Técnico Superior en Desarrollo de Aplicaciones Multimedia

Curso Académico 2021/2022 - 2º DAMS

Manejo de ficheros mediante lenguaje de programación

Unidad 1: Introducción al acceso a datos y manejo de ficheros

Alumno: José Enrique Jordán Moreno

Tabla de contenido

TABLA CON LOS NOMBRES DE BASES DE DATOS 2
EJERCICIO 1. 2
DESCARGA E INSTALACIÓN DE MONGODB: 2
EJERCICIO 2 9
FORMULARIO GUARDAR DATOS 9
PROCEDIMIENTO GUARDAR 10
PROCEDIMIENTO GUARDARARCHIVO 11
PROCEDIMIENTO PARA MOSTRAR LOS DATOS DE LOS ALUMNOS EN UN JTABLE. 12
EJERCICIO 3 13
PROCEDIMIENTO CONTADOR PARA CONTAR LOS REGISTROS 13

GESTOR De BBDD	Sistema Gestor BBDD Relacional	Soporta SQL	Soporta Procedimientos Almacenados	Soporta Transacciones	Multiplataforma	Multiusuario
MySQL	SI	SI	SI	SI	SI	SI
SQLServer	SI	SI	SI	SI	NO	SI
Oracle	SI	SI	SI	SI	NO	SI
Microsoft Access	SI	SI	SI	SI	NO	SI
PostgreSQL	SI	SI	SI	SI	SI	SI
Mongo DB	NO BBDD NoSQL	NO	NO	SI	SI	SI
Adabas	NO BBDD de listas invertidas	SI	SI	SI	SI	SI

TABLA CON LOS NOMBRES DE BASES DE DATOS

EJERCICIO 1.

DESCARGA E INSTALACIÓN DE MONGODB:

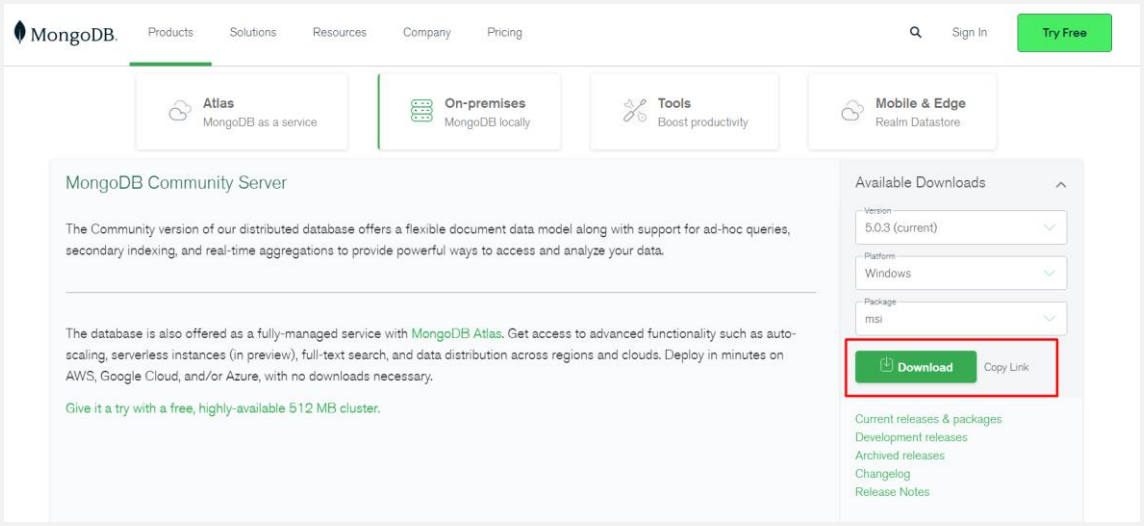
Para la descarga podemos usar estos enlaces de la web oficial:

