Proyecto 2

Carlos Gerardo Acosta Hernández
Andrea Itzel González Vargas
Luis Pablo Mayo Vega

Redes de Computadoras
Facultad de Ciencias, UNAM

Índice

1. Especificación de requerimientos

1.1. Enunciado del problema

De acuerdo con el modelo TCP/IP, *la Capa de Aplicación* es la de mayor importancia y en la que se sustenta todo el desarrollo de redes de computadoras pues está compuesta por los protocolos y demás servicios encargados de manejar, intercambiar o decodificar los datos que los usuarios se envían a través de distintos hosts para comunicarse. En otras palabras, se busca que dos o más procesos en distintas computadoras puedan ser capaces de intercambiar información y de esta manera otorgar una mayor capacidad de procesamiento y mejor rendimiento del que se tendría con un único host.

Sin embargo, la *Capa de Aplicación* no puede trabajar sola, siendo mediante un proceso de encapsulación en cada una de las capas, que los datos(PDU) se envian a las capas inferiores, añadiendo cada una de las capas información que le concierne para que sus protocolos puedan manejarlos.

Como sabemos, protocolos como HTTP, DNS, FTP, SMTP o DHCP tienen cada uno una estructura distinta ya bien definida y estandarizada en los articulos de *Request for Comments* publicados, pero eso no significa que sean los únicos protocolos disponibles en la *Capa de Aplicación*. La gran ventaja de esta capa es su adaptabilidad para que se desarrollen protocolos de acuerdo a las necesidades de la aplicación sobre la que se usarán.

1.2. Objetivo de la aplicación

Esta aplicación fue creada con el objetivo de implementar un protocolo de la capa de aplicación en el que el usuario, conectado del lado del cliente, solicite un pokémon a capturar. El servidor elegirá aleatoriamente alguno de los pokémon que estén en su base de datos, se lo ofrecerá al usuario y, si éste acepta capturarlo, también aleatoriamente se indicará si logró capturarlo o no.

El usuario también podrá ser capaz de consultar su Pokédex, con los nombres de los pokémon que ha capturado y la imagen de cada pokémon que seleccione.

El protocolo que hemos diseñado se enfocará en las acciones del usuario y el servidor que involucren una comunicación entre ambos, cada una con un tipo de mensaje específico. Es decir, si un usuario quisiera capturar un pokémon, solo tendría que enviar un mensaje con el código que el servidor entienda como "quiero capturar un pokémon"sin considerar otros aspectos como el nombre del pokémon o el tamaño de la imagen que contiene a tal pokémon.

En ese sentido, diseñar nuestro propio protocolo nos permite tener mayor control sobre la tasa de transferencia de los datos dentro de la aplicación, disminuyendo los costos de la comunicación y mejorando el desempeño del programa.

1.2.1. Casos de uso

Actor	Caso de uso	Descripción
Pokentrenador	Iniciar sesión	El usuario se conecta con el servidor y se le muestra el menú principal de la aplicación
Pokentrenador	Capturar pokémon	El usuario solicita al servidor que le muestre un pokémon y decide si intenta capturarlo (con un número finito de intentos) o no
Pokentrenador	Consultar pokedex	El usuario hace una consulta para buscar un pokémon en su pokédex
Pokentrenador	Cerrar sesión	Cierra la conexión con el servidor

