Redis dentro del set de participantes de un juego.

El requerimiento debe ser de tipo POST, con los siguientes parámetros:

- “request” = “leave_game”
- “player_id” = identificador único del participante
- “max_players” = número máximo de participantes del juego

La respuesta retorna los parámetros en la tabla 7.

“error”	**“data”**
0	OK

Tabla 7: Respuesta del servidor para eliminar un participante de una partida

3.1.5 Solicitar los participantes de una partida

El servidor consulta en Redis el set de integrantes para el identificador de la partida.

El requerimiento debe ser de tipo POST, con los siguientes parámetros:

- “request” = “game_participants”
- “game_id” = identificador único del juego

La respuesta retorna los parámetros que se observan en la tabla 8.

“error”	**“all_participants”**	**“host_player “**
0	Lista de los identificadores de los participantes	Identificador del anfitrión del partida

Tabla 8: Respuesta del servidor para consulta de los participantes de una partida

3.2 Aplicación

La aplicación se desarrolló para la plataforma iOS, diseñado para dispositivos iPhone con la versión mínima de iOS 8.

3.2.1 Registro de Usuario

El usuario es presentado con la pantalla que se puede ver en la figura 3.2.1.1. Al ingresar un nickname y presionar “Join”, se realiza el requerimiento de registro de usuario al servidor.

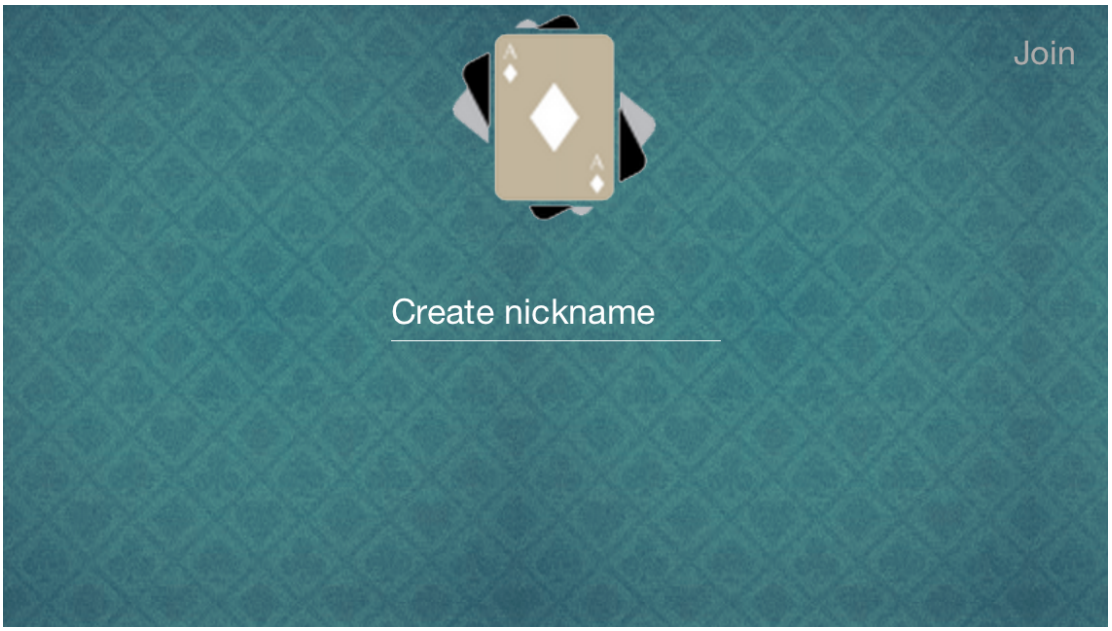

Figura 3.2.1.1: Pantalla de registro de usuario

En el caso que el nombre ya esté tomado, se le da retroalimentación al usuario mediante el celular vibrando en conjunto al nombre ingresado, y además se le muestra un mensaje para que ingrese otro nombre.

Una vez que se escoge un nombre válido, se manda un requerimiento al servidor de socket para que retorne un id único que es el que el jugador utilizará siempre para la comunicación en tiempo real.

Toda la información de usuario es guardada en una base local que provee Realm [7]. Cada vez que se inicia la aplicación, esta información es cargada a memoria desde la base local, esto nos facilita ya no tener que mostrar el registro al usuario cada vez que se inicie la aplicación de nuevo.

