 Proyecto CUPi2	ISIS-1205 Algorítmica y Programación Descripción
Ejercicio:	n12_batallaPokemon
Autor:	Equipo CUPi2
Semestre:	2016-2

Enunciado

Se quiere construir una aplicación que permita jugar Batalla Pokémon desde diferentes computadores. Este es un juego de batallas Pokémon de dos jugadores, cuyo objetivo es debilitar todos los pokémon del oponente.

Al iniciar el juego cada jugador se identifica con su alias y su nombre, se conecta al servidor para comenzar el juego y selecciona 4 pokémon. El orden en el que el jugador selecciona los pokémon al empezar el juego es el orden en el que serán utilizados durante el juego.

De cada pokémon se conoce:

- **Tipo.** Permite clasificar a los pokémon. Los posibles tipos son: Fuego, Agua, Volador, Eléctrico, Planta o Roca.
- **Nombre.** Un jugador no puede tener dos pokémon con el mismo nombre.
- **Salud.** Indica la salud que tiene el pokémon y disminuye según los ataques realizados por el oponente. La salud de todos los pokémon inicia en 30 puntos.
- **Imagen.**

El jugador con el primer turno es escogido aleatoriamente.

Después de asignar el turno comienza el juego. En cada turno el jugador puede realizar un ataque con el pokémon actual o cambiar el pokémon actual, cediendo el turno.

Un ataque hace daño al pokémon del oponente, descontándole puntos de salud. Un pokémon se considera debilitado cuando su salud es igual a cero. Un jugador gana la batalla una vez debilite los cuatro pokémon de su oponente.

El daño del ataque se define teniendo en cuenta el tipo de cada pokémon (tanto del jugador como el del oponente) y un valor aleatorio, aplicando las reglas de la Tabla 1, donde D es un valor aleatorio entre 1 y 5.

TIPO DE POKÉMON QUE ATACA	TIPO DE POKÉMON QUE RECIBE EL ATAQUE						
		FUEGO	AGUA	VOLADOR	ELÉCTRICO	PLANTA	ROCA
	FUEGO	½ D	½ D	1D	1D	2D	½ D
	AGUA	2 D	½ D	1D	1D	½ D	2D
	VOLADOR	1D	1D	1D	½ D	2D	½ D
	ELÉCTRICO	1D	2D	2D	½ D	½ D	2D
	PLANTA	½ D	2D	½ D	1D	½ D	2D
	ROCA	2D	1D	2D	1D	1D	1D

Tabla 1. Puntos perdidos por ataque según los tipos de pokémon.

Por ejemplo, si un pokémon de tipo agua ataca a uno de tipo fuego y el valor aleatorio D es 3, el pokémon de tipo fuego perderá 6 puntos (2×3) de salud. Una vez debilitado, lo sustituye el siguiente pokémon disponible, en el orden que seleccionó el jugador inicialmente.

Como un servicio adicional, el programa debe almacenar información sobre la cantidad de batallas perdidas y ganadas por cada uno de los jugadores.

El programa distribuido debe constar de dos partes: un programa servidor, encargado de mantener la información estadística del juego y permitir a los jugadores encontrarse para una batalla; y un programa cliente, a través del cual un usuario puede jugar Batalla Pokémon.

El programa cliente debe ofrecer las siguientes funcionalidades:

1. Cargar la información de los pokémon disponibles de un archivo de texto.
2. Conectarse al servidor. Para esto el usuario tiene dos alternativas:
 - a. **Registrarse:** La primera vez que un usuario ingresa debe registrarse para poder jugar. Para esto debe suministrar todos sus datos: Nombres, apellidos, alias (único), contraseña, dirección IP del servidor al cual se quiere conectar y el puerto por el cual dicho servidor se encuentra esperando conexiones. Adicionalmente, debe seleccionar el avatar (femenino o masculino) que desea usar en el juego.
 - b. **Iniciar sesión:** Si el usuario ya se encuentra registrado, debe iniciar sesión. Para esto debe suministrar su alias, su contraseña, la dirección IP del servidor y el puerto. Adicionalmente, debe seleccionar el avatar (femenino o masculino) que desea usar en el juego.
3. Seleccionar los pokémon. Una vez se establece la conexión con el oponente, cada usuario debe seleccionar los cuatro pokémon con los que jugará la batalla.
4. Visualizar información de la batalla. El jugador podrá visualizar la siguiente información:
 - a. **Información del oponente:** Nombre, cantidad de victorias y cantidad de derrotas.
 - b. **Información de los pokémon:** pokémon seleccionados por el jugador y su estado actual (debilitado o con salud), y el estado actual de los pokémon seleccionados por el usuario.

- c. **Información del juego:** Avatar propio y del oponente, pokémon actual propio y pokémon actual del oponente, y el estado de salud de los dos pokémon actuales.
 - d. **Mensajes:** Listado de mensajes de las jugadas realizadas y ataques recibidos.
 - e. **Estado de la batalla:** Este puede ser: en espera del oponente, en espera de un ataque, realizando un ataque o batalla terminada
- 5. Realizar un ataque. Realiza un ataque con el pokémon seleccionado actualmente. Una vez realizado, el turno pasa a ser del oponente.
 - 6. Recibir un ataque. Se calcula el daño recibido según el tipo de pokémon que realizó el ataque y se informa al oponente cuánto daño causó.
 - 7. Cambiar de pokémon. Para esto el usuario debe seleccionar el pokémon por el cual desea cambiar el actual. El programa debe informar a su oponente el cambio realizado. Al cambiar de pokémon, el jugador pierde su turno.

El programa servidor, por su lado, debe esperar a que los jugadores se vayan conectando. Cada vez que se conecten dos jugadores, inicia una batalla. Esto implica que el servidor debe:

- 1. Verificar la información de cada jugador cuando se conecte.
 - a. Si el jugador inicia sesión, verifica su alias y contraseña. El alias debe estar registrado y la contraseña dada debe corresponder a la que se encuentra registrada con ese, si esto no ocurre, se le notifica al usuario con un mensaje de error y no se conecta.
 - b. Si el jugador se registra, verifica el alias dado. No debe existir un jugador con el mismo alias, si esto ocurre, se le notifica al usuario con un mensaje de error y no se registra ni se conecta.
- 2. Enviar a cada jugador la información de su oponente, una vez se establece una batalla.

Una vez establecida la batalla, el servidor es el responsable de la administración del juego. Esto implica que el servidor debe:

- 1. Determinar aleatoriamente los turnos e informarle a los jugadores su turno.
- 2. Notificar el resultado de un ataque a los dos jugadores.

Una vez finalizada la batalla (cuando un jugador derrote los 4 pokémon de su oponente), el servidor debe cerrarla. Esto implica que el servidor debe:

- 1. Registrar un juego ganado al jugador victorioso.
- 2. Registrar un juego perdido al jugador derrotado.
- 3. Notificar el fin del juego a los dos jugadores.
- 4. Notificar a los dos jugadores el alias del jugador victorioso.

Finalmente, el servidor debe ofrecer las siguientes opciones al usuario:

- 1. Mostrar las batallas que se encuentren en curso.
- 2. Mostrar las estadísticas históricas del juego. Se debe mostrar el alias de cada jugador que haya participado en una batalla, la cantidad de victorias y de derrotas, y su porcentaje de efectividad. Este último se define de la siguiente manera:

$$\% \text{ Efectividad} = \text{Batallas ganadas} * 100 / \text{Total de batallas}$$

La información histórica de las batallas debe ser persistente y se debe almacenar en una base de datos, de manera que cada vez que se ejecute de nuevo el servidor, las estadísticas sean presentadas al usuario.

Interfaz Cliente

Diálogo de inicio. El usuario puede elegir si desea iniciar sesión o registrarse (Figura 1).



Figura 1. Diálogo de inicio

Si el usuario desea registrarse, se desplegará el diálogo que se muestra en la Figura 2.



Figura 2. Diálogo de registro.

Por otro lado, si el usuario desea iniciar sesión, se muestra el siguiente diálogo:



Figura 3. Diálogo de inicio de sesión.

Una vez se realiza la conexión de dos jugadores, se habilita el diálogo para seleccionar los pokémon (Figura 4).