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

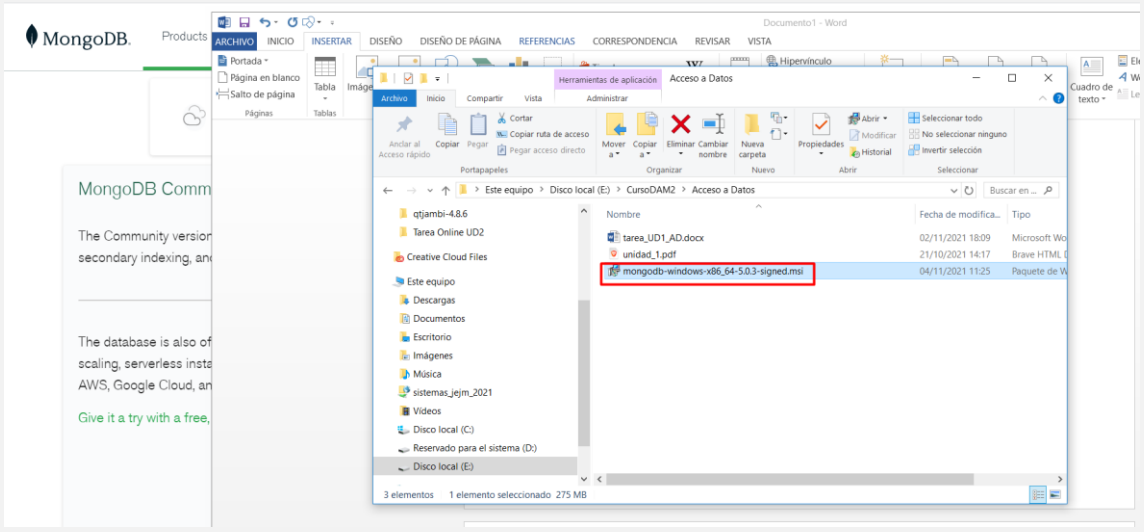
https://www.mongodb.com/try/download/community?tck=docs_server

Enlaces de Recursos utilizados:

https://www.youtube.com/watch?v=OTE5sL1xi18&t=7s

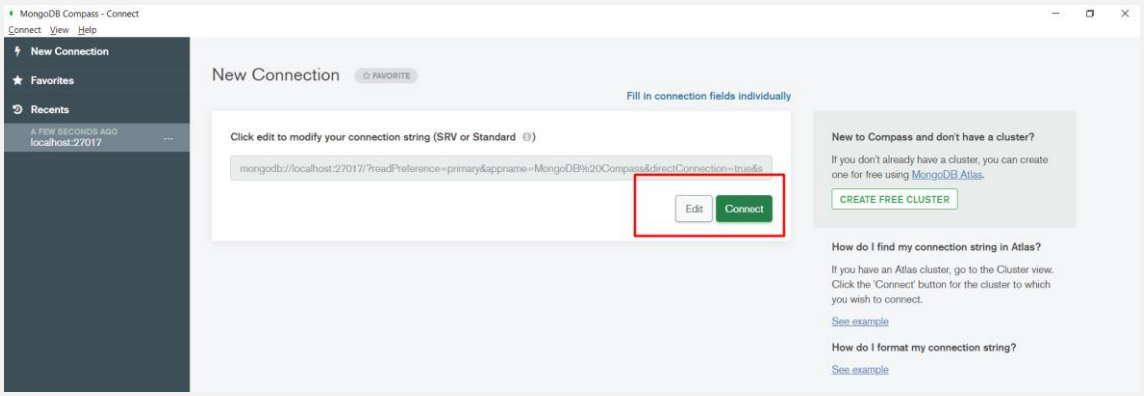


Página oficial de **MongoDB** versión **community**, descarga desde "**Download**"

