



Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación



Examen 2

Modelo de redes

Ing. en ciencias de la computación

Catedrático: Dr. Iván Olmos Pineda

Presenta: Alondra Sánchez Molina 201732614

Resumen

En el presente documento se plasma las metodologías y técnicas utilizadas para la realización del programa del segundo examen de la materia de Modelos de redes. Este documento, contiene además la descripción del programa, la explicación del método de encriptamiento seleccionado, y la explicación de las funciones de mezcla y md5 codificadas.

Introducción

La arquitectura de una red está compuesta por un conjunto de capas y protocolos, el modelo de referencia OSI de ISO consta de siete capas, física, enlace de datos, red, transporte, sesión, presentación y aplicación. Nosotros nos centraremos en la capa de enlace.

La capa de enlace de datos es la encargada de proporcionar un tránsito de datos confiables a través del enlace físico. Detecta y corrige cuando se están detectando errores, sincroniza equipo y maneja las redes de difusión. Cuando viajan los datos pueden ser interceptados por tercero, es por ello; que es necesario evitar enviar datos sensibles a través de la red; como son contraseñas de cuentas. Por ello, existen varios algoritmos de encriptación capaces de proporcionarnos el nivel de seguridad deseado.

Desarrollo

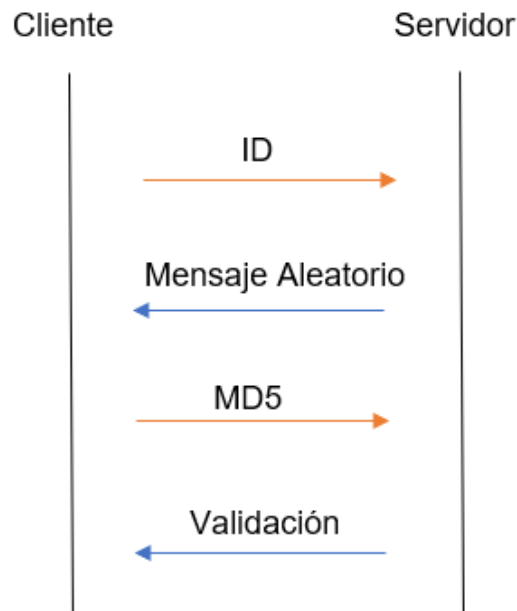
El programa está desarrollado en el lenguaje de programación Java, haciendo uso del IDE Netbeans; consta de siete clases; por parte del lado del Servidor tenemos; la clase Servidor, la cual contiene todos los métodos necesarios para que funciones nuestro servidor, la clase ServidorMain, que es nuestra clase que instancia al servidor y contiene el socket servidor para la gestión de peticiones, DataBase es simplemente el objeto que ocupa nuestro servidor para manipular los datos de nuestra base de datos.

Del lado del usuario existe la clase Cliente; la cual contiene los métodos del cliente, la clase UI_Login; es nuestra interfaz gráfica interactúa con el usuario permitiéndole registrarse o entrar al sistema. La clase ClienteMain, es la que inicia el proceso del lado del cliente y manda en pantalla la interfaz de la clase UI_Login.

Finalmente se presenta la clase MD5, la cual contiene los métodos necesarios para creación de una huella digital, y es utilizada por ambos lados, el cliente y el servidor.

Distribución Cliente - Servidor

Para una mejor visualización del sistema se crea un esquema sobre el paso de mensajes entre nuestros sockets del sistema. En el caso principal, el inicio de sesión.



El sistema comienza cuando el servidor es ejecutado para que se mantenga en espera de cualquier conexión.

Del lado del cliente, es lanzada la interfaz gráfica, la cual aún no lanza ninguna petición; cuando el usuario ingresa los datos de usuario y contraseña; se tienen dos opciones; acceder al sistema o registrarse.

Casos

A partir de esta decisión tomada, se generan cinco distintos casos listados a continuación:

El usuario procede a realizar su registro por primera vez.

Al darle clic al botón "+" el cliente envía tres parámetros al servidor.

- Si es ingreso o registro
- Usuario encriptado
- Contraseña encriptada

Siendo esta la única vez que el cliente mande la contraseña por la red; el servidor recibe los datos, descripta, busca en la base de datos el usuario recibido.

1. En el caso que no se encuentre registrado; escribe en la base y retorna al cliente "Registro con éxito"; finaliza la conexión.
2. En caso de que el usuario ya este registrado, el servidor retornará "Usuario ya utilizado" y finalizará la conexión.

Finalmente, se cerrará la conexión con el cliente.

La segunda opción del sistema, la principal, es cuando el usuario decide ingresar sesión; primeramente, este ingresa los datos y da clic en el botón iniciar sesión; el cliente encripta el id y se lo manda al usuario; este descripta y busca en la base de datos el id; aquí existen dos posibilidades.

3. El usuario no es encontrado, retornará el mensaje "ID no encontrado" y finalizará la conexión.

Si el ID si es encontrado, se generará un mensaje aleatorio que mandará al cliente; este a su vez, generará una huella digital a partir del mensaje aleatorio recibido y la contraseña proporcionada por el usuario mediante la interfaz; esta huella será enviada ahora encriptada al servidor.