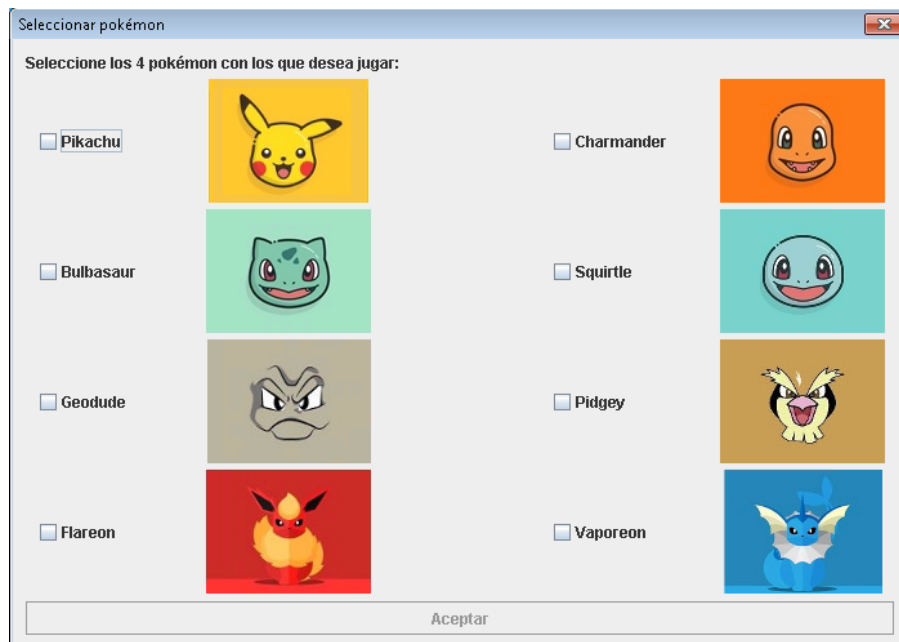


Figura 4. Diálogo para que el jugador seleccione sus pokémon.

Finalmente, en la Figura 5 se muestra la interfaz principal al momento de realizar una batalla.

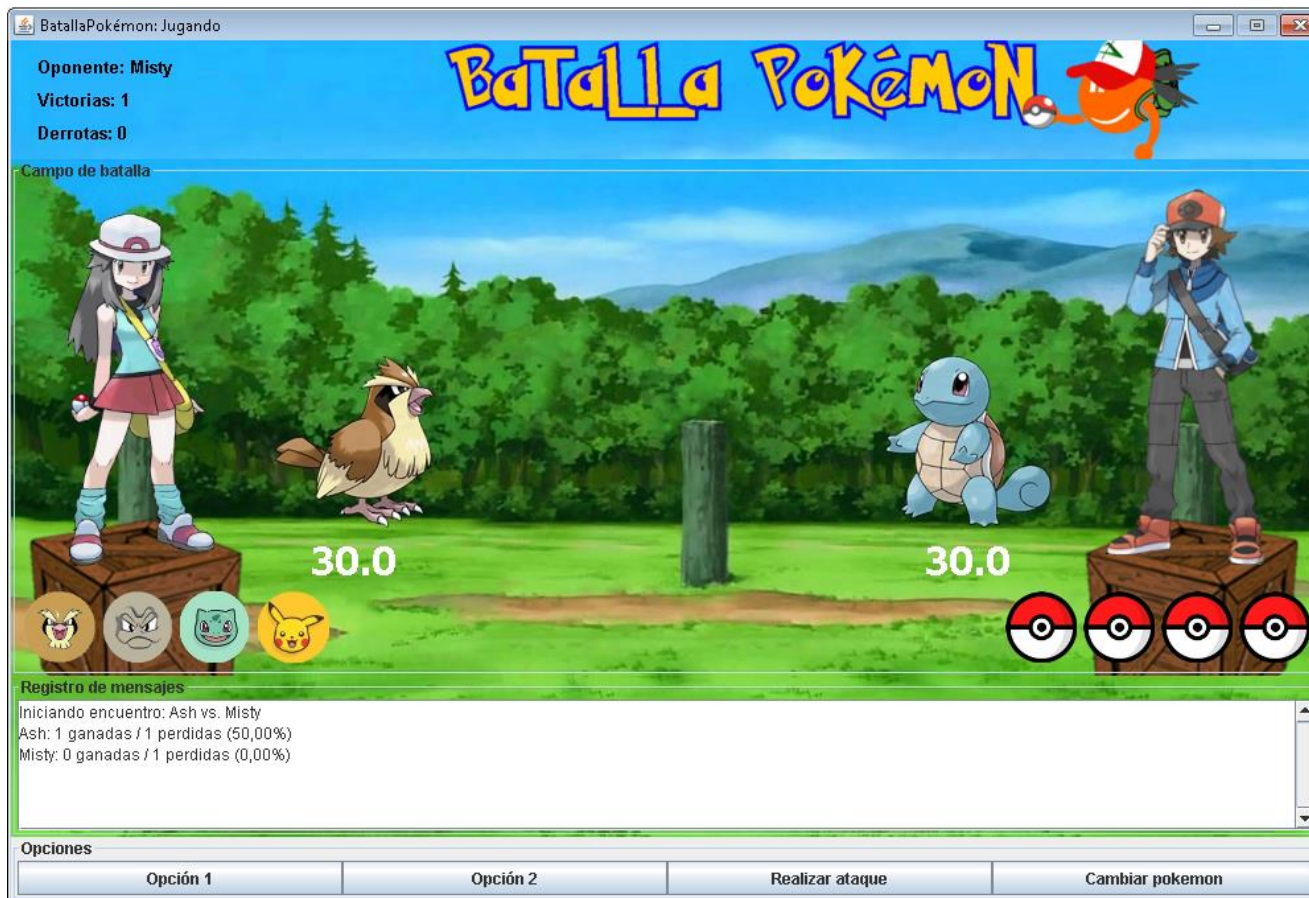


Figura 5. Interfaz principal en batalla.

Interfaz Servidor

La interfaz del servidor (Figura 6) permite refrescar la lista de batallas actuales y la lista de jugadores que han realizado una batalla.

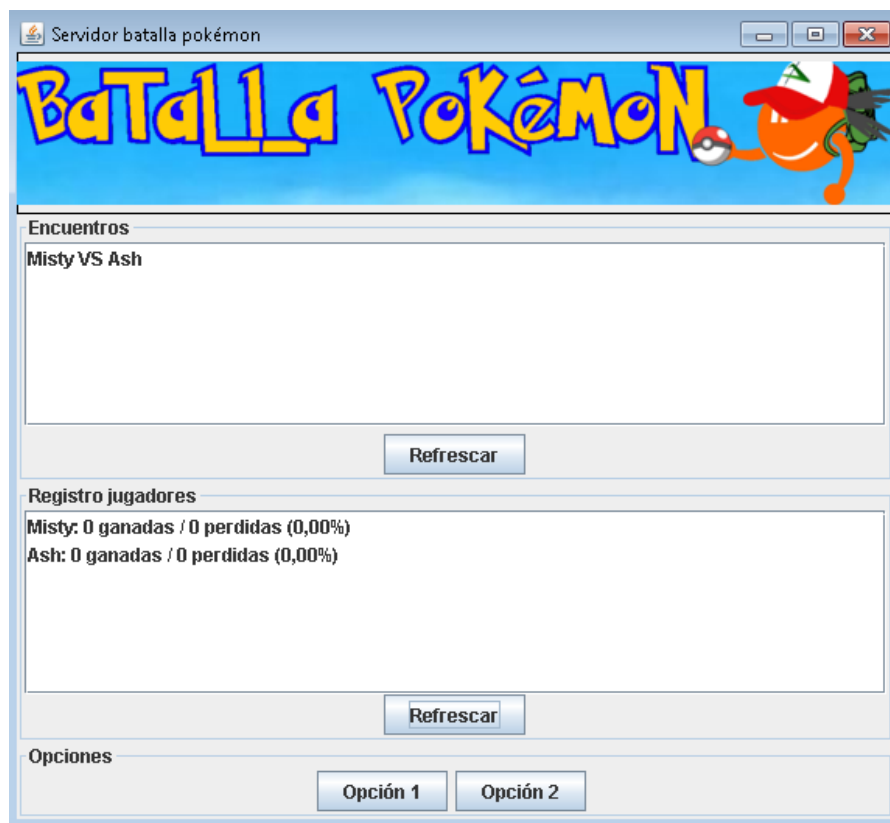


Figura 6. Interfaz del servidor.

Persistencia

Para el desarrollo de la aplicación Batalla Pokémon, se utilizará como motor de base de datos el sistema (*open source*) llamado Derby, desarrollado en el proyecto Apache Derby.

Las ventajas de utilizar un motor de base de datos Derby son:

- Está desarrollado completamente en Java y tiene un sistema de persistencia basado en archivos.
- Permite crear un servidor Derby independiente de la aplicación y establecer la comunicación con esta a través de sockets y desde diferentes máquinas.
- Es una base de datos embebida, es decir, que se puede utilizar como si fuera parte de la aplicación.

Para la aplicación se usará la base de datos embebida que consta de una única tabla para almacenar la información de los jugadores desde la primera vez que el servidor es ejecutado. La tabla incluye los siguientes campos: alias del jugador, nombre del jugador, apellidos del jugador, contraseña del jugador, victorias y derrotas. La llave primaria de la tabla es el alias del jugador. Esto implica que no pueden existir dos jugadores con el mismo alias).

En la Tabla 2 se muestra el diseño de la tabla de la base de datos con sus campos y llave primaria.