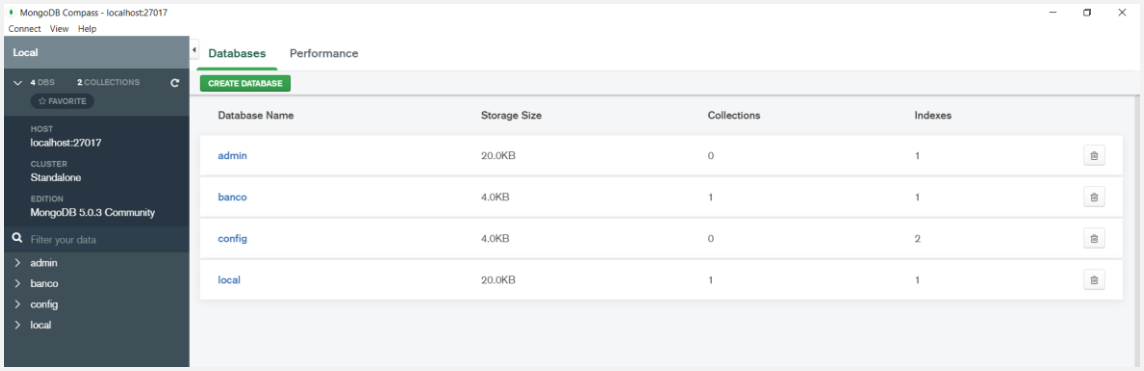


EL FICHERO DESCARGADO ES: MONGODB-WINDOWS-X86_64-5.0.3-SIGNED.MSI

Una vez descargado procederemos a la instalación por defecto, nos instalará **MongoDB** y el entorno gráfico **Compass**.

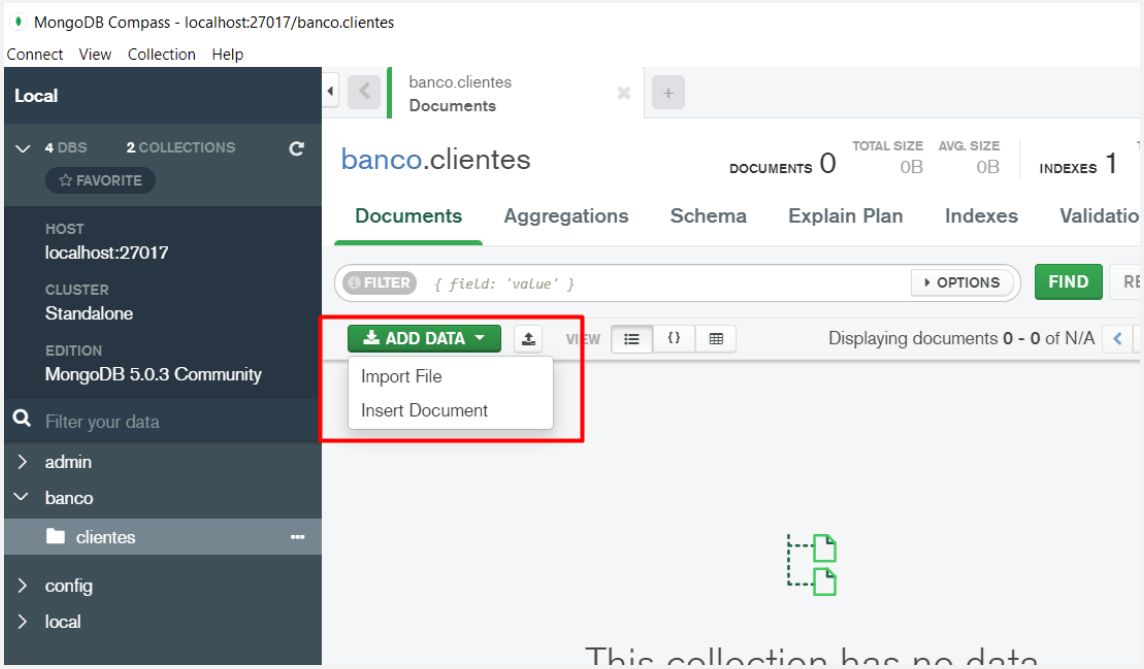


Después de la instalación ejecutamos **MongoDB Compass** , el gestor visual de **MongoDB**. Pulsaremos **Connect** para iniciar una nueva Conexión.



