MANUAL DE SERVIDOR

Julieta Tatiana Castillo

Índice

SERVIDOR

- - - - - - - - X

El servidor de nuestro aplicativo se queda a la escucha de nuestros usuarios administradores, ellos son convertidos en hilos y nos quedamos a la disposición de sus acciones.

CLASES

- - - - - - - - X

Se tiene ayuda de las siguiente clases de negocio:

- **Sentencia.**
- **Usuario.**
- **Tabla.**

Tarjetas CRC

Nombre : Sentencia	
Responsabilidades	Esta clase está dedicada a validar las sentencias del usuario y comunicarse con él.
Colaboraciones	Se necesita de la clase: **Usuario** y **Tabla.**

Descripción de la clase Sentencia

Nombre : Usuario	
Responsabilidades	Está clase registra al usuario que ingresa a nuestro programa. Se comunica con clase de datos(**Archivo**) para almacenar los datos del cliente y un directorio personal.
Colaboraciones	Se necesita de la clase: **Archivo**.

Descripción de la clase Usuario

Nombre : Tabla	
Responsabilidades	Se llama a la clase **Tabla** para crear, eliminar y visualizar las dichas tablas que el usuario desee.
Colaboraciones	Se necesita de la clase: **Archivo**.

Descripción de la clase Tabla

Nombre : Archivo	
Responsabilidades	Es la clase que se encarga de crear, añadir y eliminar los archivos. Es el controla cada detalle de ellos.
Colaboraciones	*No tiene*

Descripción de la clase Sentencia

Funcionamiento del Servidor

Servidor escuchando si entra un usuario

```
public class Servidor {
    String directorio;

    public Servidor(){
        try {
            ServerSocket server = new ServerSocket(5000); //Se declara el puerto del servidor
            Socket sv;
            System.out.println("Servidor iniciado");
            while(true){
                sv = server.accept(); //Aceptamos al usuario que se conecto
                //Instanciamos las entradas de salida y entrada
                DataInputStream in = new DataInputStream(sv.getInputStream());
                DataOutputStream out = new DataOutputStream(sv.getOutputStream());

                //Instanciamos el hilo
                Hilo hiloUno = new Hilo(in, out);
                hiloUno.start();

            }
        } catch (IOException ex) {
            Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
    public static void main(String[] args)throws IOException, InterruptedException{
        new Servidor();
    }
```

Llamada del Hilo

```
public class Hilo extends Thread {
    static Semaphore usuario = new Semaphore(3);
    private DataInputStream vin;
    private DataOutputStream vout;
    private String vmsg;

    Hilo(DataInputStream in, DataOutputStream out) {
        vin = in;
        vout = out;
    }

    public void run() {
        try {
            realizarAccion();
        }
        catch (InterruptedException ex) {
            Logger.getLogger(Hilo.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Hilo.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

public void realizarAccion() throws IOException, InterruptedException{

    while(true){
        try {
            usuario.acquire();
            vmsg = vin.readUTF().toUpperCase(); //pasamos el mensaje a mayuscula
            if(vmsg.equals("EXIT")){ //Si el mensaje es EXIT
                break; //Salimos del while
            }

            Sentencia os = new Sentencia(); //instanciamos la clase sentacia

            String[] palabraClave = vmsg.split("\\s+"); //Separamos las palabra
            String rta = os.validarSentencia(palabraClave); //validamos
            vout.writeUTF(rta); //Le enviamos la respuesta de la consulta al cliente

        }
        catch (IOException ex) {
            Logger.getLogger(Hilo.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    usuario.release();
}
```

Validando desde Sentencia

El método central que se utiliza y valida los mensajes que nos envía el servidor es ***separarPalabras***. Realiza una comparación con un archivo .txt, en donde se almacena la sintaxis de cada sentencia de nuestra base de datos.

```
public int separarPalabras(String palabra[]){
    int col=0; int tam = 0; //Numero de columna - numero de elemento del tamaño de las columnas
    int ubicacion = 0;
    for(int i=0; i < sintaxis.length; i++){
        int cant = 0; int a= 0;
        String pal[] = sintaxis[i].split("\\s+"); //Separamos la sintaxis por palabra
        while(cant!=palabra.length){ //Mientra nos nos pasemos del tamaño limite de la consulta de cliente
            switch(pal[cant]){ //según la palabra de la sintaxis
              case "/*nombreDeUsuario*/": //Se esta indicando el nombre de un usuario
                  nomUser = palabra[a]; //Guardamos el dato en una variable global
                  cant+=1;
                  a+=1;
                  break;
              case "/*contraseña*/": //Información del usuario
                  password = palabra[a]; //Lo guardamos en una variable global
                  cant+=1;
                  a+=1;
                  break;
              case "/*nombreDeLaTabla*/": //Se indica el nombre de una tabla
                  tabla = palabra[a]; //Lo guardamos en una variable global
                  //Crear tabla
                  cant+=1;
                  a+=1;
                  break;
```

```
        case "/*tamaño*/":
            a = ubicacion;
            int index = ubicacion; //Le damos a indix el mismo valor de a
            while(!palabra[a].equals(")")){ //Ponemos la cantidad de columnas que tenemos
                if(palabra[a+1].equals(")") || palabra[a+1].equals(",")){
                    tam+=1;
                }
                a+=1;
            }
            tamanio = new int[tam]; int c=0; int t=0; //declaración de variables
            while(!palabra[index].equals(")")){ //Mientras que en la consulta del cliente no se leea un )
                if(palabra[index+1].equals(",") || palabra[index+1].equals(")")){
                    if(convertirInt(palabra[index])){
                        int valor = Integer.valueOf(palabra[index]);
                        tamanio[t] = valor;
                        t+=1;
                    }
                    else{
                        return -2;
                    }
                }
                index+=1;
            }
            cant+=1;
            break;
case "/*col,col*/": //Se indica las columnas de una tabla
    ubicacion = a;
    index = a; //Le damos a indix el mismo valor de a
    if(i == 4){
        while(!palabra[a].equals(")")){ //Ponemos la cantidad de columnas que tenemos
            if(palabra[a+1].equals(")") || palabra[a+1].equals(",")){ //Comparamos si es igual a una , o )
                col +=1; //Guardamos la cantidad de columnas que se encontraron
            }
            a+=1;
        }
        columna = new String[col]; //Le asigamos a un vector la cantidad de columnas que tenemos
        c=0; t=0;
        while(!palabra[index].equals(")")){ //Mientras que en la consulta del cliente no se leea un )
            if(palabra[index+1].equals(")") || palabra[index+1].equals(",") ){ //Preguntamos la palabra es igual , o )
                columna[c] = palabra[index]; //Guardamos el nombre de la columna
                c+=1;
            }
            index+=1;
        }
        cant+=1;
    }
    if(i == 2){
        while(!palabra[a].equals(")")){ //Ponemos la cantidad de columnas que tenemos
            if(palabra[a+1].equals(":")){ //Comparamos si es igual a una , o )
                col +=1; //Guardamos la cantidad de columnas que se encontraron
            }
            a+=1;
        }
        columna = new String[col]; //Le asigamos a un vector la cantidad de columnas que tenemos
        c=0; t=0;
        while(!palabra[index].equals(")")){ //Mientras que en la consulta del cliente no se leea un )
```

```
                while(!palabra[index].equals(")")){ //Mientras que en la consulta del cliente no se leea un )
                    if(palabra[index+1].equals(":")){ //Preguntamos la palabra es igual , o )
                        columna[c] = palabra[index]; //Guardamos el nombre de la columna
                        c+=1;
                    }
                    index+=1;
                }
                cant+=2;
            }
            break;
        case "/*nombreDeBaseDeDatos*/": //Indicamos el nombre de una base de datos
            bdd = palabra[a]; //Lo guardamos en una variable global
            cant+=1;
            a+=1;
            break;
        case ";": //Si se lee un ; se termino la sentencia
            System.out.println("Se termino la sentencia " + i);
            return i; //Mandamos el numero de la sintaxis
        default:
            if(pal[cant].equals(palabra[a])){
                //System.out.println("Entro al if de pal " + pal[cant] + " y de palabra " + palabra[a]);
                cant+=1;
                a+=1;
            }
            else{
                //System.out.println("No entro al if de pal " + pal[cant] + " y de palabra " + palabra[a]);
                cant = palabra.length; //La palabra no coinsiden pasamos al siguiente registro
            }
            break;
    }
  }
}
```