La siguiente pantalla que se le presenta al usuario en la Figura 3.2.1.2, es la del menú principal, en la cual podrá escoger si se desea jugar una partida local contra la computadora, o una partida multijugador

Figura 3.2.1.2: Pantalla de Menú Principal

3.2.2 Multijugador

Para el caso de multijugador, se le presenta una pantalla en donde tiene la opción de buscar una partida de 2 o de 4 jugadores (Figura 3.2.2.1).

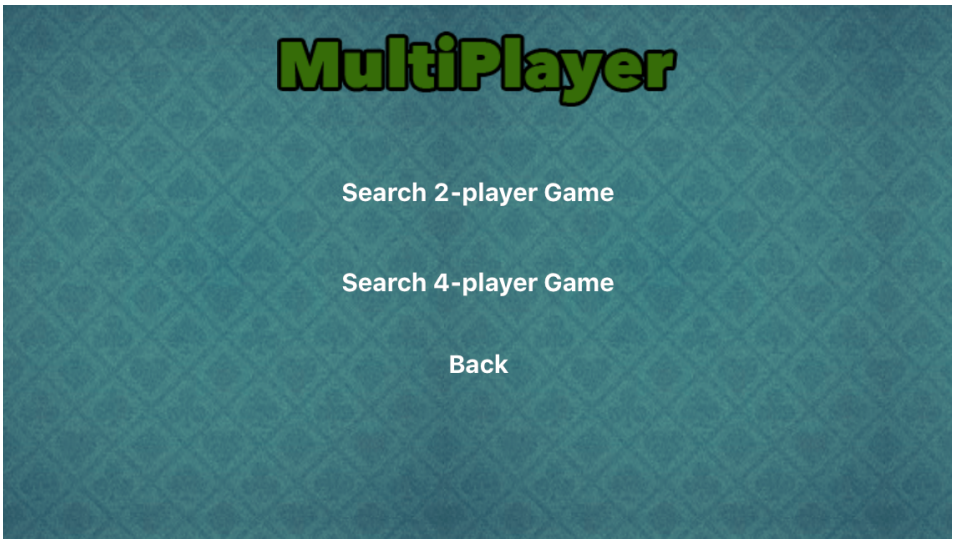

Figura 3.2.2.1: Pantalla de MultiPlayer

Al seleccionar cualquiera de estos dos opciones de 2 o 4 jugadores, se le muestra la Figura 3.2.2.2 al usuario, mientras que se envía un requerimiento al servidor para buscar un juego disponible.

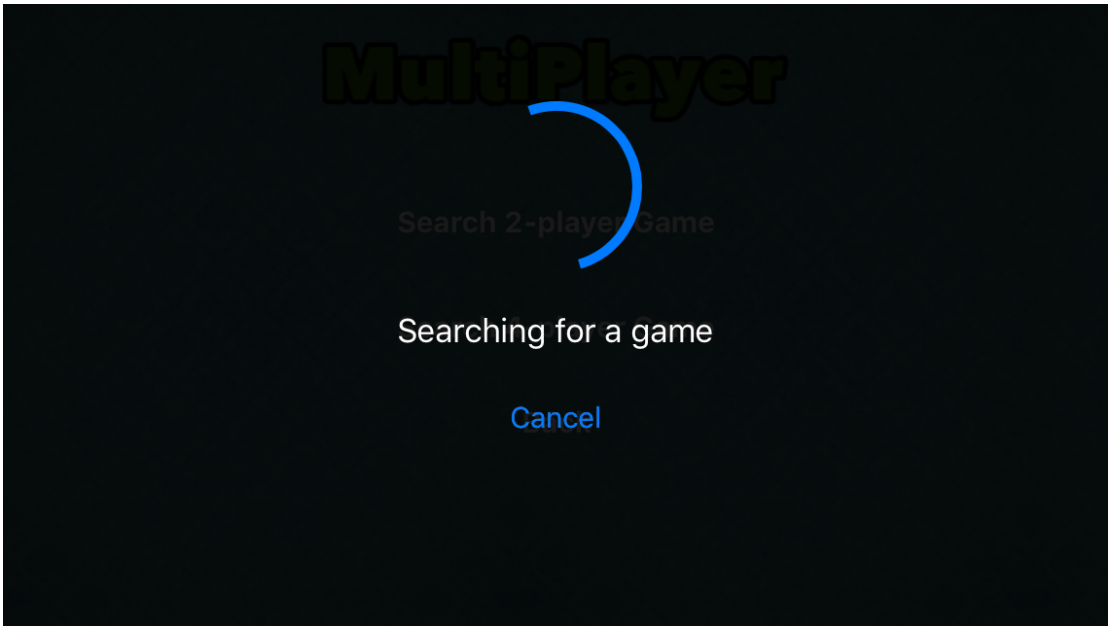

Figura 3.2.2.2: Pantalla de buscando un juego

Una vez que el servidor responda, se guarda en memoria los identificadores de los participantes y del anfitrión del juego, y se notifica a todos a través del socket, que ha habido un cambio en los integrantes del juego. Los usuarios que reciban esta notificación realizarán un requerimiento al servidor para actualizar los participantes del juego.