Al mismo tiempo, el usuario creará su propia huella con el mismo mensaje aleatorio y con la contraseña que recuperará de la base de datos.

Finalmente, el servidor la recibirá, descriptará y comprará la huella del cliente con la huella creada por el; si estas coinciden entonces se genera el cuarto caso.

4. Se retorna al cliente un mensaje de "Bienvenido al sistema"; finaliza la conexión.
5. Si no coinciden las huellas, el servidor retornará el mensaje de "Contraseña incorrecta".

Independientemente si se cierre la ventana de ingreso al sistema o no, el socket se cerrará cada vez que finalice alguno de estos casos.

Para finalizar la ejecución de la interfaz de ingreso solo es necesario dar clic en la x para cerrar la ventana. Para finalizar al servidor, como es un servidor y siempre debe estar escuchando en espera de peticiones es necesario matar el programa.

Función Mezcla

La función mezcla se encuentra tanto como en el Cliente.java como en el Servidor.java, recibe como parámetros el mensaje aleatorio previamente creado y la contraseña del usuario; en resumen, genera un array de caracteres del tamaño de la suma de ambas cadenas recibidas, genera un for que avanzará un espacio a la vez para agregar carácter por carácter de ambas cadenas; se agregará una letra de la contraseña cada representación de esta en su número ASCII más, dependiendo si este número es par o no; 2 o uno a la posición; en caso contrario que el índice no se encuentre en esa posición se agregará un carácter del mensaje aleatorio.

Después de esto, la cadena formada se volverá a reordenar tomando un carácter del lado izquierdo y luego otro del lado derecho.

Pseudocódigo algoritmo Mezcla:

```
Mezcla(ma, pass)
{
    para cada posición en el array de aux
        si j(indice del recorrido en pass) es igual al tamaño del array pass
            g = 1; //bandera para no incrementar j
        si no
            num = (int)elemento j del arreglo pass

        si el número generado es par al número se le agrega 2 si no se suma 1

        si el número es igual a l y g = 0
            a nuestro array aux se agrega el carácter j de la cadena pass
            se incrementa el índice j
            se iguala a 0 el índice l
        si no
            a nuestro array aux se agrega el carácter k de la cadena ma
            se incrementa el índice k

        se incrementa el índice l

    si tam modulo 2 es igual a 0
        significa que la cadena es par se recorre y se concatena m primero con el
        primer carácter de la izquierda y luego con último carácter.
    si no
        significa que la cadena es impar
        if se llega al carácter de en medio
            este se coloca en la cadena
```

```

        si no
            se recorre y se concatena m primero con el
            primer carácter de la izquierda y luego con ultimo carácter.

se retorna m //nuestra mezcla de mensaje aleatorio y contraseña
}

```

MD5 Message Digest Algorithm 5

Es un algoritmo de reducción criptográfico que permite obtener una huella digital de un archivo. De esta forma, a la hora de descargar un determinado archivo como puede ser un instalador, el código generado por el algoritmo, también llamado hash, viene “unido” al archivo. Un hash MD5 está compuesto por 32 caracteres hexadecimales y una codificación de 128 bits.

Para la creación de este sistema, se programó una clase llamada MD5; la cual es utilizada desde ambos lados, tanto del cliente como el servidor para la creación de huellas digitales.

La clase MD5 costa de cinco atributos globales; las variables de tipo long A, B, C y D; que son inicializadas con valores específicos para su posterior manipulación y un array de tipo long de 64 elementos.

El primer método a describir será initT() cuya función es el llevarlo de este arreglo con los resultados de una operación aritmética. Tomando en cuenta que la función Math.sin recibe en radianes.

$$T[i] = (\text{long}) (((\text{double})4294967296L) * (\text{Math.abs}(\text{Math.sin}(i+1))));$$

Posteriormente, se encuentran 4 métodos que contienen 4 operaciones lógicas.

La función mPrima, recibe un string m, que es nuestra mezcla; genera una nueva cadena m' que contiene el mensaje original m más k bits de relleno, de tal forma que la longitud de m' sea un múltiplo de 512 bits. A continuación, se explica el funcionamiento de esta función:

1. La cadena se convierte en un array de caracteres, se recorre este array, y mientras se va recorriendo se obtiene el valor ascii de cada carácter para convertirlo ahora a binario; si este no es de 8 bits, se rellena del lado izquierdo con 0's.
2. Ahora que se convirtió el mensaje original se procede a ver si se debe crear relleno o no; hay que hacer notar que se reservarán 64 bits para el tamaño del mensaje, entonces se aplica el modulo de 512 a esto más el tamaño en bits del mensaje original; dando origen a dos casos.