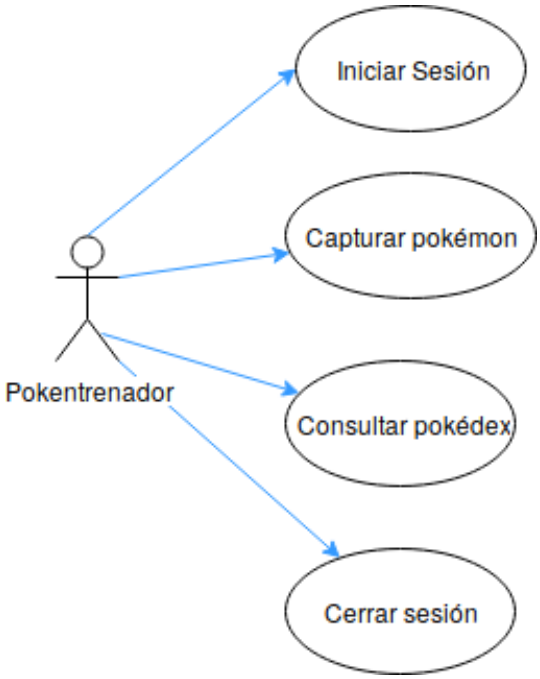
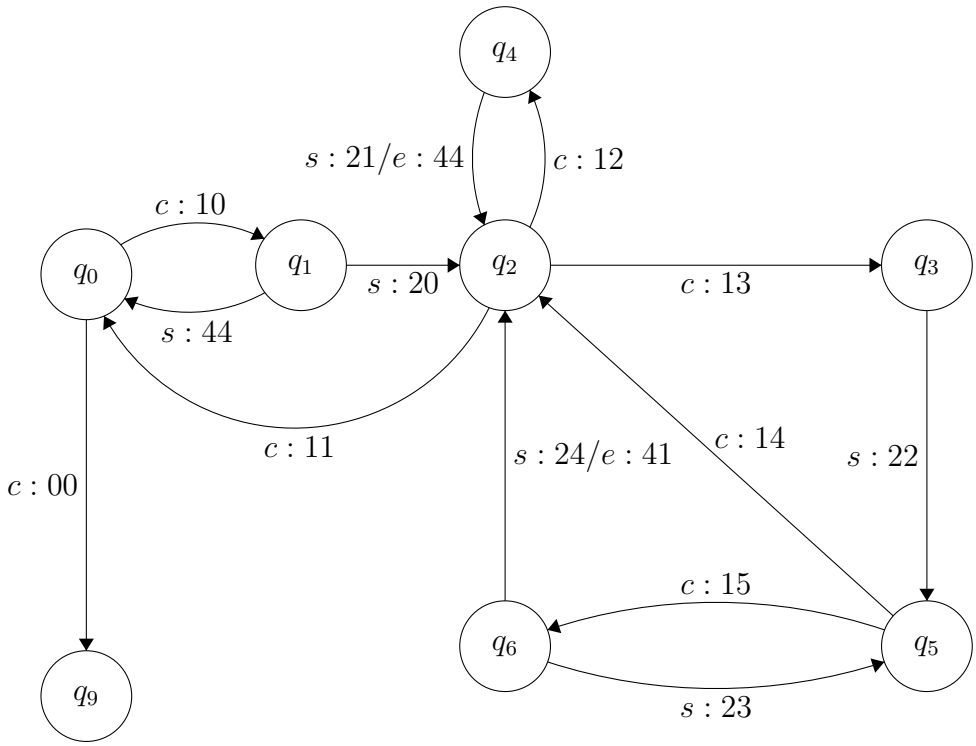


Figura 1: Diagrama de casos de uso de la aplicación

2. Diseño del protocolo

2.1. Máquina de Estados Finita

Máquina de Estados Finita para el protocolo de la capa de aplicación.



2.2. Descripción de los estados

Estado	Descripción
q_0	Conexión establecida, inicio de aplicación.
q_1	Inicio de sesión.
q_2	Menú de juego.
q_3	Solicitud de captura de *pókemon*.
q_4	Búsqueda de un pókemon en la *pókedex*.
q_5	Aparición de un *Pókemon* salvaje.
q_6	Intento de captura de *pókemon*.
q_9	Cierre de conexión.

2.3. Descripción de los mensajes en la comunicación cliente-servidor

Código	Segmento	Descripción
00	`|long|code|`	Termina la conexión para un cliente.
10	`|long|code|name|`	Solicitud de inicio de sesión del cliente. El parámetro `name` se refiere al nombre de usuario del cliente.
11	`|long|code|`	Solicitud de cierre de sesión del cliente.
12	`|long|code|name|`	Acceso y consulta a la *Pokédex* mediante el nombre de un pokémon.
13	`|long|code|`	Acceso a la opción de captura.
14	`|long|code|`	El usuario rechaza el pokémon salvaje ofrecido por el servidor.
15	`|long|code|attempts|name|`	Intento de captura del pokémon salvaje ofrecido.
20	`|long|code|`	Inicio de sesión del cliente exitoso.
21	`|long|code|long_n|name|long_img|img`	Resultado exitoso de la Pokédex con imagen del pokémon.
22	`|long|code|attempts|name|`	Selección aleatoria de un pokémon salvaje para el cliente. Incluye el máximo número de intentos.
23	`|long|code|attempts|name|`	Pokémon no capturado, disminución del número de intentos.
24	`|long|code|long_n|name|long_img|img|`	Pokémon capturado con imagen incluída.
41	`|long|code|`	Error: Máximo número de intentos de captura alcanzados.
44	`|long|code|`	Error: Consulta infructuosa.

2.4. Diseño de la base de datos

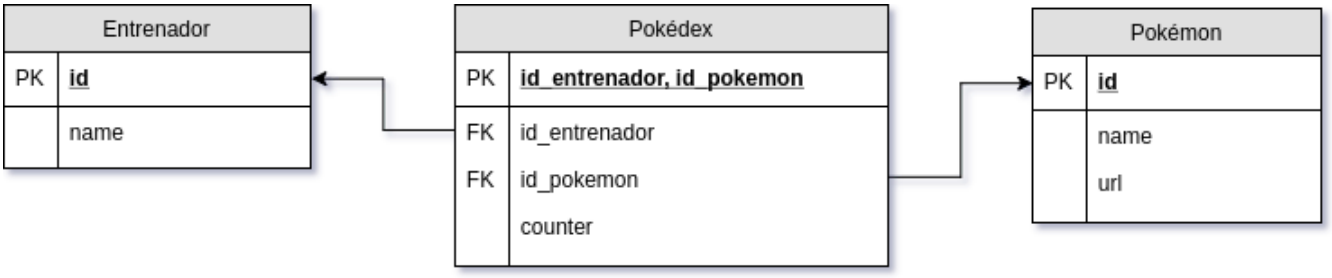


Figura 2: Diagrama de la base de datos relacional.