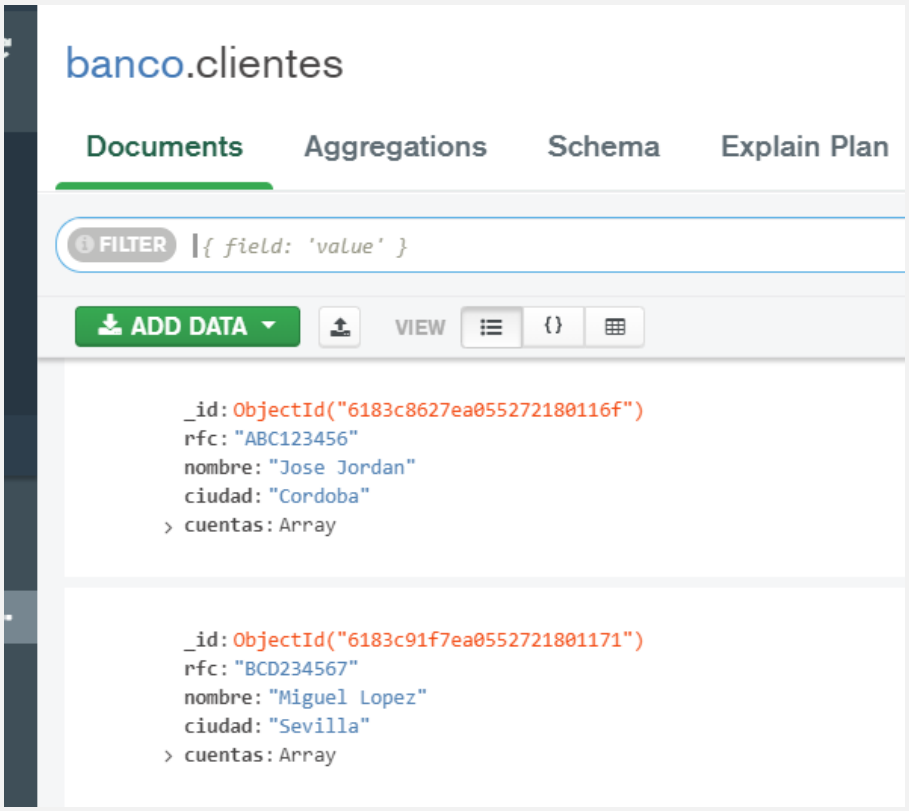
Pulsamos en **Create database** y le damos un nombre a database y collection

Para el ejemplo crearemos una database con el nombre banco y collection name clientes.

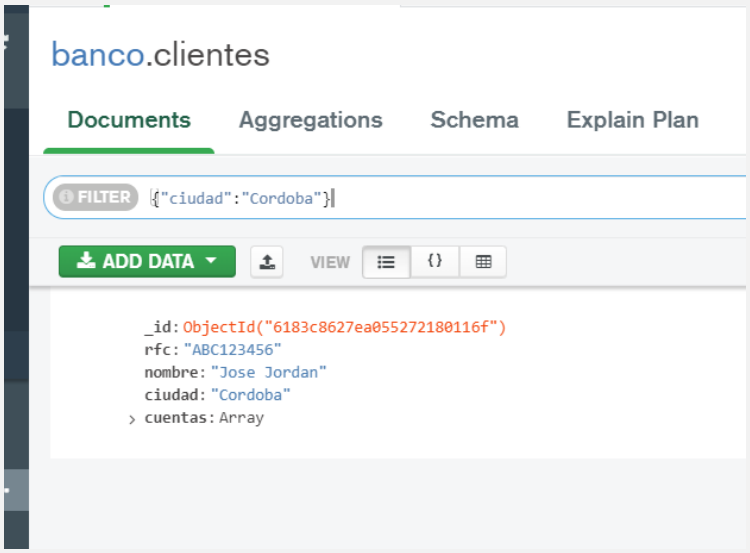


Pulsamos en **banco.clientes** e Insertamos un documento en Insert Document.

Colección Clientes seria como la tabla Clientes en SQL.

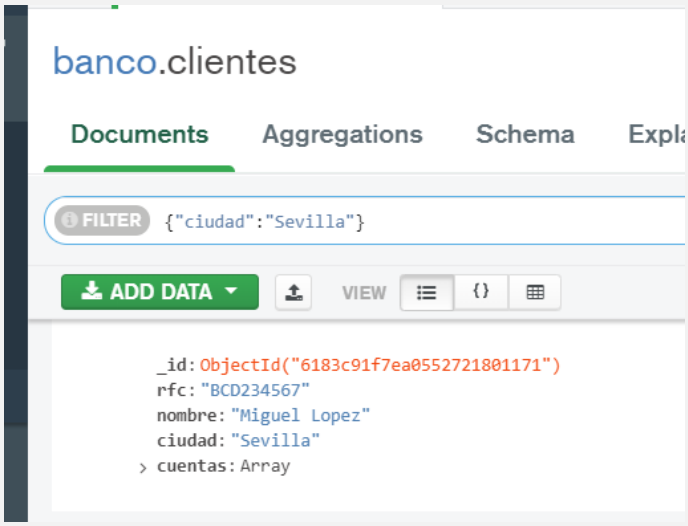


Añadimos algunos datos de ejemplo.