Lo que se hace acá es validar las palabras clase de nuestra sentencia como lo sería ***create***, ***select***, ***drop***, ***insert*** o ***select.*** Cuando llegamos a los datos que son ingresados por nuestro usuario, entramos según el case que corresponde y guardamos el dato. Devolvemos el número de renglón de la sentencia leída y nos vamos al método ***validarSentencia***. A partir de el número de renglón realizamos la acción:

1. crear usuario
2. Crear tabla
3. crear base de datos
4. insertar registro
5. visualizar tabla
6. Eliminar tabla

```
public String validarSentencia(String consulta[]){
    try {
        int i = consulta.length-1; //Guardamos el tamaño del vector
        if((consulta[i]).equals(';')){ //Preguntamos si es igual a un ;
            return "Se esperaba un ;"; //Sino lo es la sentencia no es validad
        }
        cantidadRegistro();
        int sent = separarPalabras(consulta); //Separamos la consulta del cliente
        Usuario oUser = new Usuario();
        boolean usuario;
        System.out.println("Numero de sentencia " + sent);
        switch(sent){ //Sent guarda el numero de la sintaxis
            case 0: //El 0 se conecta un usuario
                /*NO SE USA*/
                break;
            case 1: //El 1 Se crea un usuario
                usuario = oUser.validarUsuario(nomUser);
                if(usuario == false){
                    String msg = oUser.nuevoRegistro(nomUser, password);
                    return msg;
                }
                return "El nombre del usuario no es valido, eliga otro";
```

```
    case 2: //El 2 crea tabla en una base de datos
        usuario = oUser.validarUsuario(nomUser);
        if(usuario){
            Tabla oTabla = new Tabla(nomUser);
            String file = bdd + "/" + tabla + ".txt";
            oTabla.crearTabla(columna, tamanio, file );
            columna = null;
            tamanio = null;
        }
        else{
            return "No existe el usuario indicado";
        }
        break;
    case 3: //El 3 crea una base de datos
        usuario = oUser.validarUsuario(nomUser);
        if(usuario){
            String dic = "Documentos/Usuario/" + nomUser + "/" + bdd;
            if(oUser.crearCarpeta(dic)!=3){
                return "Se creo la base de datos con exito!";
            }
            else{
                return "No puede haber base de datos con el mismo nombre";
            }
        }
        break;
```

Clase de Usuario

Esta clase se comunica con **Archivo** quien se encarga de la creación y eliminación de los archivos .txt

```
public class Usuario {
    private int carac[] = new int[4];
    private String campo[] = new String[4];
    private String nombreFile;
    private String directorio;
    public String regUsuario[] = null;

    public Usuario() throws IOException{
        //Cantidad de caracteres
        carac[0] = 6;   //id
        carac[1] = 20; //nombre
        carac[2] = 25; //contraseña
        carac[3] = 25; //fecha

        //llenamos los campos
       for(int i = 0; i < campo.length; i++){
           campo[i] = "";
       }
        directorio = "Documentos/Administrado/";
        nombreFile = "Usuario.txt";
    }
    public String nuevoRegistro(String nomUsuario, String password) throws IOException{
        Archivo oArch = new Archivo();
        campo[1] = nomUsuario;
        campo[2] = password;
        campo[3] = obtenerFecha();


        oArch.nombreFile = nombreFile;
        oArch.directorio = directorio;
        oArch.crearDirectorio();
```

```
public String nuevoRegistro(String nomUsuario, String password) throws IOException{
    Archivo oArch = new Archivo();
    campo[1] = nomUsuario;
    campo[2] = password;
    campo[3] = obtenerFecha();


    oArch.nombreFile = nombreFile;
    oArch.directorio = directorio;
    oArch.crearDirectorio();
    boolean val = oArch.nuevoRegistro(campo, carac);
    if(val){
        String direccion = "Documentos/Usuario/";
        oArch.directorio = direccion + nomUsuario;
        oArch.crearDirectorio();
        return "Se creo su Usuario";
    }
    return "No se pudo realizar la sentencia";
}

public boolean validarUsuario(String nombre){
    if(!nombre.equals(" ")){
        Archivo oArch = new Archivo();
        oArch.nombreFile = directorio + nombreFile;
        int val = oArch.buscarCampo(nombre, carac);
        switch(val){
            case -1:
                return false;
            case 0:
                return false;
            case 1:
                return true;
        }
    }
    return false;
}
public int crearCarpeta(String dic){
    Archivo oArch = new Archivo();
    oArch.directorio = dic;
    int val = oArch.crearDirectorio();
    return val;
}
public String obtenerFecha(){
    DateTimeFormatter fecha = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm");
    String sFecha = fecha.format(LocalDateTime.now());
    return sFecha;
}
```