Tabla	Campos	Tipo	Llave primaria
jugadores	alias	varchar(32)	alias
	nombre	varchar(32)	
	apellidos	varchar(50)	
	contrasenia	varchar(12)	
	victorias	int	
	derrotas	int	

Tabla 2. Diseño de la tabla para la base de datos.

Protocolo de comunicación

A continuación, se presenta el protocolo de comunicación que establece cuáles mensajes (y en qué orden) se deberán enviar para realizar cada una de las tareas del sistema. Es responsabilidad tanto del servidor como de los clientes ser capaz de interpretar este protocolo y realizar las tareas necesarias.

En términos generales, los mensajes que se envían tienen el siguiente formato:

<COMANDO>;;<PARÁMETRO1>:::<PARÁMETRO2>::: ... :::<PARÁMETROn>

En dónde:

- <COMANDO>: Indica la solicitud del usuario al servidor o la respuesta del servidor al usuario.
- <PARAMETRO>: Indica la información necesaria para resolver la petición.
- Para diferenciar el comando de los parámetros, se utiliza el separador “;;;” (3 “punto y coma”).
- Para diferenciar los parámetros, se utiliza el separador “:::” (3 “dos puntos”).

Protocolo para iniciar sesión:

Caso 1: El jugador inicia sesión correctamente en el servidor.

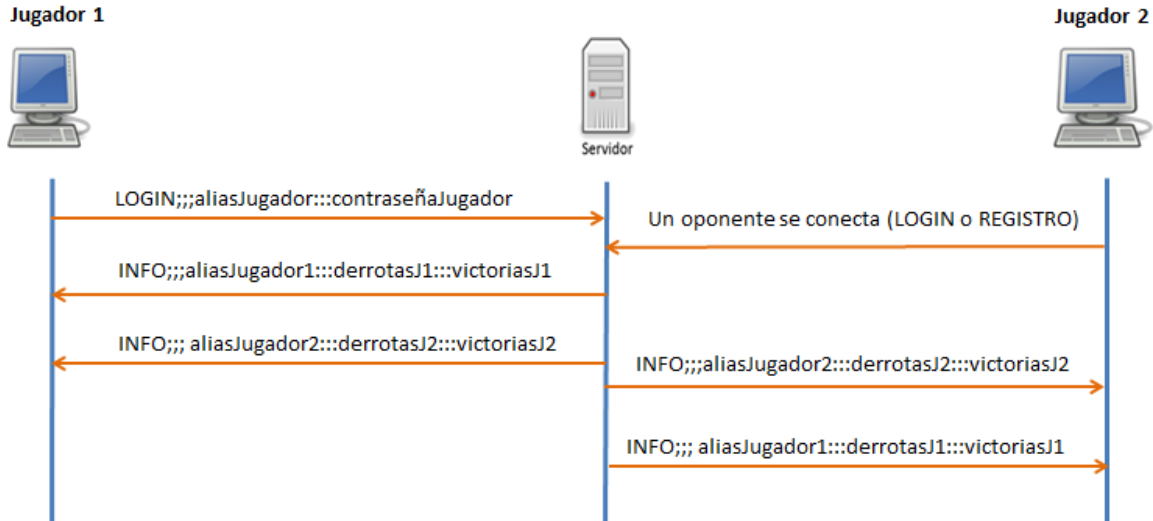


Figura 7. Protocolo de inicio de sesión.

En la Figura 7 se aprecia el protocolo de inicio de sesión que consiste en que el programa cliente envía un mensaje al servidor con el comando LOGIN, seguido del alias y la contraseña del jugador que desea iniciar sesión. El servidor verifica la existencia del alias y correspondencia de la contraseña en la base de datos. Posteriormente, una vez se inicie el encuentro con otro jugador, el servidor envía un mensaje con el alias y las estadísticas (victorias y derrotas) del jugador y un mensaje con la información del oponente, con el comando INFO.

La sintaxis de cada mensaje es explicada en la **Tabla 3** y **Tabla 4**:

Comando cliente	Parámetros	Descripción
LOGIN	aliasJugador	Representa el alias del jugador que se va a conectar al servidor.
	contraseñaJugador	Representa la contraseña del jugador que se va a conectar al servidor.

Tabla 3. Sintaxis del mensaje del cliente para el protocolo de inicio de sesión.

Comando servidor	Parámetros	Descripción
INFO	aliasJugador	Representa el alias del jugador.
	derrotas	Representa el número de derrotas del jugador.
	victorias	Representa el número de victorias del jugador.

Tabla 4. Sintaxis del mensaje del servidor cuando existe un oponente.

A continuación, un ejemplo concreto:

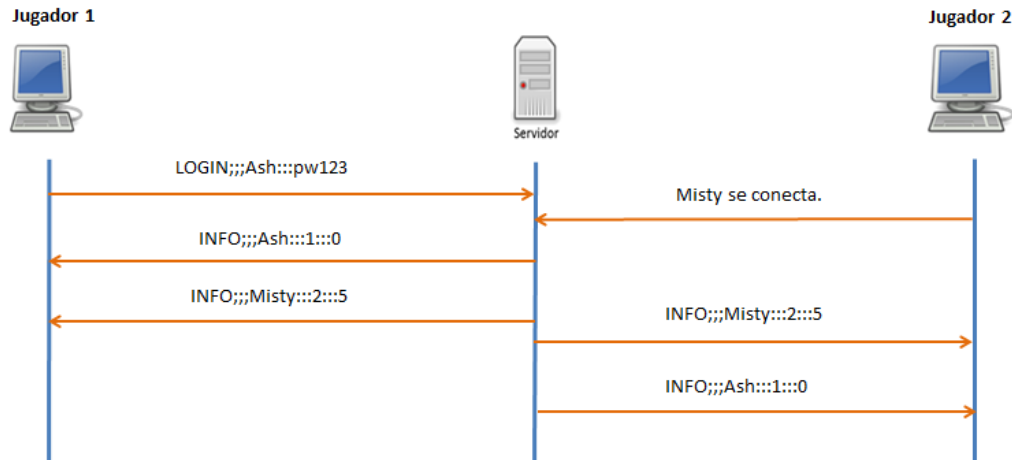


Figura 8. Ejemplo de protocolo de inicio de sesión.

En la Figura 8 se ilustra el escenario en el que el jugador con el alias “Ash” desea iniciar sesión, por lo tanto, el programa envía el mensaje con comando LOGIN, el alias “Ash” y la contraseña “pw123”. El servidor encuentra en su base de datos un jugador con el alias y contraseña dada. Una vez se inicie el encuentro con otro jugador, envía un mensaje con comando INFO indicando la información del jugador y otro mensaje con comando INFO indicando la información del oponente: alias “Misty”, derrotas “2” y victorias “5”.

Caso 2: Un jugador con un alias que no está registrado trata de iniciar sesión:



Figura 9. Protocolo de inicio de sesión cuando ocurre un error.

En la Figura 9 se aprecia el protocolo de inicio de sesión en presencia de un error. Ocurre cuando el programa cliente envía un mensaje al servidor con el comando LOGIN, seguido del alias y la contraseña del jugador que

desea iniciar sesión. El servidor no encuentra en su base de datos un jugador con el alias dado, por lo tanto le envía al programa cliente un mensaje con el comando ERROR y el mensaje de error.

La sintaxis de cada mensaje es explicada en la **Tabla 5** y **Tabla 6**:

Comando cliente	Parámetros	Descripción
LOGIN	aliasJugador	Representa el alias del jugador que se va a conectar al servidor.
	contraseñaJugador	Representa la contraseña del jugador que se va a conectar al servidor.

Tabla 5. Sintaxis del mensaje del cliente para el protocolo de inicio de sesión.

Comando servidor	Parámetros	Descripción
ERROR	mensajeDeError	Representa el mensaje de error que se desea informar al usuario.

Tabla 6. Sintaxis del mensaje del servidor para el protocolo de inicio de sesión en caso de error.

A continuación, un ejemplo concreto:



Figura 10. Ejemplo del protocolo de inicio de sesión cuando ocurre un error.

La Figura 10 muestra el siguiente ejemplo: el jugador envía una petición de inicio de sesión con el comando LOGIN y el alias “Ash12”, contraseña “pw123”. El servidor no encuentra en su base de datos un jugador con el alias dado, por lo tanto envía el mensaje de error con comando: ERROR y el mensaje “El jugador no está registrado”.

Caso 3: Un jugador se conecta al servidor para iniciar sesión y la contraseña no coincide con el alias ingresado.