- Si el mensaje original mas los 64 bits para el tamaño cumple, entonces no se necesita relleno, se calcula el tamaño, se convierte en bits y de ser necesario se llenan a 8 bits.
 - Si no, entonces se obtiene el residuo; como nuestra contraseña nunca será mayor a 64 bytes se puede usar el modulo para obtener cuantos bits faltan. Se agregan a la cadena relleno se calcula el ahora tamaño y se vuelve aplicar lo mismo de rellenar.
3. Finalmente se juntan nuestras 3 cadenas a una sola de bits; los bits del mensaje original, del relleno, y del tamaño; primero los 32 últimos, y luego los 32 primeros.

En el siguiente modulo, bloques; se manda a llamar al módulo mPrima, recibiendo el string de caracteres, se crean las subdivisiones de bits, primero en 512 bits y después cada bloque de 512 bits en 16. Retornándolos.

Ahora, para un manejo de la parte iterativa del programa, se decidió crear una función por cada etapa. Teniendo estas dentro, solamente la operación que se realizará con los bits.

En el método de control iterativo, se obtienen los bloques, se realiza el recorrido por bloques, y se realizan las 16 operaciones de las cuatro etapas existentes. Sin olvidar la actualización de nuestras variables A, B, C y D.

Finalmente, en el método creacionHuella, el método que es llamado por Cliente y Servidor; se inicializa el arreglo T, y se manda a llamar al método controlIterativo para que se realicen todos los pasos correspondientes a la manipulación de nuestras cuatro variables. Se recupera, y se convierten a binario, para crear la cadena de bits que será nuestra huella digital; puesto que nuestras variables son long, es decir; de 64 bits; se invierten y se extraen solamente los 32 bits menos significativos de cada una; agregándolos a nuestra huella en el orden correcto.

Algoritmo de encriptamiento AES

El Advanced Encryption Standard es un esquema de cifrado por bloques desarrollado por el gobierno de los Estados Unidos lanzado como estándar en el año 2002.

Criptografía simétrica significa que con la misma clave con la que genero el mensaje codificado se puede desencriptar y recuperar el mensaje original, por lo que solo requiere almacenar una sola clave.

AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 o 256 bits, mientras que Rijndael puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits.

Primeramente, está el método de crearClave, el cual consiste en pasar una clave de ser una cadena de texto a un objeto SecretKeySpec el cual es usado por las clases que incluye Java, para esto obtendremos el SHA-1 del texto que será la clave, lo pasaremos a un arreglo de bytes y usaremos eso para el constructor de la clase SecretKeySpec.

El método de encriptación recibe los datos a encriptar, esta; es realizada con un objeto Cypher el cual se inicializa indicando cual algoritmo de encriptación se usara, que operación encriptado o desencriptado se realizara y la llave que se usara.

Hecho esto basta con pasarle los datos como un arreglo de bytes al método doFinal y le regresara el arreglo de bytes del resultado de la encriptación.

Para recuperar los datos también se usa un objeto Cypher para hacer el trabajo y se inicializa con ayuda de la clave de encriptación, la principal diferencia es que se le indica que la operación será la desencriptación de datos.

Ya preparado el objeto Cypher basta con pasarle los bytes de los datos encriptados al método doFinal y se obtendrá los bytes del mensaje original, solo se convierten de byte[] a String.

Ejemplos

A continuación, se mostrarán cinco ejemplos que hacen referencia a los cinco casos posibles del sistema:

Para todos estos casos aplicará la siguiente base de datos.

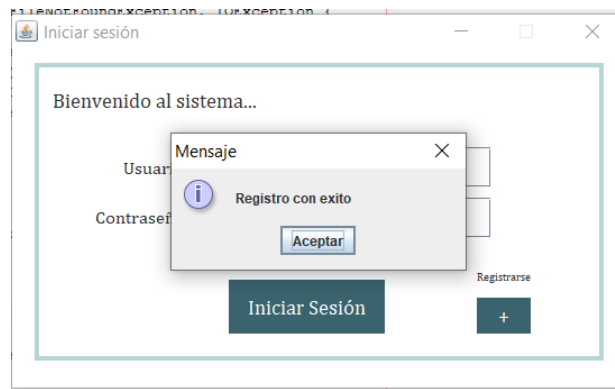


```
dataBase: Bloc de notas
Archivo Edición Formato Ver Ayuda
JuanP3r3z,juanPi452
alo132,jukil000
Samu99,garfield45
GabsP1,perez7788
GarciaMem2,password2
AntonyS69,imironM4n
12Ventura,robets34
richardContreras,riky99
juana45,Flores3
athenea45,jukilo132
diegoT34,contr99
```


1. Registro de usuario exitoso



Se ingresan los datos a registrar.