Una vez que se haya llegado al máximo de participantes para el juego, el anfitrión crea un grupo con los identificadores de los participantes en el servidor de socket, esto se realiza para facilitar la comunicación entre los participantes. Una vez creado el grupo, el anfitrión les notifica a los otros participantes, y les envía el mazo de cartas para que todos empiezen con el mismo estado de juego.

Tanto el mazo de cartas y el orden de jugadores está sincronizado, esto facilita que para cada jugada, solo sea necesario enviar, a través del socket, la carta jugada junto al arreglo de cartas seleccionadas de la mesa.

Una vez ya dentro del tablero de juego (Figura 3.2.2.3), se aplican todas las reglas conocidas del juego Cuarenta.

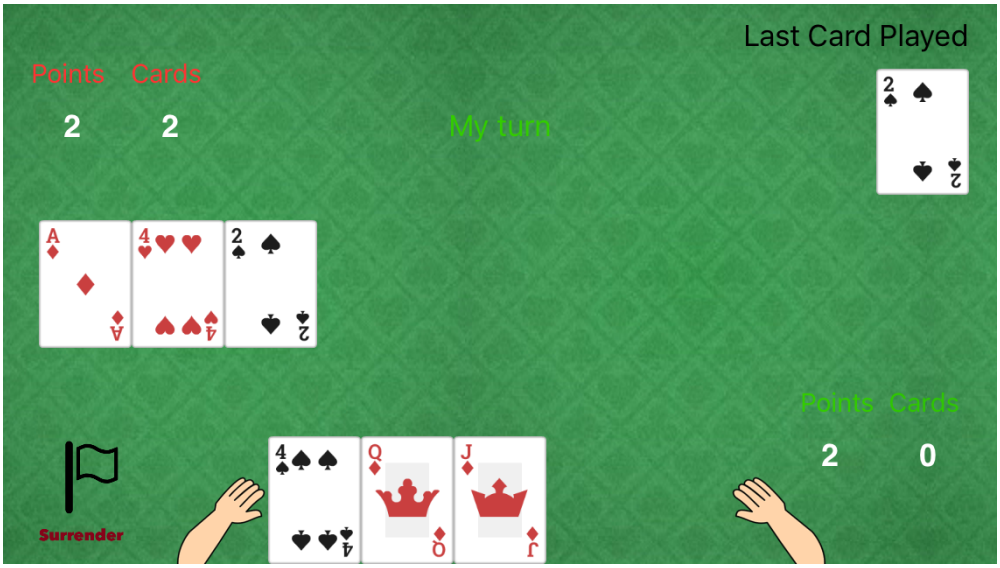

Figura 3.2.2.3: Pantalla del tablero de juego

3.2.3 Un solo jugador

Para el caso de un solo jugador, se muestra una pantalla similar a la del multijugador, la cual se puede ver en la Figura 3.2.3.1.