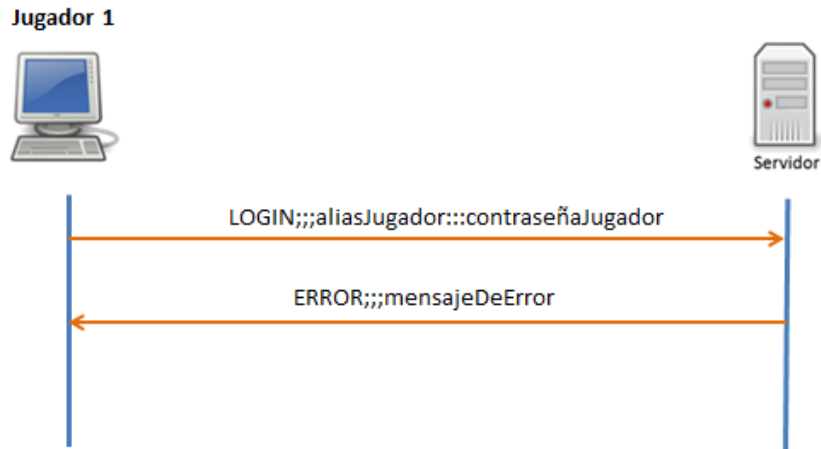


Figura 11. Protocolo de inicio de sesión cuando ocurre un error.

En la Figura 11 se aprecia el protocolo de inicio de sesión en presencia de un error. Ocurre cuando el programa cliente envía un mensaje al servidor con el comando LOGIN, seguido del alias y la contraseña del jugador que desea iniciar sesión. El servidor encuentra en su base de datos un jugador con el alias dado pero la contraseña no coincide, por lo tanto, le envía al programa cliente un mensaje con el comando ERROR y el mensaje de error.

La sintaxis de cada mensaje es explicada en la **Tabla 7** y en la **Tabla 8**:

Comando cliente	Parámetros	Descripción
LOGIN	aliasJugador	Representa el alias del jugador que se va a conectar al servidor.
	contraseñaJugador	Representa la contraseña del jugador que se va a conectar al servidor.

Tabla 7. Sintaxis del mensaje del cliente para el protocolo de inicio de sesión.

Comando servidor	Parámetros	Descripción
ERROR	mensajeDeError	Representa el mensaje de error que se requiere informar al usuario.

Tabla 8. Sintaxis del mensaje del servidor para el protocolo de inicio de sesión en caso de error.

A continuación, un ejemplo concreto:



Figura 12. Ejemplo del protocolo de inicio de sesión cuando ocurre un error.

La Figura 10 muestra el siguiente ejemplo: el jugador envía una petición de inicio de sesión con el comando LOGIN y el alias “Ash”, contraseña “pw000”. El servidor encuentra un jugador con el alias dado, pero con una contraseña diferente, por lo tanto, envía el mensaje de error con comando: ERROR y el mensaje “La contraseña que ingresó no corresponde al usuario con ese alias.”

Protocolo para registrar un jugador:

Caso 1: Se registra un jugador que no se encuentra en la base de datos.

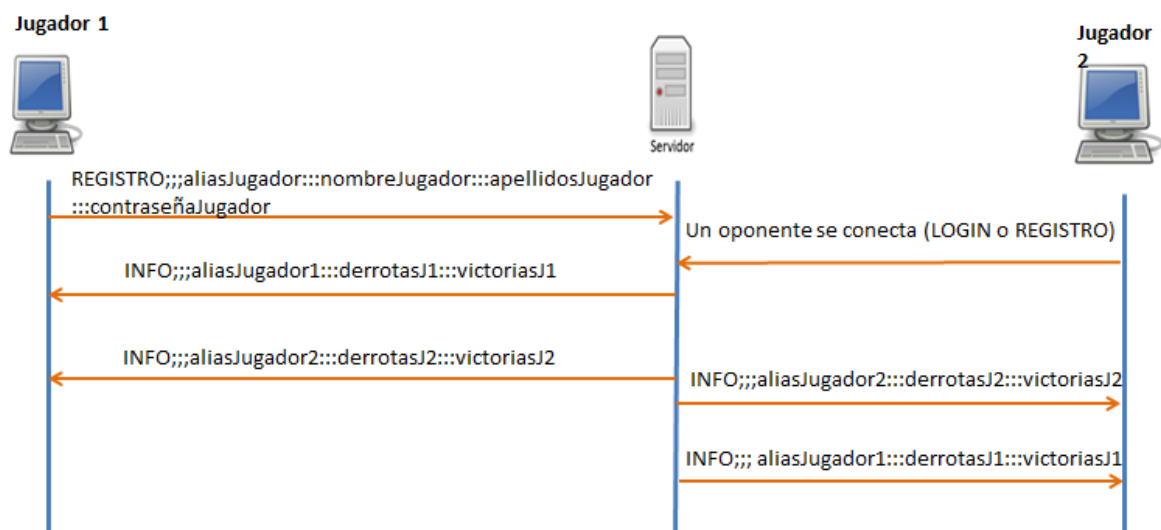


Figura 13. Protocolo de registro de un jugador.

En la Figura 13 se aprecia el protocolo de registro que consiste en que el programa cliente envía un mensaje al servidor con el comando REGISTRO, seguido de los datos del jugador que desea registrarse: alias, nombre, apellidos y la contraseña. El servidor verifica que no exista un jugador con el mismo alias. Posteriormente, una vez se inicie el encuentro con otro jugador, envía un mensaje con el alias y las estadísticas (victorias y derrotas) del jugador y otro mensaje con la información del oponente, con el comando INFO.

La sintaxis de cada mensaje es explicada en la **Tabla 9** y **Tabla 10**:

Comando cliente	Parámetros	Descripción
REGISTRO	aliasJugador	Representa el alias del jugador que desea registrarse.
	nombreJugador	Representa el nombre del jugador que desea registrarse.
	apellidosJugador	Representa los apellidos del jugador que desea registrarse.
	contraseñaJugador	Representa la contraseña del jugador que desea registrarse.

Tabla 9. Sintaxis del mensaje del cliente para el protocolo de registro de un jugador.

Comando servidor	Parámetros	Descripción
INFO	aliasJugador	Representa el alias del jugador.
	derrotas	Representa el número de derrotas del jugador.
	victorias	Representa el número de victorias del jugador.

Tabla 10. Sintaxis del mensaje del servidor.

A continuación, un ejemplo concreto:

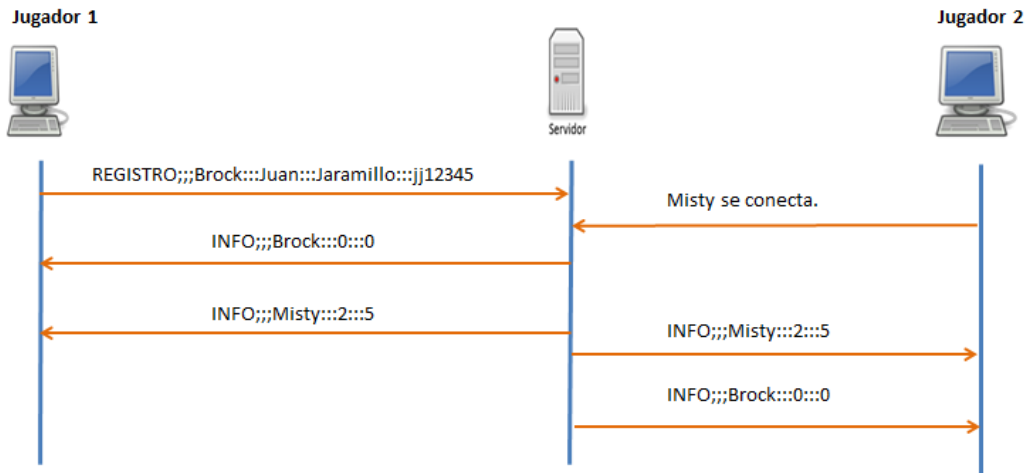


Figura 14. Ejemplo del protocolo de registro de un jugador.

En el ejemplo que muestra la Figura 14, el usuario Juan Jaramillo desea registrarse en el sistema, por lo tanto, el programa cliente envía el mensaje con comando REGISTRO seguido del alias “Brock”, nombre y apellidos: Juan Jaramillo y la contraseña “jj12345”. El servidor revisa en la base de datos si existe un jugador con el alias “Brock”, si no lo encuentra procede a agregar el usuario. Una vez se conecta un oponente, en este caso “Misty”,

el servidor le envía un mensaje con comando INFO, indicando la información del jugador (alias: Brock, derrotas y victorias: 0) y uno con la información de su oponente: alias "Misty", derrotas "2" y victorias "5".