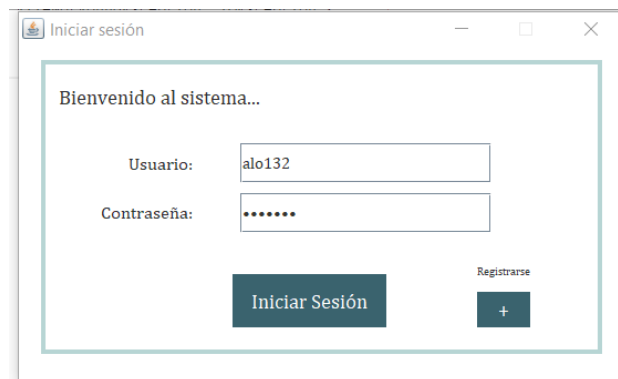


El sistema retorna el mensaje de registro con éxito.

```
dataBase: Bloc de notas
Archivo Edición Formato Ver Ayuda
JuanP3r3z,juanPi452
alo132,jukil000
Samu99,garfield45
GabsP1,perez7788
GarciaMem2,password2
AntonyS69,imironM4n
12Ventura,robets34
richardContreras,riky99
juana45,Flores3
athenea45,jukilo132
diegoT34,contr99
robert67,rob3468
```

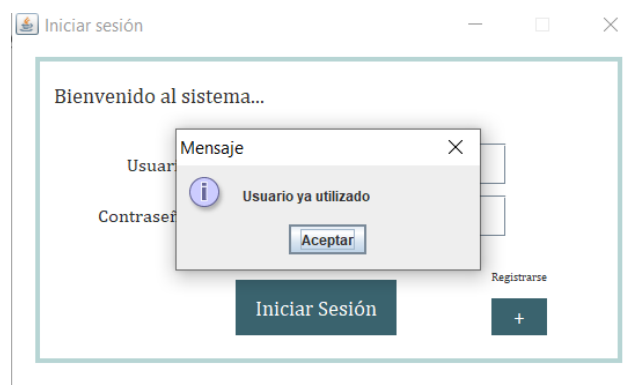
Ahora la base se ve así.

2. Usuario ya esta registrado



A screenshot of a web application window titled "Iniciar sesión". The window contains a form with the heading "Bienvenido al sistema...". Below the heading are two input fields: "Usuario:" with the value "alo132" and "Contraseña:" with masked characters ".....". At the bottom of the form are two buttons: "Iniciar Sesión" and "Registrarse" with a "+" icon. The "Registrarse" button is disabled.

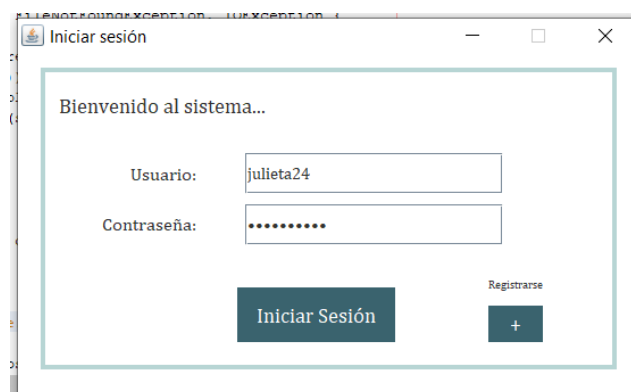
Se ingresan los datos a registrar.



A screenshot of the same "Iniciar sesión" window. An error message dialog box titled "Mensaje" is overlaid on the form. The message says "Usuario ya utilizado" with an information icon. There is an "Aceptar" button in the dialog. The background form is partially obscured by the dialog.

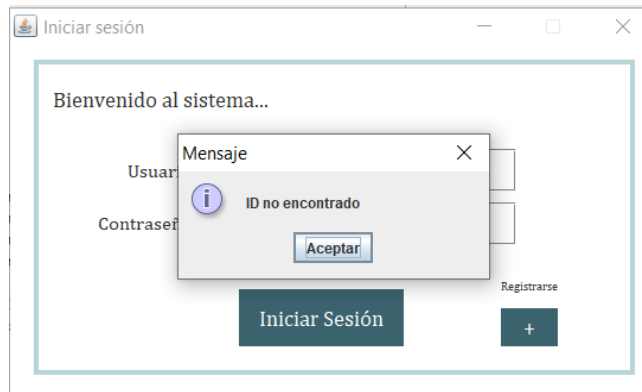
El sistema retorna un mensaje de Usuario ya utilizado.

3. Usuario no encontrado



A screenshot of the "Iniciar sesión" window. The "Usuario:" field contains the value "julieta24" and the "Contraseña:" field contains masked characters ".....". The "Iniciar Sesión" and "Registrarse" buttons are visible at the bottom.

Se ingresan los datos para iniciar sesión.

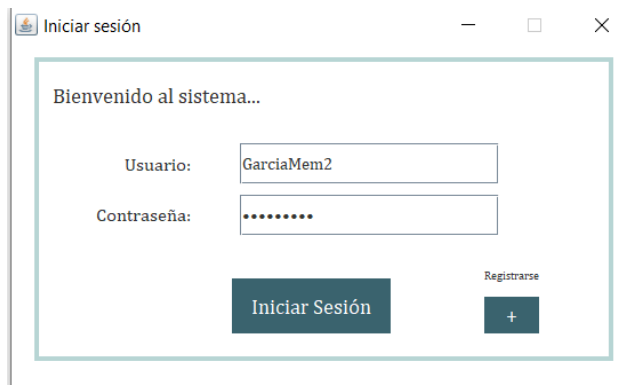


Si el servidor no encuentra en la base de datos el id ingresado, en vez del mensaje aleatorio regresa una cadena con el mensaje “ID no encontrado” y se finaliza la conexión.