Clase Tabla

Su objetivo es crear las tablas de los usuarios y realizar las funciones que el cliente cómo añadir registros, mostrar tabla y eliminar la misma.

```
public class Tabla {
    String directorio = "Documentos/Usuario/";
    public Tabla(String nomUser){
        directorio = directorio + nomUser;
        System.out.println("La dirección del archivo es: " + directorio);
    };

    public boolean crearTabla(String[] campo, int[] carac, String file){
        Archivo oArch = new Archivo();
        oArch.nombreFile = directorio + "/" + file;
        System.out.println("Direccion de carpeta " + oArch.nombreFile);
        boolean val = oArch.crearTabla(campo, carac);
        System.out.println("La creación de la tabla es: "+val );
        return val;
    }
```

```
public String validarCampos(String[] campos, String file){
    Archivo oArch = new Archivo();
    oArch.nombreFile = directorio + "/" + file;
    String columna="";
    int cant = oArch.obtenerColumnas();
    if(cant == -1){
        return "La tabla no existe";
    }
    else{
        if(!(campos.length > cant)){
            boolean val = false;
            int[] entidad = oArch.obtenerTamaño(cant);
            for(int i=0; i < campos.length; i++){
                if(campos[i].length() < entidad[i]){
                    columna = campos[i];
                    val = true;
                }
            }
            if(!val){
                System.out.println("Error con : " + columna);
                return "sobrepasa el limite de la columna";
            }
            oArch.espacio = true;
            oArch.nuevoRegistro(campos, entidad);
        }
        else{
            return "Hay campos demás";
        }
    }
    return "";
}
```

```
public String verTabla(String file){
    directorio = directorio + "/" + file;
    System.out.println("directorio " + directorio);
    Archivo oArch = new Archivo();
    oArch.nombreFile = directorio;
    return oArch.verTabla();
}
public String elimnarTabla(String file){
    directorio = directorio + "/" + file;
    System.out.println("directorio " + directorio);
    Archivo oArch = new Archivo();
    oArch.nombreFile = directorio;
    return oArch.eliminarTabla();
}
```

Clase Archivo

Es la clase central que se encarga de todas las funciones del servidor.

```
public class Archivo {
    public String directorio = "";
    public String nombreFile = "";
    public boolean crearCarpeta = false;
    public boolean espacio = false;
```

```java
public boolean nuevoRegistro(String[] vcampos, int[] n){
    //Instaciamos las siguiente clases para escribir en el archivo
    FileWriter fw = null; BufferedWriter bw = null;
    try{
        //Le asignamos al campo[0] una id
        vcampos[0] = nuevaId(vcampos, n);
        System.out.println(vcampos[0]);
        String carpeta = directorio+nombreFile;
        //instanciamos la clase e indicamos true para que no se sobreescriba lo que ingresamos
        fw = new FileWriter(carpeta, true);
        bw = new BufferedWriter(fw);
        //Realizamos un for para escribir los datos ingresados
        for(int i = 0; i < vcampos.length; i++){
            System.out.println("Campos: " + vcampos[i]);
            //Le asignamos un limite de caracteres a cada campo
            String carac = "%-"+String.valueOf(n[i])+"s";
            //Le asignamos nuevamente al vector de campo ya con su limite de caracteres
            vcampos[i] = String.format(carac, vcampos[i]);
            //Lo escribimos en el archivo txt
            bw.write(vcampos[i]);
        }
        if(espacio){
            bw.write("\n");
        }
        //Cerramos la clase
        bw.close(); fw.close();
        return true;
    }catch(Exception e){
        e.printStackTrace();
    }return false;
}
```

```java
public String nuevaId(String[] campos, int [] n){
    FileReader fr;

    try{
        String carpeta = directorio + nombreFile;
        File arch = new File(carpeta);
        if(!arch.exists()){ //Si no exite el archivo lo crea
            arch.createNewFile();
        }
        if(arch.length() == 0){ //si el archivo se encuntra vacio
            return "1"; //devolvemos 1, ya que serìa nuestra primera id
        }
        else{
            fr = new FileReader(carpeta);

            int reader = fr.read(); //Se lee todo el archivo
            int espacio = 0; //Se guardara el espacio que ocupa cada registro
            String id = ""; //Se guardara la id del registro
            int repeticiones=0; //Contara cada caracter que se lee
            for(int i = 0; i< n.length; i++){
                espacio = espacio + n[i]; //Guardamos el espacio de cada registro
            }
            String txt = ""; char caracter;
```

```
            while(reader != -1){ //Si llega a -1 quiere decir que llego al final del archivo
                caracter = (char)reader;
                txt = txt + String.valueOf(caracter);
                reader = fr.read();
                repeticiones+=1;
                if(repeticiones == n[0]){ //si la repeticio es igual n[0](limite de caractere de id)
                    id = txt; //Se guarda la id
                    System.out.println("dentro del if: "+id);
                }//Si repeticiones es igual espacio significa que se leyo un registro de dato completo
                if(repeticiones == espacio){
                    System.out.println(txt);
                    txt=""; //Vaciamos la variable para leer el siguiente registro
                    repeticiones = 0; //declaramos de nuevo 0
                }
            }
            System.out.println("afuera del if: "+id);
            if(convertirInt(id.trim())){ //Si se puede convertir en un int
                return String.valueOf(Integer.valueOf(id.trim())+1);
            }
            else{ //Si no puede esta leyendo una entidad
                return "1"; //devuelve 1
            }
        }
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return "nada";
}
```

```
public int buscarCampo(String campo, int[] n){
    FileReader fr = null;
    BufferedReader br = null;
    try{
        File arch = new File(nombreFile);
        if(!arch.exists()){ //Si el archivo no existe
            return -1;
        }
        fr = new FileReader(nombreFile);
        br = new BufferedReader(fr);
        String lectura; int espacio = 0;
        for(int i=0; i < n.length; i++){ espacio = espacio + n[i];}
        lectura = br.readLine(); //Leamos el archivo

            int cant = lectura.length()/espacio;
            String[] registro = new String[cant];
            //Separa los registros
            for(int i = 0; i < registro.length; i++){
                registro[i]= lectura.substring(0, espacio);
                lectura = lectura.substring(espacio);
            }
            for(int i=0; i< registro.length; i++){
                String reg = registro[i]; //guardamos el registro en una variable
```

```
            for(int i=0; i< registro.length; i++){
                String reg = registro[i]; //guardamos el registro en una variable

                for(int x=0; x < n.length; x++){
                    String leer = reg.substring(0, n[x]); //leemos el campo segun el tamaño de la "tabla"
                    reg = reg.substring(n[x]); //eliminamos el tamaño del campo
                    if((leer.trim()).equals(campo)){
                        return 1;
                    }
                }
            }
        return 0;
    }
    catch(Exception e){
        e.printStackTrace();
    }
        return 0;
}
```

```
public int crearDirectorio(){
    File carpeta = new File(directorio); //Pasamos la dirección de nuestro directorio
    if (carpeta.exists()==false) { //si las carpeta no exiten
        if (carpeta.mkdirs()) {
            System.out.println("Directorio creado");
            return 1;
        }
        else {
            System.out.println("Error al crear directorio");
            return 2;
        }
    }
    else{
        System.out.println("La carpeta existe");
        return 3;
    }
}
```

```java
public boolean crearTabla(String[] vcampos, int[] n){
    try{
        File arch = new File(nombreFile);
        if(!arch.exists()){ //Si no exite el archivo lo crea
            arch.createNewFile();
        }
        FileWriter fw = new FileWriter(nombreFile, true);
        BufferedWriter bw = new BufferedWriter(fw);
        //Realizamos un for para escribir los datos ingresados
        for(int i = 0; i < vcampos.length; i++){
            System.out.println("Campos: " + vcampos[i]);
            //Le asignamos un limite de caracteres a cada campo
            String carac = "%-"+String.valueOf(n[i])+"s";
            //Le asignamos nuevamente al vector de campo ya con su limite de caracteres
            vcampos[i] = String.format(carac, vcampos[i]);
            //Lo escribimos en el archivo txt
            bw.write(vcampos[i]);
        }
        bw.write("\n");
        //Cerramos la clase
        bw.close();
        fw.close();
        return true;
    }
    catch(Exception e){
        e.printStackTrace();
        System.out.println("No se pudo crear la tabla");
    }
    return false;
}
```

```
public int obtenerColumnas(){
    try{
        File arch = new File(nombreFile);
        if(!arch.exists()){ //La tabla no existe
            return -1;
        }
        FileReader fr = new FileReader(nombreFile);
        BufferedReader br = new BufferedReader(fr);
        String txt = ""; char caracter;
        String linea = br.readLine();
        System.out.println(linea);
        String[] columna = linea.split("\\s+");

        return columna.length;
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return -1;
}
```

```
public int[] obtenerTamaño(int tamanio){
    int[] colTamanio = new int[tamanio];
    try{
        File arch = new File(nombreFile);
        FileReader fr = new FileReader(nombreFile);
        BufferedReader br = new BufferedReader(fr);

        String linea = br.readLine();
        String txt = ""; int index = 0;
        int i;
```