Caso 2: Se registra un jugador que ya existe en la base de datos:

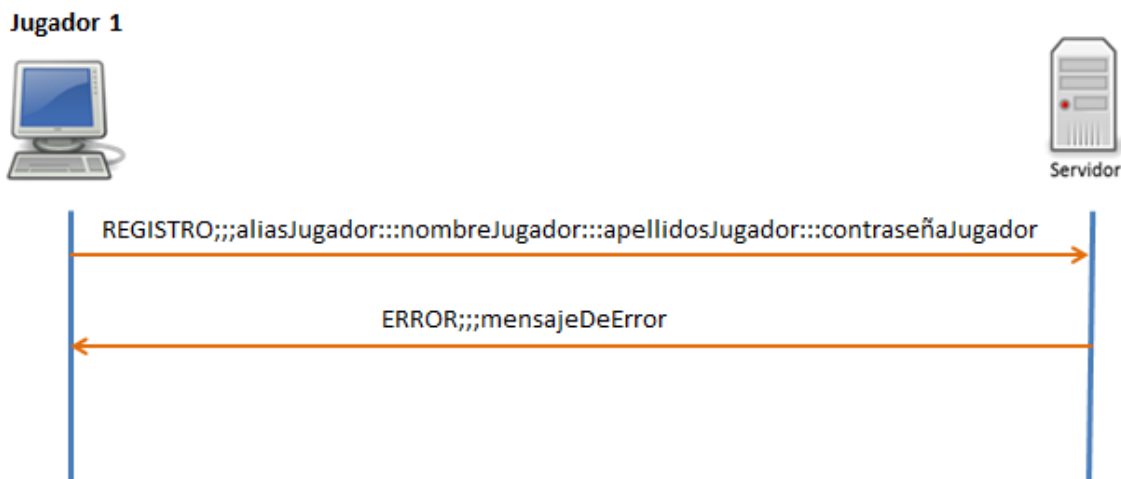


Figura 15. Protocolo de registro de un jugador cuando ocurre un error.

En la Figura 15 se aprecia el protocolo de registro de usuario en caso de error, en donde el programa cliente envía la solicitud de registro al servidor con un mensaje con comando REGISTRO, seguido del alias, nombre, apellidos y la contraseña del usuario que desea registrar. El servidor revisa en su base de datos por un usuario con el alias dado, en caso de encontrarlo entonces envía un mensaje de error al programa cliente con comando ERROR seguido del mensaje de error.

La sintaxis de cada mensaje es explicada en la **Tabla 11** y **Tabla 12**:

Comando cliente	Parámetros	Descripción
REGISTRO	aliasJugador	Representa el alias del jugador que desea registrarse.
	nombreJugador	Representa el nombre del jugador que desea registrarse.
	apellidosJugador	Representa los apellidos del jugador que desea registrarse.
	contraseñaJugador	Representa la contraseña del jugador que desea registrarse.

Tabla 11. Sintaxis del mensaje del cliente para el protocolo de registro de un jugador.

Comando servidor	Parámetros	Descripción
ERROR	mensajeDeError	Representa el mensaje de error que se requiere informar al usuario.

Tabla 12. Sintaxis del mensaje del servidor para el protocolo de registro de un usuario en caso de error.

A continuación, un ejemplo concreto:

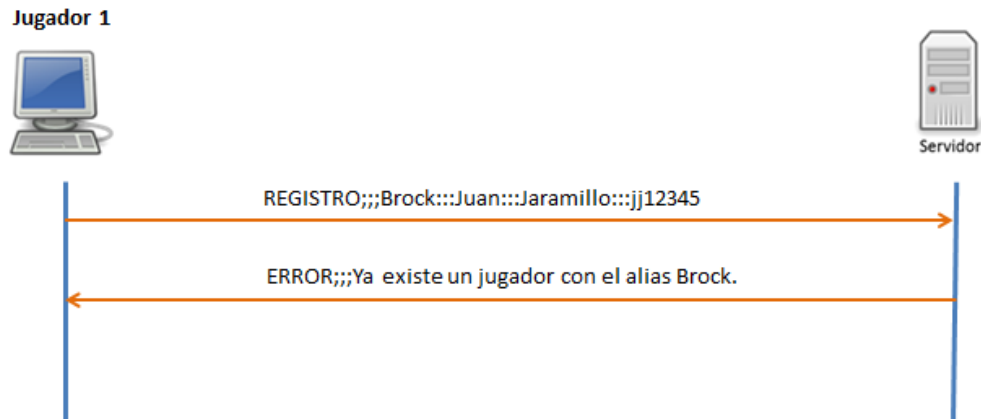


Figura 16. Ejemplo del protocolo de inicio de sesión cuando ocurre un error.

La Figura 16 muestra el siguiente ejemplo: el jugador envía una petición de registro de jugador con el comando `REGISTRO` seguido del alias, nombre, apellidos y contraseña del jugador: “Brock”, “Juan”, “Jaramillo” y “jj12345” respectivamente. El servidor al revisar la base de datos encuentra un jugador con el alias, por lo tanto envía al programa cliente el mensaje de error.

Protocolo para asignar turno:

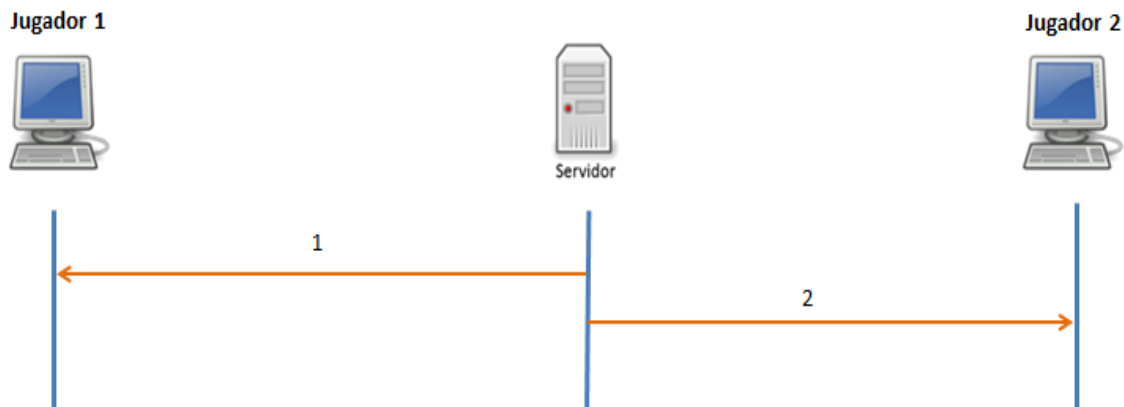


Figura 17. Protocolo para asignar turno.

En la Figura 17. Protocolo para asignar turno. Figura 9 se aprecia el protocolo para asignar turno. Ocurre cuando el servidor inicia una batalla con dos clientes. El servidor elige aleatoriamente el turno que tendrán los jugadores, enviando un mensaje al cliente de cada jugador con el comando 1 y el comando 2, para informar el primer y segundo lugar, respectivamente.

La sintaxis de cada mensaje es explicada en la Tabla 13:

Comando servidor	Parámetros	Descripción
1		Representa el primer turno.
2		Representa el segundo turno.

Tabla 13. Sintaxis del mensaje del servidor para el protocolo de asignación de turnos.

A continuación, un ejemplo concreto:

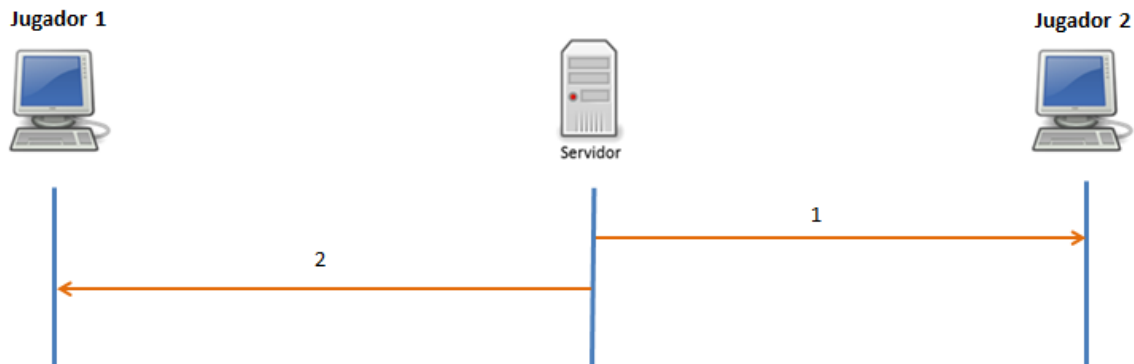


Figura 18. Ejemplo del protocolo para asignar turno.

En la Figura 18 , se muestra el siguiente ejemplo: el servidor escoge aleatoriamente los turnos y envía un mensaje al Jugador 1 informando que tendrá el segundo turno y uno al Jugador 2, informando que tendrá el primer turno.

Protocolo para enviar la selección inicial de un jugador:



Figura 19. Protocolo para enviar la selección de un jugador.

En la Figura 19 se aprecia el protocolo para enviar al oponente la selección inicial del jugador. Ocurre cuando el cliente del jugador envía un mensaje con comando SELECCION seguido del nombre del

pokémon y el avatar con el que jugará. El servidor, recibe y envía el mismo mensaje al jugador oponente. El oponente recibe el mensaje y guarda los datos enviados.