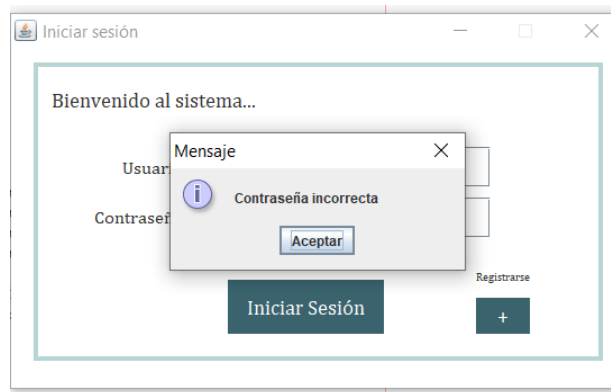
A screenshot of a terminal window with two tabs: "huellasDigitales (run)" and "huellasDigitales (run) #2". The first tab is active and shows the following text:

```
run:|
-----SERVIDOR-----
Esperando conexión...
Conexion aceptada
Usuario recibido: Julieta24
Conexión finalizada
```

4. Contraseña incorrecta



Este caso ocurre cuando se ingresa un usuario que, si se encuentra registrado en la base, pero la huella que manda el usuario no coincide con la huella que genera el servidor.



Por el lado del cliente se manda este mensaje gráficamente, y, para fines de la revisión del examen se muestra en consola los valores transmitidos.

```

Output x Usages
huellasDigitales (run) x huellasDigitales (run) #2 x
run:
----CLIENTE----
Mensaje Aleatorio recibido: jFt`zclK2V,-E;dzIN\p
Mezcla del cliente: jKFst0`Kz>cml*k72SV',O-`E|;..
Huella del cliente: 10001001101111111000000110110
|

```

Cabe recalcar que al ser el mensaje aleatorio de 4096 bytes su visualización es algo costosa, pero apretando las teclas Ctrl+R estando en el apartado de Output se visualizará todos los datos; esto sucederá en este caso y en el caso 5.

Del lado del servidor, igual se visualizarán para fines de revisión estos valores.

```

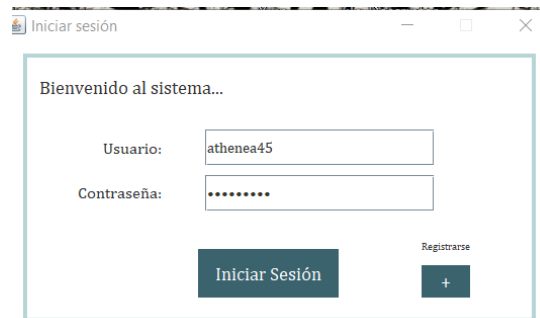
huellasDigitales (run) x huellasDigitales (run) #2 x
Usuario recibido: Julieta24
Conexión finalizada
Conexión aceptada
Usuario recibido: GarciaMem2
Mezcla del servidor: jKFst0`Kz>cml*k72SV',O-`E|;..dMzMIWNj\Hprut=s/a3j=_#UVODBSyS2Sm6H?j*oR(o)!$a5s/yYt_F(,))E:9t%l}2u2)/""Yebi4v3m0\24BT?vWKEKk; tPkirH6N"7otLHio[Ja+
Huella del servidor: 1000100110111111100000011010010110011000001100000001001011111010000001101001100011001101101010100001011100111010101101000100
Huella del cliente recibida: 100010011011111110000001101001111001100000110000000100100101110010000111001000110011110000001100010111001010100011110111
Conexión finalizada
|

```

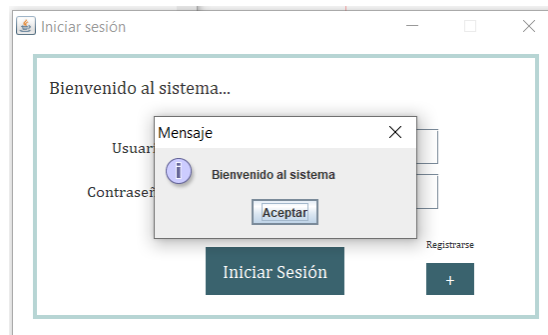
En este caso podemos observar que las huellas no son las mismas.

5. Contraseña correcta

a) Ejemplo 1



El usuario inicia sesión con un ID que se encuentre dentro de la base y con la contraseña correcta.