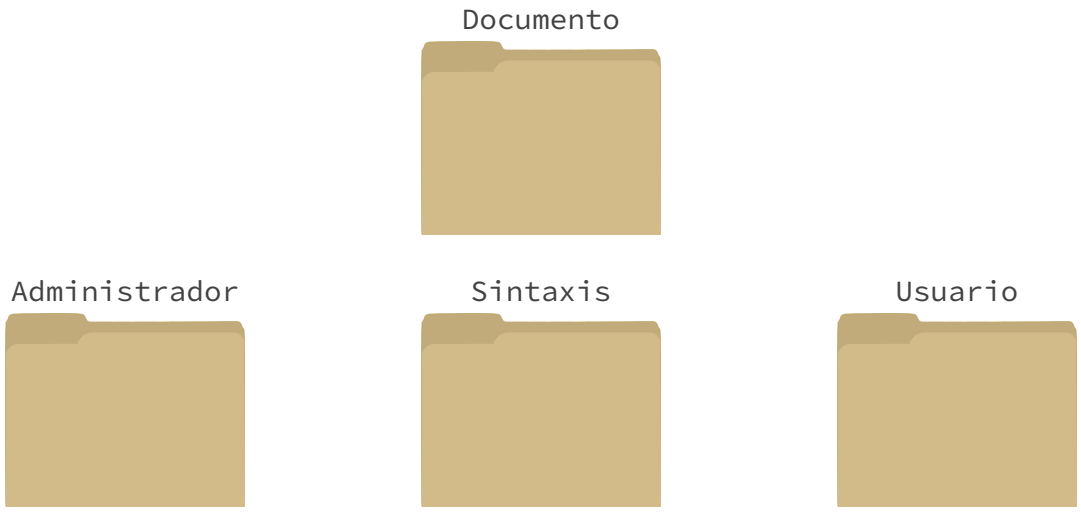
```
public int[] obtenerTamaño(int tamanio){
    int[] colTamanio = new int[tamanio];
    try{
        File arch = new File(nombreFile);
        FileReader fr = new FileReader(nombreFile);
        BufferedReader br = new BufferedReader(fr);

        String linea = br.readLine();
        String txt = ""; int index = 0;
        int i;
        for (i=0; i < linea.length()-1; i++) {
            if(linea.charAt(i) == 32){ //Si se encuentra un espacio
                //Verificamos que encuentre otro espacio
                if(linea.charAt(i+1) == 32 || (linea.charAt(i) >= 65 && linea.charAt(i) <= 90)){
                    txt = txt + String.valueOf(linea.charAt(i));
                }
                else{
                    if(index <= colTamanio.length){
                        colTamanio[index] = txt.length()+1;
                        txt = "";
                    }
                    index+=1;
                }
            }
            else if((linea.charAt(i) >= 65 && linea.charAt(i) <= 90)){ //Si se encuentra una letra
                //Verificamos que encuentre otro letra
                if((linea.charAt(i+1) >= 65 && linea.charAt(i+1) <= 90) || linea.charAt(i+1) == 32){
                    txt = txt + String.valueOf(linea.charAt(i)); //La guardamos
                }
            }
        }
```

```
            colTamanio[index] = txt.length()+1;
            return colTamanio;
        }
        catch(Exception e){
            e.printStackTrace();
        }
        return null;
    }

    private boolean convertirInt(String cadena) {
        boolean resultado;
        try {
            Integer.parseInt(cadena);
            resultado = true;
        } catch (NumberFormatException excepcion) {
            resultado = false;
        }
        return resultado;
    }

    public String verTabla(){
        try{
            File arch = new File(nombreFile);
            if(!arch.exists()){ //La tabla no existe
                return "la tabla no existe"; //No se puede visualizar ninguna tabla
            }
            FileReader fw = new FileReader(nombreFile);
            BufferedReader bw = new BufferedReader(fw);

            String contenido; String tabla = "";

            while((contenido = bw.readLine()) != null){
                tabla = tabla + contenido + "\n";
            }
            //Cerramos la clase
            bw.close();
            fw.close();
            return tabla;
        }
        catch(Exception e){
            e.printStackTrace();
            System.out.println("No se pudo crear la tabla");
        }
        return "ocurrio un error";
    }
```

```
public String eliminarTabla(){
    try{
        File archivo = new File(nombreFile);
        boolean estatus = archivo.delete();
        if (!estatus) {
            return "Error no se ha podido eliminar el  archivo";
        }
        else{
            return "Se ha eliminado la Tabla exitosamente";
        }
    }
    catch(Exception e){

    }
    return "no se encontro la Tabla";
}
```

Arquitectura

La arquitectura se basa en 4 carpetas determinadas.