La sintaxis de cada mensaje es explicada en la Tabla 14:

Comando servidor	Parámetros	Descripción
SELECCION	pokemonSeleccionado	Representa el nombre del pokémon seleccionado para jugar.
	avatarSeleccionado	Representa el avatar seleccionado para jugar.

Tabla 14. Sintaxis del mensaje del servidor para el protocolo para enviar la selección del jugador.

A continuación, un ejemplo concreto:

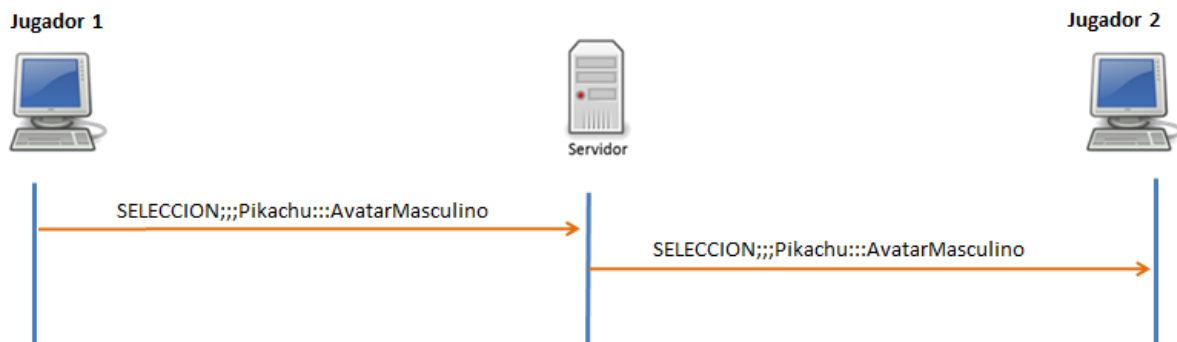


Figura 20. Ejemplo del protocolo para enviar la selección de un jugador.

En la Figura 18, se muestra el siguiente ejemplo: El Jugador 1 envía el mensaje SELECCION indicando que eligió a Pikachu como pokémon inicial y un avatar masculino. Luego, el servidor envía este mensaje al oponente. El oponente almacena la información recibida.

Protocolo para realizar un ataque:

Caso 1: El pokémon atacado tiene puntos de salud después del ataque.

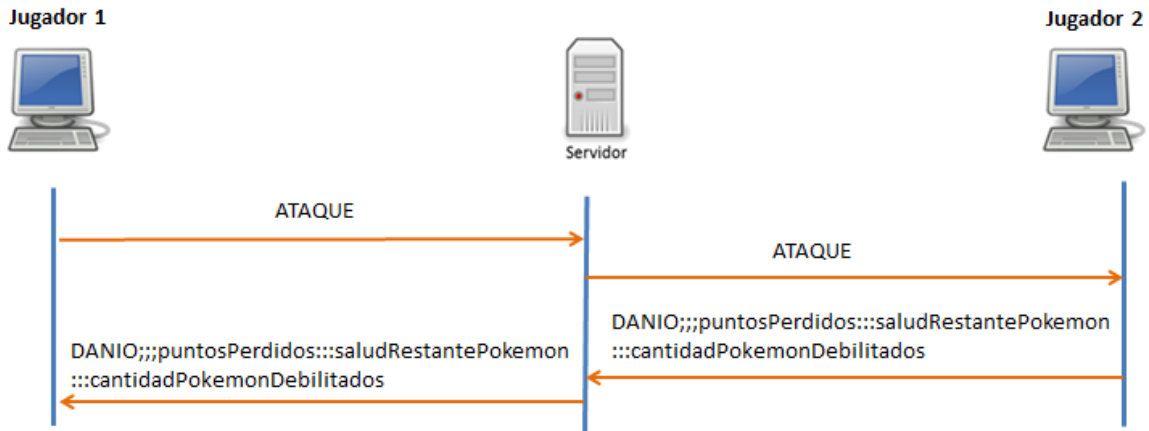


Figura 21. Protocolo para realizar un ataque y el pokémon atacado queda salud.

La Figura 21 muestra el protocolo para realizar un ataque, que consiste en que el programa cliente envía un mensaje con el comando ATAQUE. Una vez recibido el mensaje, el servidor envía esta misma información al oponente. El oponente recibe el mensaje y calcula el daño que recibe su pokémon, según el tipo de pokémon de su oponente. El programa cliente del oponente, envía un mensaje con el comando DANIO seguido de los puntos que perdió su pokémon, la salud con la que quedó, y la cantidad de pokémon debilitados hasta el momento. Luego, el servidor enviar el mismo mensaje al jugador, para informar el resultado de su ataque. Una vez el jugador recibe el mensaje, actualiza los mensajes que tiene de la batalla y la cantidad de pokémon debilitados.

La sintaxis de cada mensaje es explicada en la Tabla 15:

Comando cliente	Parámetros	Descripción
ATAQUE		Representa el ataque realizado por el jugador.
DANIO	puntosPerdidos	Representa la cantidad de puntos que perdió el pokémon tras ejecutar el ataque enviado por el jugador.
	saludRestantePokemon	Salud actual del pokémon, luego de efectuar el ataque enviado.
	cantidadPokemonDebilitados	Cantidad de pokémon que tienen la salud en cero, luego de efectuar el ataque.

Tabla 15. Sintaxis del mensaje del cliente para el protocolo para realizar un ataque.

A continuación un ejemplo concreto:

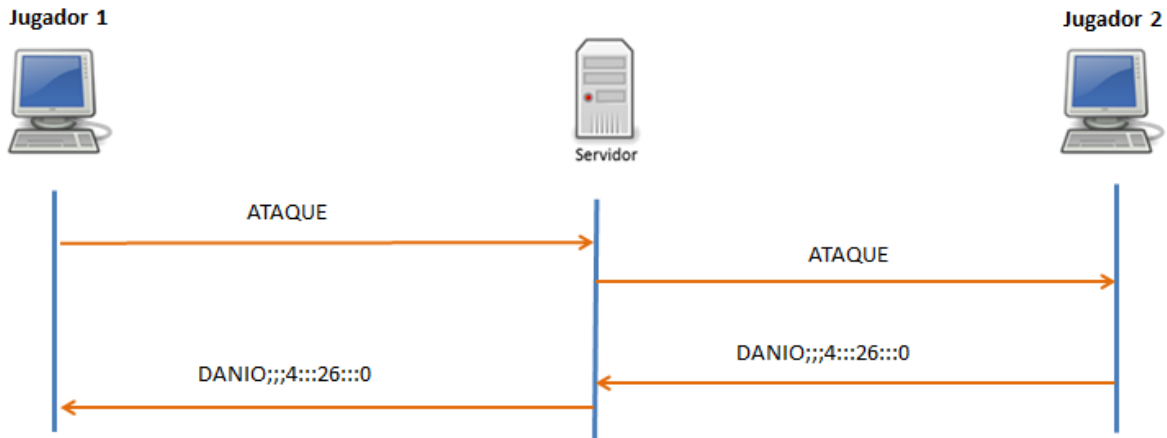


Figura 22. Ejemplo del protocolo para realizar un ataque.

La Figura 22 muestra el siguiente ejemplo: el jugador envía un mensaje con el comando ATAQUE. El servidor reenvía el mismo mensaje al oponente. El oponente calcula el daño recibido y envía un mensaje con el comando DANIO seguido del daño que tuvo el Pokémon: "4", la salud con la que el Pokémon quedó: "26" y la cantidad de Pokémon que tiene su salud en cero: "0". Una vez el servidor recibe el mensaje, se lo envía al cliente del jugador.

Caso 2: El Pokémon atacado queda con cero puntos de salud.

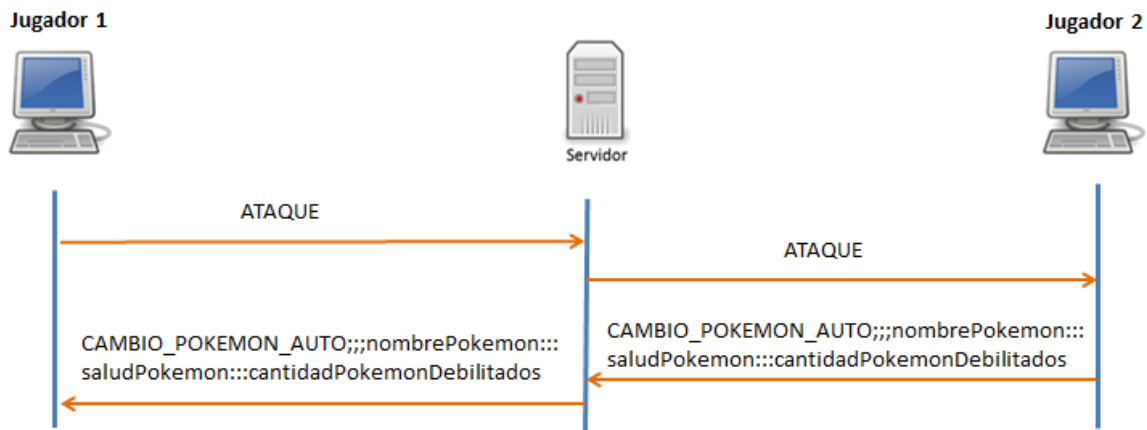


Figura 23. Protocolo para realizar un ataque y el Pokémon del oponente queda con cero puntos de salud.