Se manda un mensaje al usuario de Bienvenido al sistema.

```
Output X Usages
huellasDigitales (run) x huellasDigitales (run) #2 x
Huella del cliente recibida: 100010011011111100000110110011110011000011000000010010010111001000011101001100011001111000001110001011101010100011110111
Conexión finalizada
Conexión aceptada
Usuario recibido: athenea45
Mezcla del servidor: |C>yB*Plw6v#PB7Ce& #>+qwp\s+SAC<(v1L.cE;DarK_mrh)=;QJIX)*(2IeW1F:F$8x42+v')U6$pk4%Hsgiaeih/sho.8hF#_D6"\m8'"Ef1_jd#qT#eJ)x{PmvFNS"@JvTW+
Huella del servidor: 10001001101111101000010111010001100000000101100001010010000111010011000110001011100110100010111001110101111011100001011
Huella del cliente recibida: 100010011011111010000101110100011001100001100000001011000010100100001110100110001100010111001101001110111011100001011
Conexión finalizada
|
<
```

Del lado del servidor podemos ver desde la conexión aceptada, el id recibido, la mezcla creada, la huella creada, la huella recibida y que la sesión fue finalizada.

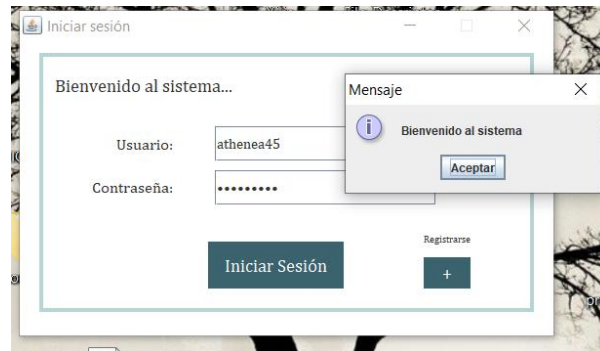
Como podemos ver ambas huellas coinciden.

```
Output X Usages
huellasDigitales (run) x huellasDigitales (run) #2 x
run:
-----CLIENTE-----
Mensaje Aleatorio recibido: |>BPwvP7e +qpsGC(1.E;aKmh=QI)(IWFFS4+'U$K$siehso8F_6\8"f_dg%XjFvN"Jt=2bdmqQ:8XW 2H*.<ht6g=1rlOWD 1PLW._"6Q)"s'? (CVBY0\QWjeJ
Mezcla del cliente: |C>yB*Plw6v#PB7Ce& #>+qwp\s+SAC<(v1L.cE;DarK_mrh)=;QJIX)*(2IeW1F:F$8x42+v')U6$pk4%Hsgiaeih/sho.8hF#_D6"\m8'"Ef1_jd#qT#eJ)x{PmvFNS"
Huella del cliente: 100010011011111010000101110100011001100001100000001011000010100100001110100110001100010111001101001011100111010111011100001011
|
<
```

Del lado del cliente tenemos el mensaje aleatorio recibido la mezcla y la huella generadas.

b) Ejemplo 2

En el caso que este mismo usuario quisiera ingresar nuevamente al sistema, la huella no volverá a hacer la misma.



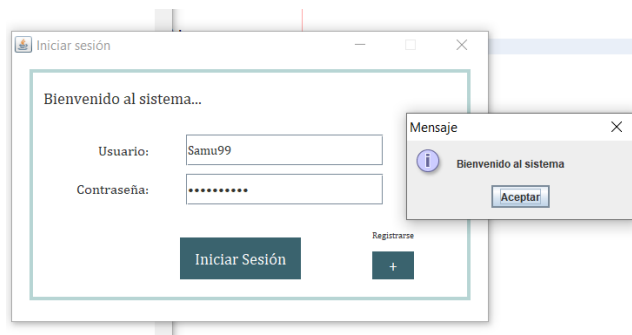
Servidor:

```
Output x Usages
huellasDigitales (run) x huellasDigitales (run) #2 x
Conexión aceptada
Usuario recibido: athenea45
Mezcla del servidor: ]7Mp"+7TbB;v:^) yo8`W;i2\q76S,3$lpDB`e"Zoda|Kf]g,g~*RuAb=$R(2$.%u0t@FQD\ tCS,s<zX2Bg;/c\8tWm|o<zKitULeyXceKEHSSp[z:THQe"J%15xLaH2!bs^;M2!8
Huella del servidor: 10001001101111101010000010100110100110000110000000001011001111100000111010011000111111000011000000101110010010110111101101100
Huella del cliente recibida: 10001001101111101010000010100110100110000110000000001011001111100001110100110001111110000110000001011100100101101111011011100
Conexión finalizada
```