Tenemos 3 niveles:

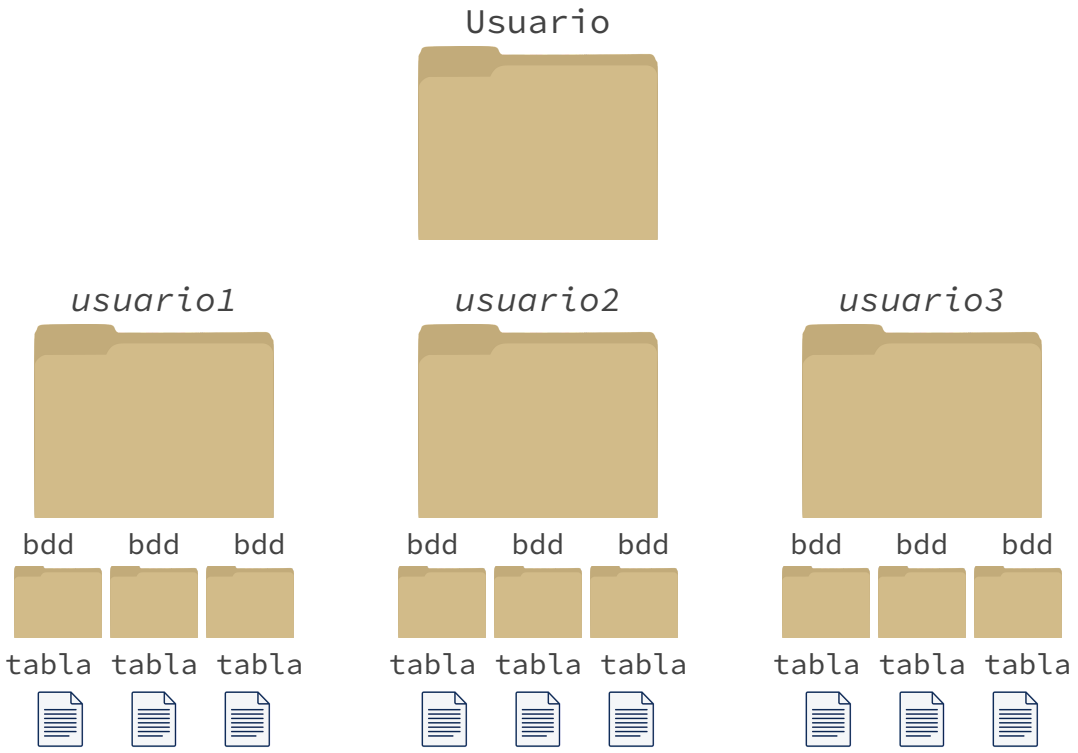
Nivel 1:

- Documentos : Es nuestra carpeta *“padre”* que guarda toda nuestra arquitectura del aplicativo.

Nivel 2:

Se encuentra la carpeta documentos que contiene

- **Administrador** : se guarda una archivo .txt que guarda los datos de los usuario registrado
- **Sintaxis**: en ella se encuentra un archivo *sentencia.txt* en donde se halla las sintaxis que se utilizan en nuestra base.
- **Usuario** :



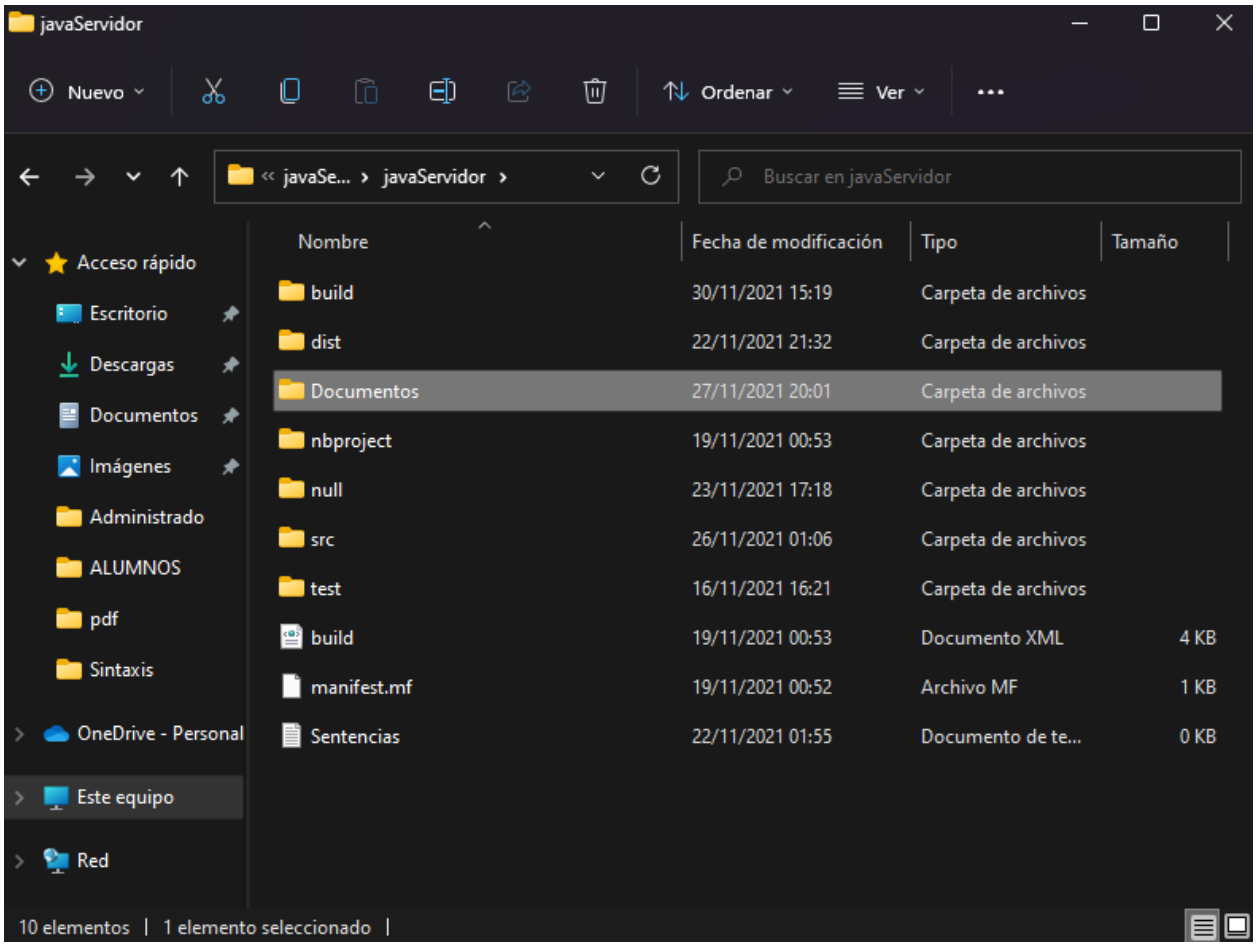
Usuario contiene las carpetas de los clientes que se registraron(*usuario1, usuario2 y usuario3 son usuarios de ejemplos*) en nuestro programa.

Dentro de las carpetas de nuestros clientes, se encuentran otras que serían las bases de datos de nuestro usuario.