La Figura 23 muestra el protocolo para realizar un ataque en caso de que el Pokémon del oponente pierda toda su salud. El jugador envía un mensaje con el comando ATAQUE. El servidor reenvía el mensaje. El oponente calcula el daño, si la salud del Pokémon del oponente queda en cero, este le envía un mensaje al servidor con

el comando CAMBIO_POKEMON_AUTO, seguido del nombre del nuevo pokémon seleccionado, la salud de ese pokémon y la cantidad de pokémon con salud en cero. El servidor envía esta misma información al jugador.

La sintaxis de cada mensaje es explicada en la Tabla 16:

Comando cliente	Parámetros	Descripción
ATAQUE		Representa el ataque realizado por el jugador.
CAMBIO_POKEMON_AUTO	nombrePokemon	Representa el nombre del nuevo pokémon seleccionado por el oponente.
	saludPokemon	Representa la salud del nuevo pokémon seleccionado por el oponente.
	cantidadPokemonDebilitados	Representa la cantidad de pokémon que quedaron con salud en cero.

Tabla 16. Sintaxis del mensaje del cliente para el protocolo para realizar un ataque.

A continuación un ejemplo concreto:

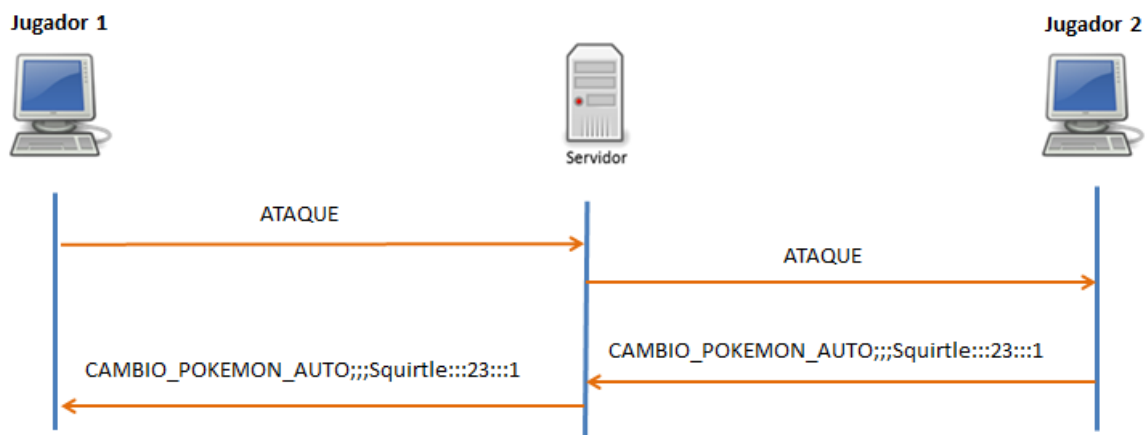


Figura 24. Ejemplo del protocolo para realizar un ataque y el oponente debe cambiar automáticamente su pokémon.

En la Figura 24, el cliente del jugador envía un mensaje con el comando ATAQUE. El servidor notifica al cliente del oponente con el mismo mensaje. El cliente del oponente calcula el daño y notifica con un mensaje que cambió automáticamente su pokémon a “Squirtle”, la salud de este pokémon es 23 y tiene un pokémon con salud en cero. El servidor recibe el mensaje y notifica al cliente del jugador.

Caso 3: El jugador se desconecta y el servidor estaba esperando su ataque.

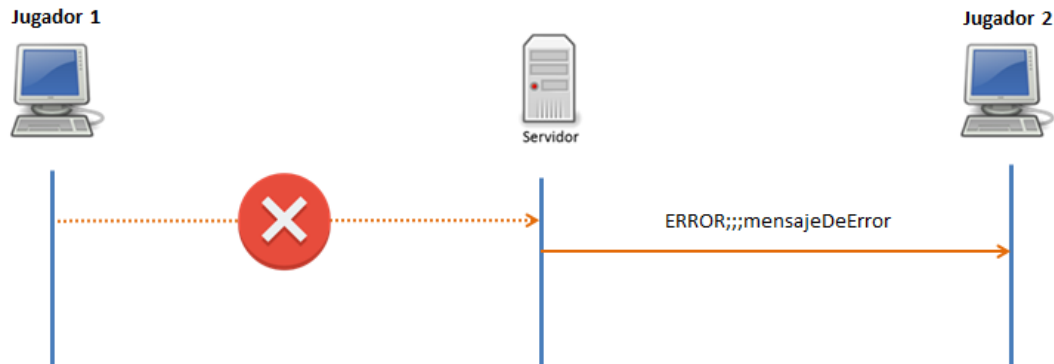


Figura 25. Ejemplo del protocolo para realizar un ataque, en caso de error.

En la Figura 25 se aprecia cuando ocurre un error, en el protocolo para realizar un ataque. El jugador que tiene el turno, se desconecta y el servidor notifica al oponente, enviando el mensaje con el comando ERROR seguido del mensaje.

La sintaxis de cada mensaje es explicada en la Tabla 17:

Comando servidor	Parámetros	Descripción
ERROR	mensajeDeError	Representa el mensaje de error que se requiere informar al usuario.

Tabla 17. Sintaxis del mensaje del servidor para el protocolo de realizar un ataque en caso de error.

A continuación un ejemplo concreto:

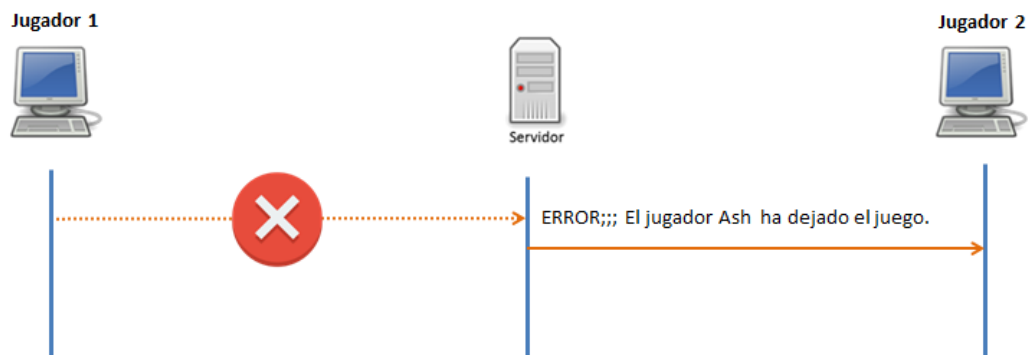


Figura 26. Ejemplo del protocolo para realizar un ataque, en caso de error.

En la Figura 26 se aprecia el siguiente ejemplo: el jugador con el alias “Ash” se desconecta. El servidor envía un mensaje de error al oponente, con el comando ERROR, seguido del mensaje: “El jugador Ash ha dejado el juego”.

Protocolo para cambiar de pokémon:

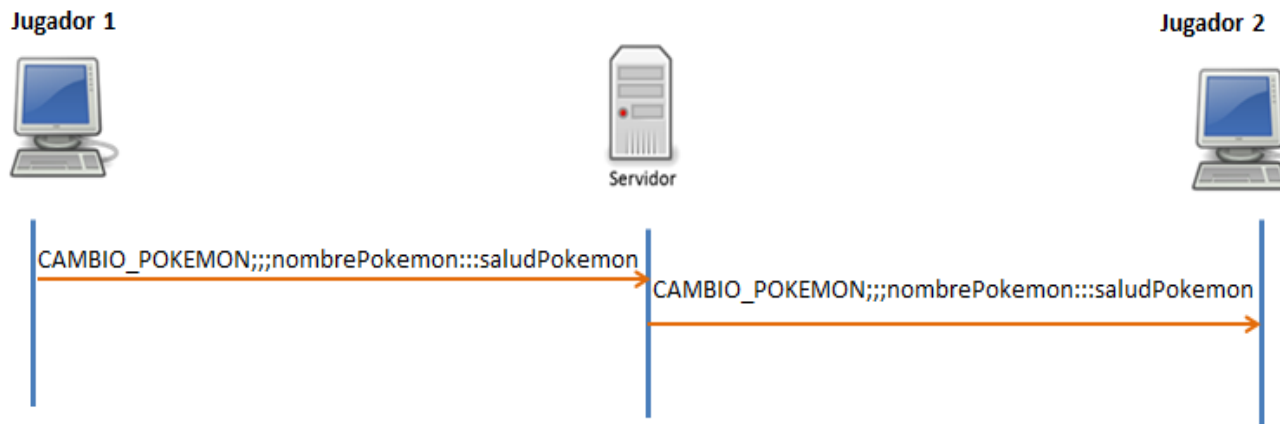


Figura 27. Protocolo para cambiar de pokémon.