3. Implementación del protocolo

3.1. Especificación del ambiente de desarrollo

Lenguaje de programación:	Java	**versión:**	1.8
Herramienta de construcción de software:	Apache Ant	**versión:**	1.9.9
Sistema Manejador de Base de Datos:	sqlite	**versión:**	2.8.17
Driver de conectividad JDBC:	sqlite-jdbc	**versión:**	3.16.1
Pruebas Unitarias:	JUnit	Hamcrest	

3.2. Estructura del proyecto

```
.
|- proyecto2.jar
|- build
|- - ConexionBD.class
|- - redes
|- - - MultiThreadPrueba.class
|- - - Mensaje12.class
|- - - Mensaje21.class
|- - - ConexionBD.class
|- - - Mensaje24.class
|- - - Estado.class
|- - - ClienteHilo.class
|- - - FabricaMensaje.class
|- - - MensajeGenerico.class
|- - - Pokentrenador$1.class
|- - - Mensaje15.class
|- - - Pokentrenador.class
|- - - Controlador.class
|- - - test
|- - - - ControladorTest.class
|- - - Mensaje10.class
|- - - Pokeservidor.class
|- - - Mensaje22.class
|- - - Proyecto2.class
|- - - MultiThreadPrueba$Hilo.class
|- - - Imagen.class
|- - - Mensaje23.class
|- - - ClienteHilo$1.class
|- static
|- - requirements
|- - poke_script.py
|- man
|- - proyecto2.1
|- - proyecto2.1.gz
|- documentos
```

```
|- - proyecto2.out
|- - proyecto2.log
|- - graphic
|- - -
|- - proyecto2.toc
|- - proyecto2.tex
|- - proyecto2.pdf
|- - proyecto2.aux
|- doc
|- - proyecto2.out
|- - proyecto2.log
|- - proyecto2.toc
|- - proyecto2.pdf
|- - proyecto2.aux
|- src
|- - sql
|- - - poke_app_db.sql
|- - redes
|- - - Mensaje10.java
|- - - Estado.java
|- - - MultiThreadPrueba.java
|- - - Pokeservidor.java
|- - - MensajeGenerico.java
|- - - Mensaje15.java
|- - - Imagen.java
|- - - Controlador.java
|- - - Pokentrenador.java
|- - - ClienteHilo.java
|- - - ConexionBD.java
|- - - Mensaje24.java
|- - - test
|- - - - ControladorTest.java
|- - - Mensaje22.java
|- - - Mensaje21.java
|- - - Mensaje23.java
|- - - FabricaMensaje.java
|- - - Mensaje12.java
|- - - Proyecto2.java
|- build.xml
|- arbol.txt
|- lib
|- - junit.jar
|- - hamcrest-core.jar
|- - sqlite-jdbc-3.16.1.jar
```

4. Uso y pruebas del protocolo

4.1. Manual de uso

4.1.1. Instalación y conexión de cliente/servidor

La aplicación cuenta con una página de manual para sistemas operativos *NIX, en donde se detalla más en el uso de la aplicación con algunos ejemplos sencillos.
Para poder compilar y ejecutar el programa por primera vez, se requiere contar con la versión 1.8 o superior de `Java` así como la versión 1.9.9 o superior de `Apache Ant`, entonces puede emplear el comando:

```
[user@host p02] ant
```

Y el sistema automatizado de `Apache Ant` generará todos los archivos necesarios para que el programa funcione. En caso de haber errores, el mismo `Ant` le indicará cuáles son.

Una vez compilado, se puede iniciar la aplicación con la instrucción en linea de comando:

```
[user@host p02] java -jar proyecto2
```

4.1.2. Inicio de sesión

Una vez establecida la conexión entre cliente y servidor, el servidor le pedirá al usuario que ingrese su nombre para poder iniciar sesión o cerrar la conexión entre ambos hosts. Si el nombre de usuario ingresado se encuentra registrado en la base de datos de la aplicación, el servidor le mostrará al usuario con sesión iniciada su menú principal de juego.

4.1.3. Uso dentro de la aplicación

Una vez iniciada la sesión del usuario, éste tendrá acceso a un menú principal en el que se muestre las opciones que le otorga la aplicación, como capturar un pokémon, consultar su pokédex o cerrar sesión.

Si el usuario quiere capturar un pokémon, el servidor le mostrará aleatoriamente alguno de los nombres de pokémon registrados y el usuario decidirá si desea o no capturarlo. Si acepta el desafío, tendrá que lograr capturarlo antes de llegar al límite de intentos permitido.

Para consultar su pokédex, el usuario deberá ingresar el nombre del pokémon que quiera buscar. Si el pokémon ya fue capturado, se le mostrará la imagen de tal pokémon y recibirá un mensaje de error si el nombre del pokémon no es el correcto o el usuario no lo ha capturado.

Si el usuario decide cerrar su sesión, regresará a la pantalla de inicio de la aplicación y se le preguntará si desea iniciar sesión de nuevo o cerrar la conexión entre los hosts.