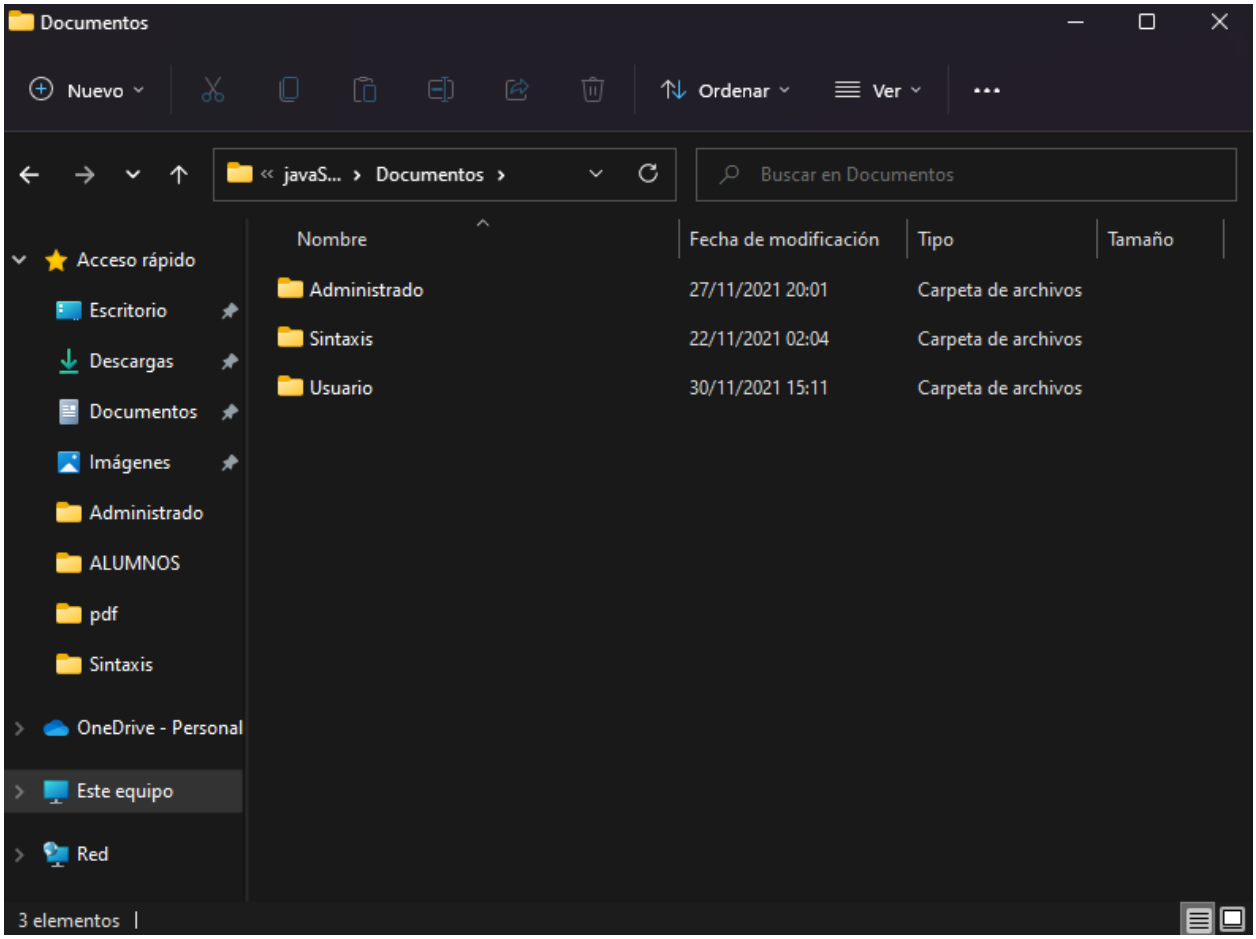
Las tablas se guardan según el usuario que realizó la consulta y la base de datos, las tablas son archivos .txt