En la Figura 27 se aprecia el protocolo para cambiar de pokémon. El jugador en turno envía un mensaje al servidor con el comando CAMBIO_POKEMON, seguido del nombre del pokémon seleccionado y la salud de este. Una vez recibido el mensaje, el servidor envía el mismo mensaje al oponente.

La sintaxis de cada mensaje es explicada en la Tabla 18:

Comando cliente	Parámetros	Descripción
CAMBIO_POKEMON	nombrePokemon	Representa el nombre del nuevo pokémon seleccionado.
	saludPokemon	Representa la salud del nuevo pokémon seleccionado.

Tabla 18. Sintaxis del mensaje del cliente para el protocolo de cambiar pokémon.

A continuación un ejemplo concreto:

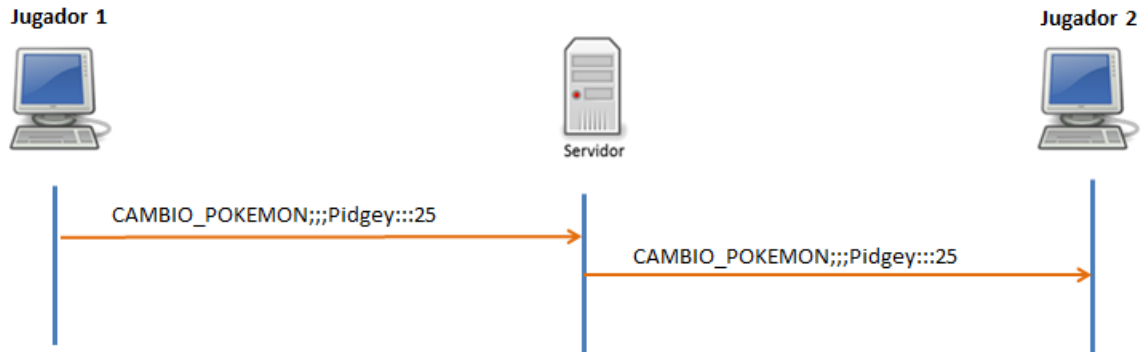


Figura 28. Ejemplo del protocolo para cambiar de Pokémon.

En la Figura 28 se muestra el siguiente ejemplo: El cliente del jugador envía un mensaje con el comando CAMBIAR_POKEMON seguido del nombre del Pokémon y su salud: "Pidgery" y 25 respectivamente. El servidor notifica al cliente del oponente, reenviando el mensaje.

Protocolo para finalizar la batalla:

El Pokémon atacado queda con cero puntos de salud y era el último Pokémon disponible.

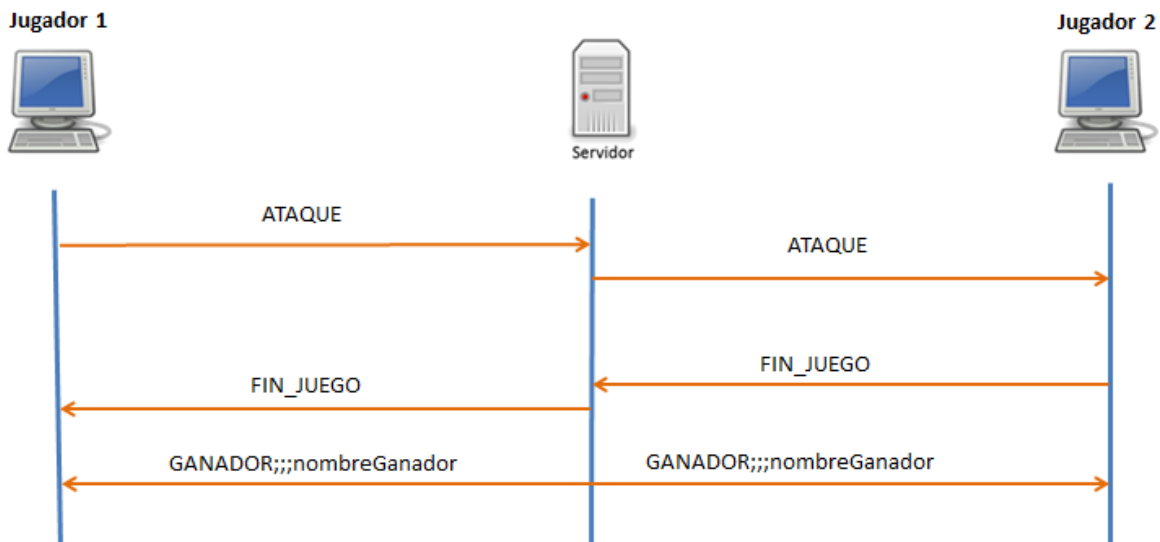


Figura 29. Protocolo para realizar un ataque, en caso de que finalice el juego.

La Figura 29 muestra el protocolo para realizar un ataque y finalizar el juego. Ocurre cuando el cliente del jugador envía un mensaje con el comando ATAQUE. El servidor reenvía el mensaje al cliente del oponente. Si la salud del Pokémon del oponente queda en cero, y era el último Pokémon disponible, el jugador le informa al servidor que el juego finalizó, enviando el mensaje con el comando FIN_JUEGO. Una vez el servidor recibe el

mensaje, envía esta misma información al oponente. Luego, envía un mensaje con el comando GANADOR, seguido del alias del ganador, a ambos clientes.

La sintaxis de cada mensaje es explicada en la Tabla 19 **Tabla 19** y Tabla 20 **Tabla 20**:

Comando	Parámetros	Descripción
ATAQUE		Representa el ataque realizado por el usuario.
FIN_JUEGO		Representa el fin de la batalla.

Tabla 19. Sintaxis del mensaje del cliente para el protocolo para realizar un ataque.

Comando servidor	Parámetros	Descripción
GANADOR	aliasGanador	Representa al jugador que ganó la batalla.

Tabla 20. Sintaxis del mensaje del servidor para el protocolo para finalizar la batalla.

A continuación un ejemplo concreto:

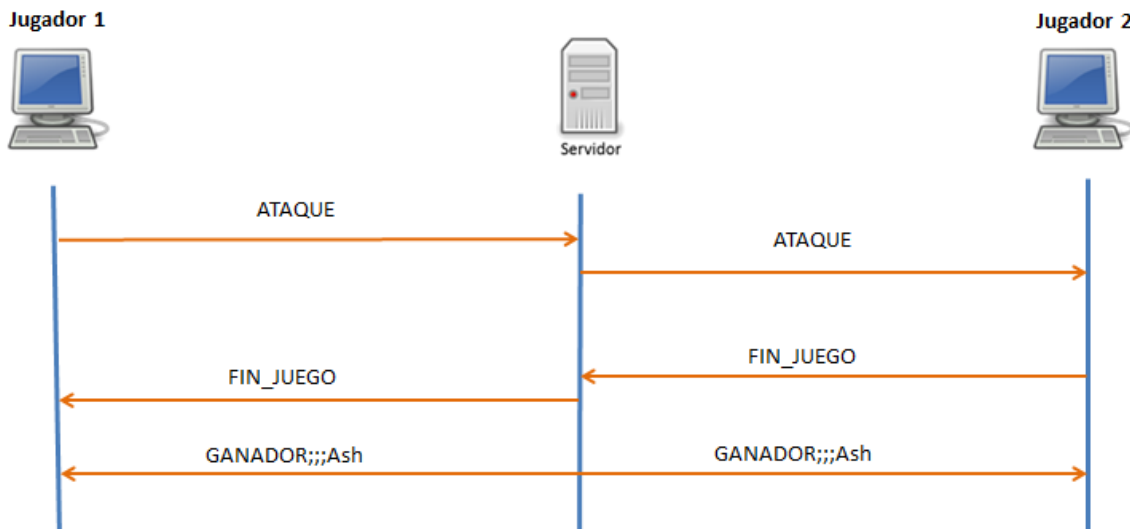


Figura 30. Ejemplo del protocolo para realizar un ataque, en caso que finalice el juego.

En la Figura 30 se aprecia el siguiente ejemplo: El cliente del jugador envía un mensaje con el comando ATAQUE. El servidor reenvía el mensaje al cliente del oponente. El cliente del oponente calcula el daño y notifica que el juego terminó, dado que su último pokémon perdió toda la salud, enviando un mensaje con el comando FIN_JUEGO al servidor. El servidor recibe el mensaje y notifica al cliente del jugador. El servidor notifica a ambos clientes el ganador mediante el mensaje con el comando GANADOR, seguido del alias "Ash".