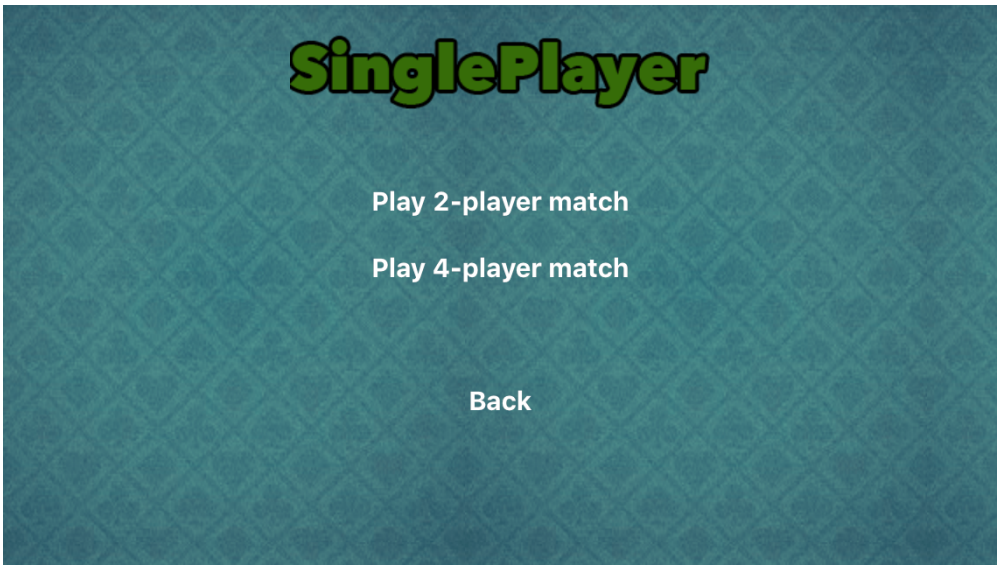

Figura 3.2.3.1: Pantalla del menú de un solo jugador

La inteligencia artificial fue basada en un algoritmo llamado MiniMax [8], y primero debemos saber cuales son las posibles acciones que pueden haber en un juego:

- Capturar una carta que tenga el mismo valor que mi carta
- Capturar varias cartas que sumando den el mismo valor que mi carta
- Luego de realizar una de las anteriores capturas, seguir capturando las cartas por secuencia de valores.
- Jugar la carta a la mesa

Entonces se declaró un protocolo cuyos métodos consisten en:

- Dado un set de cartas en la mesa y un set de cartas en la mano, conseguir cual es la mejor jugada que se puede hacer
- Dado el conocimiento de todas las cartas que se han jugado, determinar cual es la mejor carta a jugar

MiniMax se lo implementa para el primer método, el cual retornará la mejor carta a jugar a ser jugada, en conjunto con las cartas seleccionadas de la mesa para capturar. Las reglas de decisión de Minimax es la siguiente:

1. La “caída” siempre va a ser la mejor jugada por lo que suma dos puntos directos.
2. Capturar cartas por suma es la siguiente, ya que captura más cantidad de cartas que la captura por el mismo valor, y da la opción de seguir capturando por secuencia.
3. Captura por el mismo valor, el cual sigue siendo una mejor opción que simplemente jugar una carta a la mesa
4. La captura por secuencia siempre se realiza en caso que sea posible

En caso que no se pueda realizar ninguna captura, se utilizará el segundo método, el cual retornará la carta con la menor probabilidad de que el oponente pueda realizar una caida.

3.2.4 Interacción

Para jugar una carta, simplemente se la toca, pero en el caso que se quieran capturar cartas, es necesario seleccionarlas antes de jugar la carta, caso contrario se activa la regla de “soplar” de cuarenta, la cual le otorga las cartas capturadas al oponente.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. La implementación del algoritmo de Minimax presenta un oponente de nivel medio.
2. La comunicación a través de sockets es rápido y confiable.
3. No es intuitivo la forma de seleccionar las cartas y jugarlas.

Recomendaciones

1. Se debe expandir la implementación del algoritmo de inteligencia artificial para que considere tres turnos a futuro.
2. Mejorar la interacción de usuario con respecto al juego para que sea más natural.
3. Guardar el estado de una partida, para restaurarla en el evento de una desconexión a la partida,

BIBLIOGRAFÍA

[1] “Cuarenta”. Disponible en Android: https://play.google.com/store/apps/details?id=com.cuarenta_game , [septiembre 25, 2015].

[2] “Cuarenta”. Disponible en iOS: https://itunes.apple.com/us/app/cuarenta/id1018948625?mt=8 , [septiembre 25, 2015].

[3] “40 Caida y limpia”. Disponible en Android: https://play.google.com/store/apps/details?id=com.iglooJuan.ficto40 , [septiembre 25, 2015].

[4] “40 Caida y limpia”. Disponible en iOS: https://itunes.apple.com/us/app/40-caida-y-limpia/id935492399?mt=8 , [septiembre 25, 2015].

[5] “Cuarenta Classic”. Disponible en Android: https://play.google.com/store/apps/details?id=com.cuarenta_game_lite , [septiembre 25, 2015].

[6] Redis Technology, Disponible en: http://redis.io/ , [junio 10, 2015].

[7] Realm Technology, Disponible en: http://realm.io/ , [junio 23, 2015].

[8] Millington, (2009, agosto). Artificial Intelligence For Games, 2nd Ed. pp. 671-686.