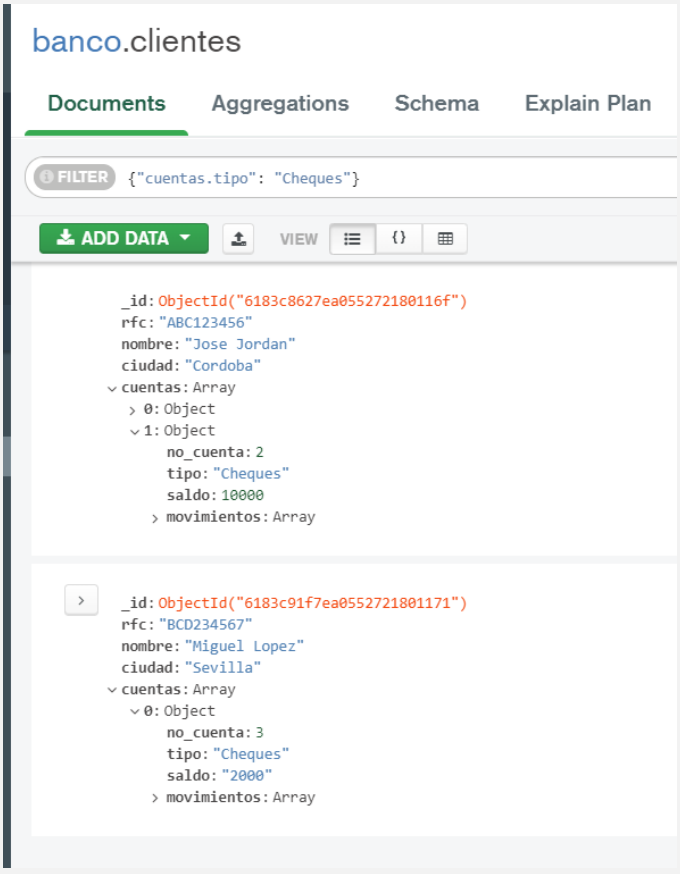
Para las consultas en filter entre llaves abrimos comillas indicamos nuestra **consulta :**

{"ciudad":"Cordoba"}



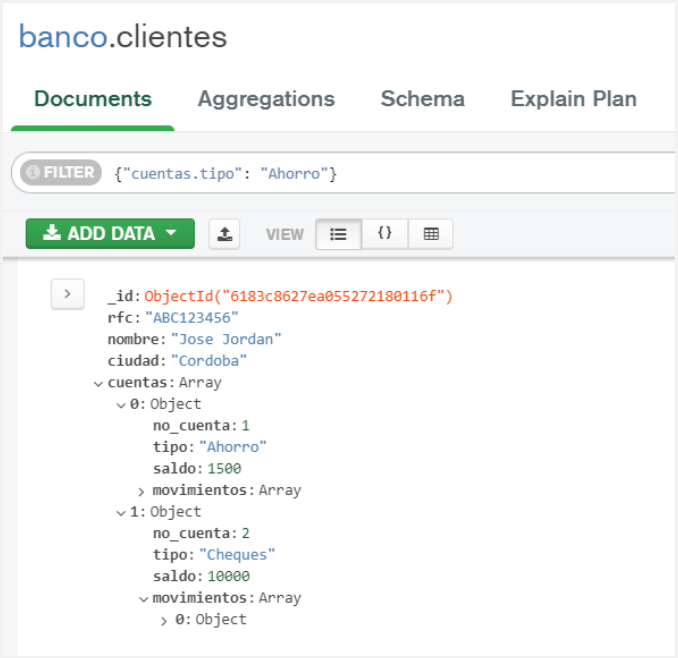
{"ciudad":"Sevilla"}

De esta forma nos filtra los registros y nos muestra la ciudad indicada.