Ejemplo

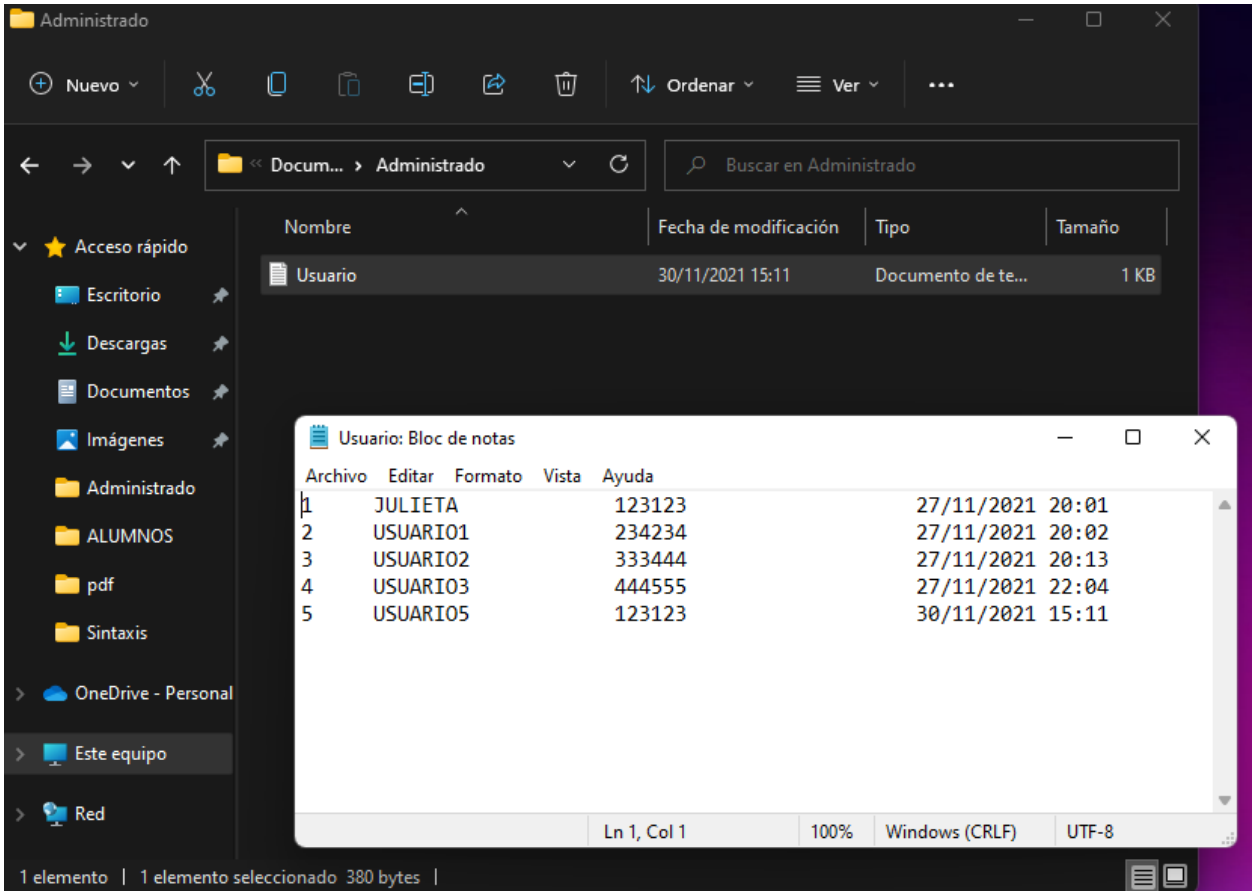
Buscamos la carpeta Documento



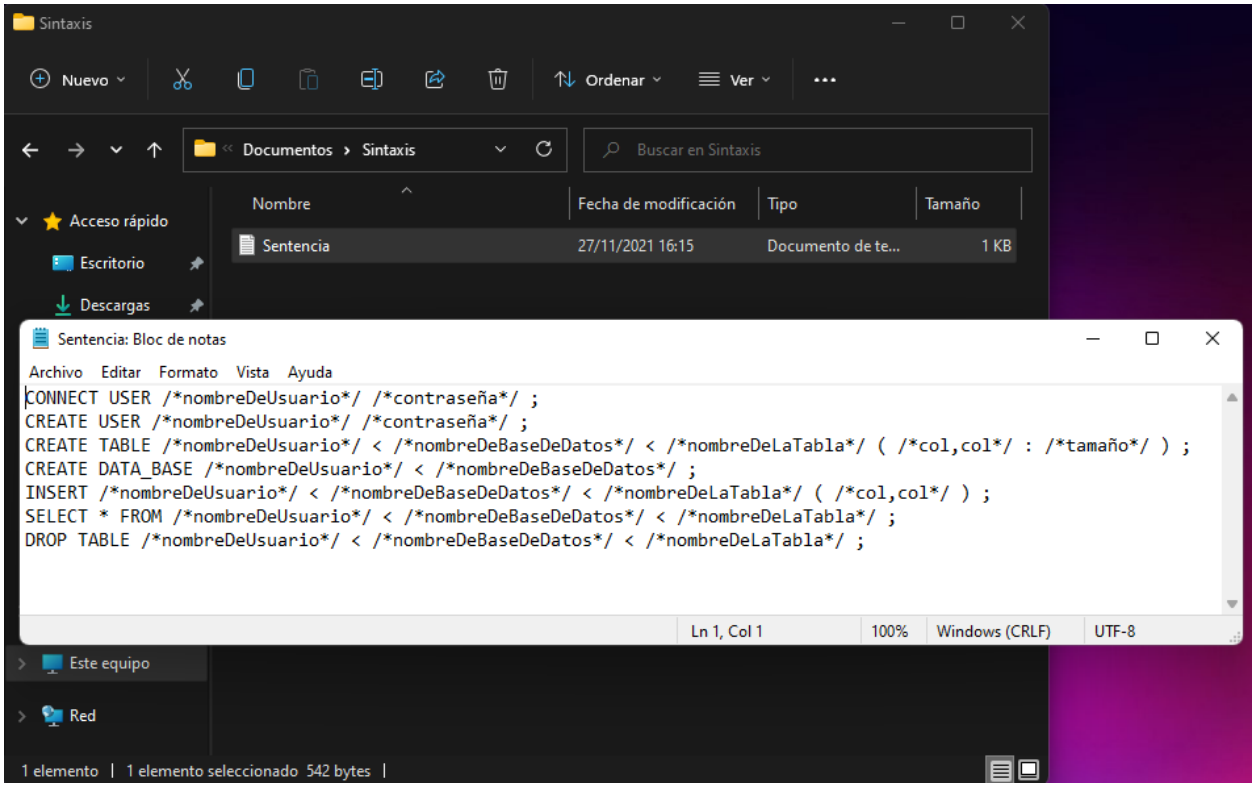
Ingresamos a ella


Documentos
Nuevo
Ordenar
Ver
javaS... › Documentos ›
Buscar en Documentos
Acceso rápido
Escritorio
Descargas
Documentos
Imágenes
Administrado
ALUMNOS
pdf
Sintaxis
OneDrive - Personal
Este equipo
Red
Nombre
Fecha de modificación
Tipo
Tamaño
Administrado
27/11/2021 20:01
Carpeta de archivos
Sintaxis
22/11/2021 02:04
Carpeta de archivos
Usuario
30/11/2021 15:11
Carpeta de archivos
3 elementos