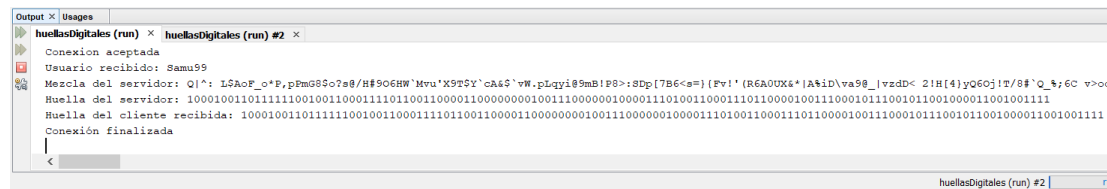
Cliente:

```
Output x Usages
huellasDigitales (run) x huellasDigitales (run) #2 x
-----CLIENTE-----
Mensaje Aleatorio recibido: ]M"?b;:)y8Wi\7831D`"oaK],-RA=R2.utFD C,<XB;c8W|<KtLycKHS[:HeJlxa2b^M\ 979k3e&AiZk }N,(=c)m[K'=q|E2Lr%94#_(Z=OuJzym"Fa,vx_hr+`
Mezcla del cliente: ]7Mp"+7TbB;v:^) yo8`W;i2\q76S,3$lpDB`e"Zoda|Kf]g,g~*RuAb=$R(2$.%u0t@FQD\ tCS,s<zX2Bg;/c\8tWm|o<zKitULeyXceKEHSSp[z:THQe"J%15xLaH2!bs
Huella del cliente: 10001001101111101010000010100110100110000110000000001011001111100001110100110001111110000110000001011100100101101111011011100
```

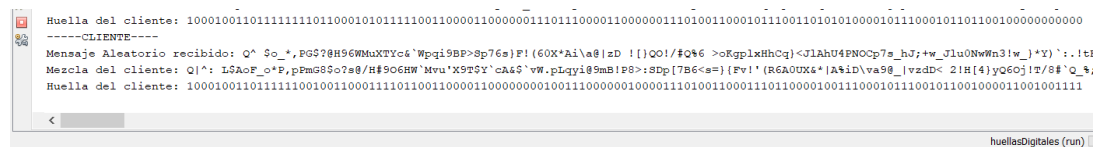
c) Ejemplo 3



Servidor:



Cliente:



Conclusión

Al finalizar la realización del sistema que simula una conexión entre un usuario un servidor haciendo énfasis en la protección de contraseñas; puedo concluir que es de vital importancia buscar siempre la mejor manera de proteger la información que pasa por la red; algoritmos como AES, nos proporcionan una mayor seguridad; pero el hacer uso de huellas digitales, como la que crea MD5 es sumamente eficiente puesto que los datos que envía el cliente a nuestro servidor siempre estarán protegidos. El cuidar de la seguridad de la red, es de vital importancia para un sistema informático.

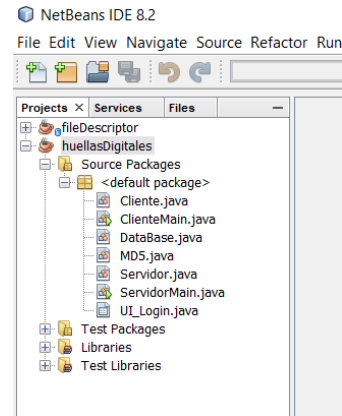
Bibliografía

- Comer, D. E., & Soto, H. A. A. (1996). Redes globales de información con Internet y TCP/IP (Vol. 1). Prentice hall.
- Olmos, I. (marzo 20, 2020). Sesión 7. Capa de enlace de datos Recuperado de: <https://www.cs.buap.mx/~iolmos/>
- Knudsen, R. A. E. B. L. (1998). Serpent: A proposal for the advanced encryption standard. In First Advanced Encryption Standard (AES) Conference, Ventura, CA.
- Raygoza, D. (septiembre 13, 2019). Encriptar Java con AES Recuperado de: <https://medium.com/el-acordeon-del-programador/encryptaci%C3%B3n-aes-en-java-ebb81ddf82b>
- P. Hawkes, M. Paddon, arid G. G. Rose, Musings on the Wang et al. MD5 collision, at <http://eprint.iacr.org/2004/264.pdf>
- Rivest, R., & Dusse, S. (1992). The MD5 message-digest algorithm.

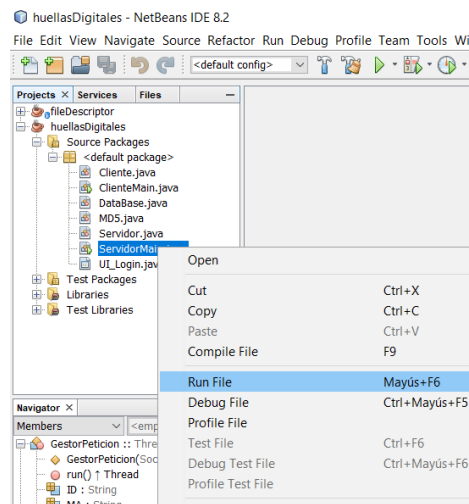
Pequeño manual de usuario

Este sistema puede ser ejecutado haciendo uso de Netbeans IDE.

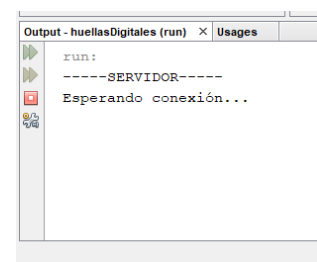
Estando dentro del programa, el proyecto huellasDigitales, es abierto; se listan 7 clases que fueron anteriormente mencionadas en el reporte.



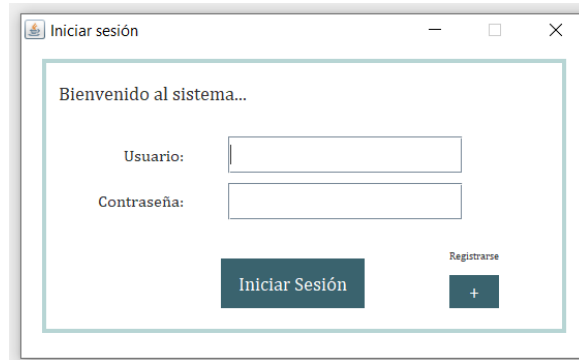
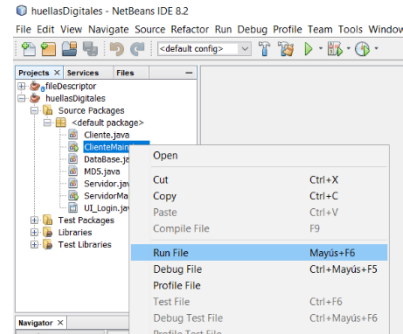
Para inicializar el servidor; es necesario darle clic derecho → Run File a la clase con el nombre ServidorMain.java



Esto hará que en el apartado Output se muestre el siguiente mensaje, donde indica que nuestro servidor espera por una petición.



Ahora iniciaremos a ClienteMain, el cual no permitirá una conexión inmediatamente, si no que solo lanzara la interfaz gráfica al usuario.

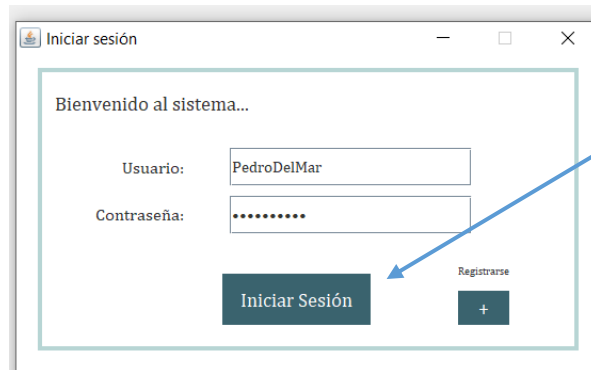


A partir de aquí se generan dos opciones, llenar los datos e iniciar sesión, o registrarse como nuevo usuario.

- Para registrarse como nuevo usuario se llenarán los datos, y después se dará clic en el botón debajo de la palabra registrarse con un símbolo de “+”.



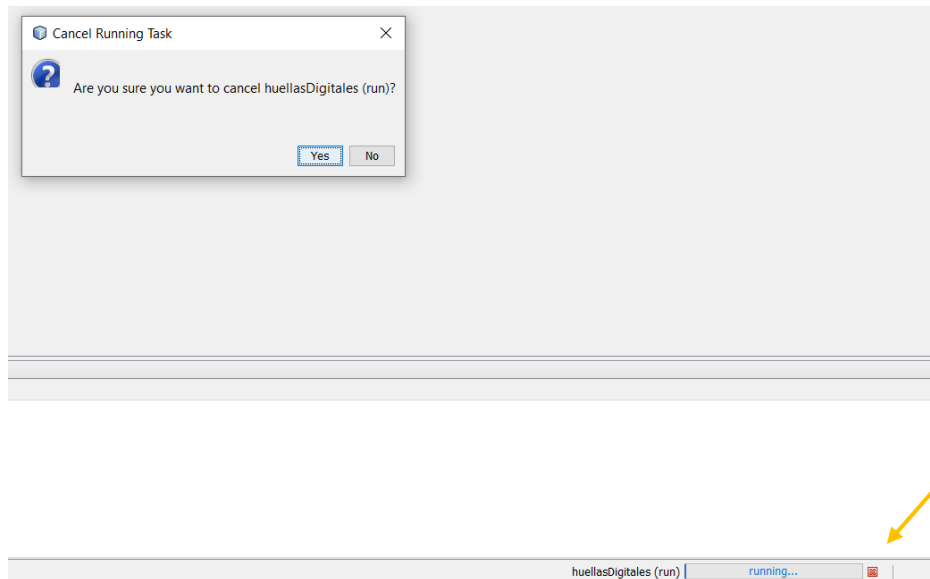
- Para iniciar sesión de igual manera, introducimos los datos primero, pero ahora se da clic en el botón iniciar sesión.



Finalmente se espera por el mensaje con el resultado del servidor, al dar click al mensaje saliente, los rectángulos se limpian y puede volver a realizar una acción o cerrar la ventana y volver a correr el programa ClienteMain sin necesidad de cerrar el servidor.

Es importante volver a recalcar que el cliente envía una petición al servidor cada vez que se utilizan los botones, y se termina esta conexión cada vez que se procesan.

Para cerrar el programa ServidorMain, es necesario cancelar la ejecución de este.



Para ello daremos clic en la x y después en Yes del mensaje saliente.