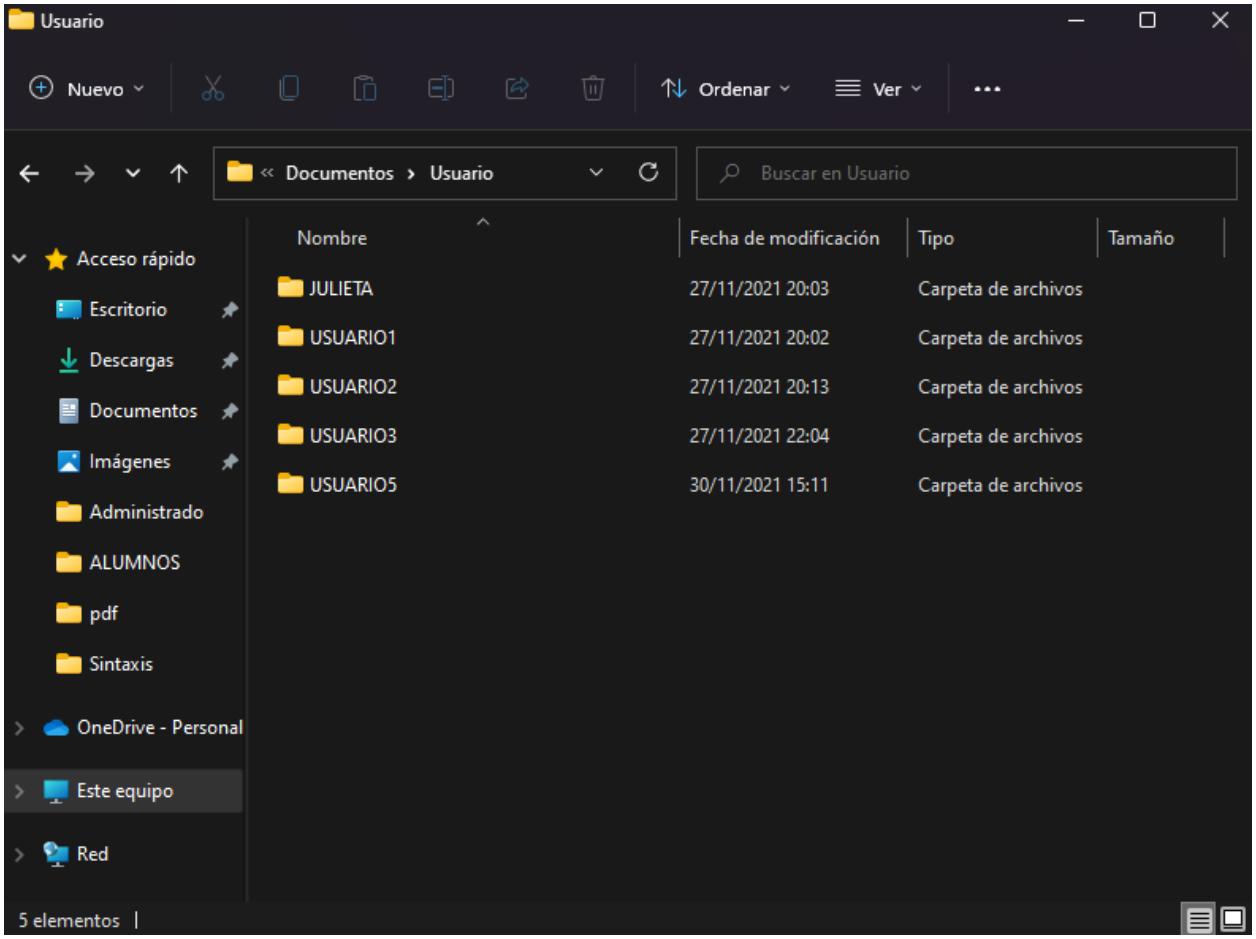
Administrador



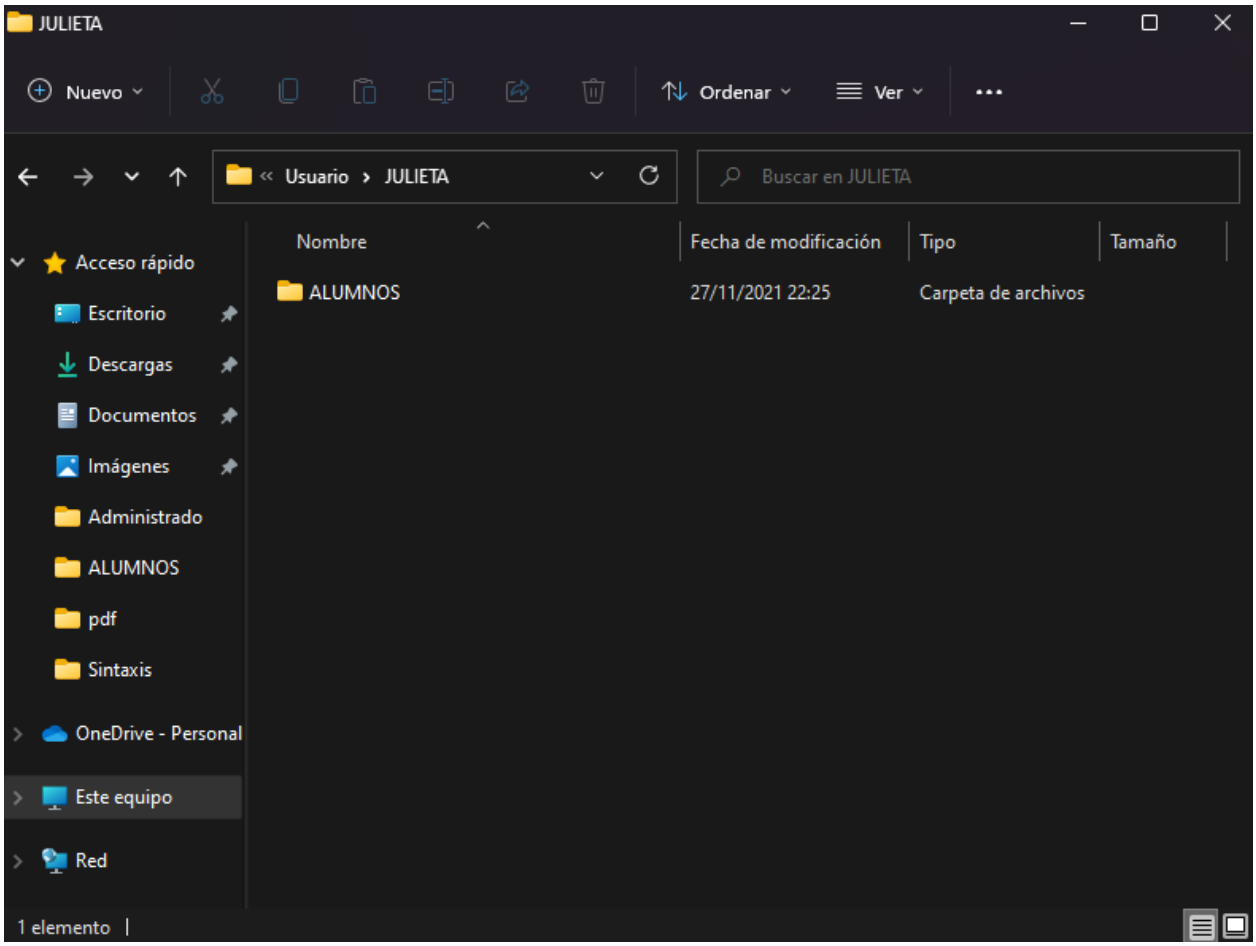
Sintaxis


Sintaxis
Nuevo
Ordenar
Ver
Documentos › Sintaxis
Buscar en Sintaxis
Nombre
Fecha de modificación
Tipo
Tamaño
Acceso rápido
Escritorio
Descargas
Sentencia
27/11/2021 16:15
Documento de te...
1 KB
Sentencia: Bloc de notas
Archivo Editar Formato Vista Ayuda
CONNECT USER /*nombreDeUsuario*/ /*contraseña*/ ;
CREATE USER /*nombreDeUsuario*/ /*contraseña*/ ;
CREATE TABLE /*nombreDeUsuario*/ < /*nombreDeBaseDeDatos*/ < /*nombreDeLaTabla*/ (/*col,col*/ : /*tamaño*/) ;
CREATE DATA_BASE /*nombreDeUsuario*/ < /*nombreDeBaseDeDatos*/ ;
INSERT /*nombreDeUsuario*/ < /*nombreDeBaseDeDatos*/ < /*nombreDeLaTabla*/ (/*col,col*/) ;
SELECT * FROM /*nombreDeUsuario*/ < /*nombreDeBaseDeDatos*/ < /*nombreDeLaTabla*/ ;
DROP TABLE /*nombreDeUsuario*/ < /*nombreDeBaseDeDatos*/ < /*nombreDeLaTabla*/ ;
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
Este equipo
Red
1 elemento | 1 elemento seleccionado 542 bytes |

Usuario


Usuario
Nuevo
Ordenar
Ver
Documentos › Usuario
Buscar en Usuario
Acceso rápido
Escritorio
Descargas
Documentos
Imágenes
Administrado
ALUMNOS
pdf
Sintaxis
OneDrive - Personal
Este equipo
Red
Nombre
Fecha de modificación
Tipo
Tamaño
JULIETA 27/11/2021 20:03 Carpeta de archivos
USUARIO1 27/11/2021 20:02 Carpeta de archivos
USUARIO2 27/11/2021 20:13 Carpeta de archivos
USUARIO3 27/11/2021 22:04 Carpeta de archivos
USUARIO5 30/11/2021 15:11 Carpeta de archivos
5 elementos

Dentro de un usuario



En una carpeta de base de datos