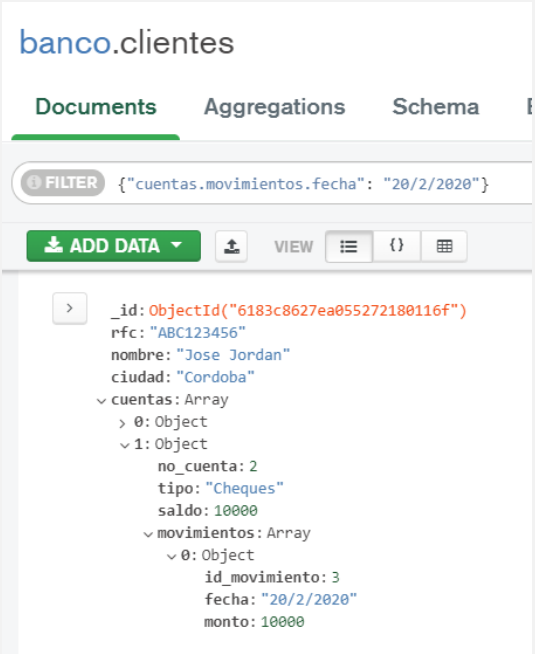
En esta consulta buscamos los clientes que tienen cuentas tipo cheques

{"cuentas.tipo": "Cheques"}



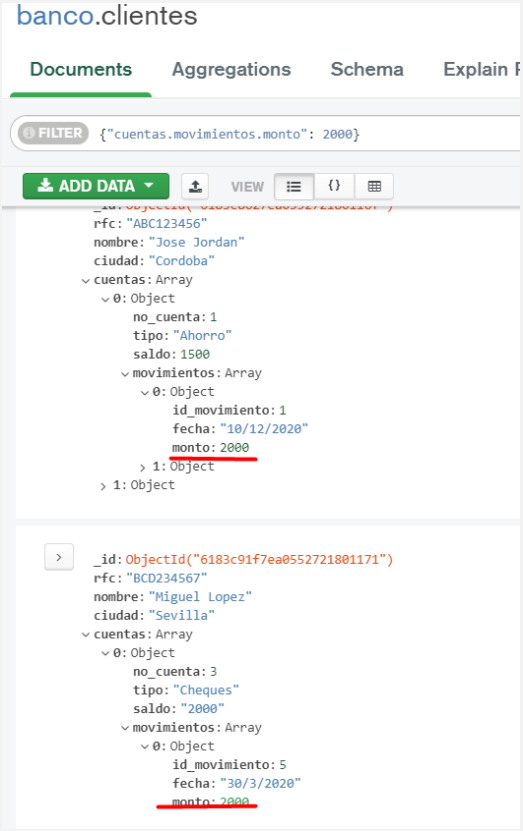
En la siguiente consulta buscamos los clientes que tienen cuentas de ahorro.

{"cuentas.tipo": "Cheques"}



{"cuentas.movimientos.fecha": "20/2/2020"}

Aquí buscamos movimientos en cuentas con una fecha determinada.

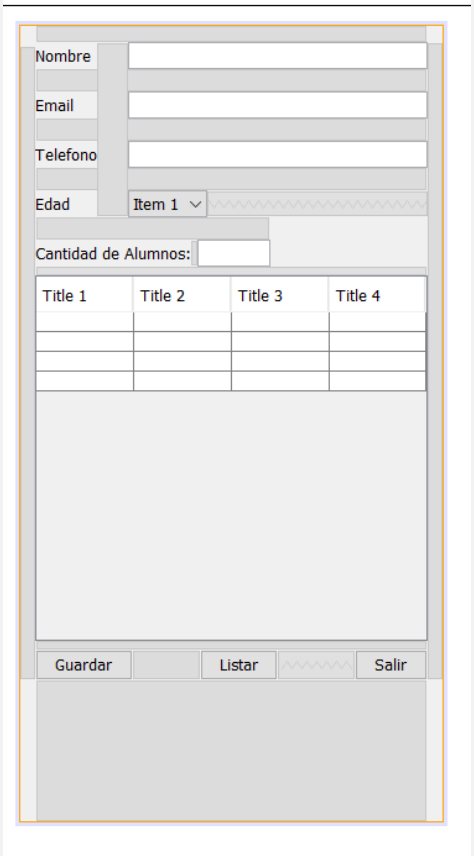


{"cuentas.movimientos.monto": 2000}

Con esta consulta buscamos los movimientos en cuentas con un monto de 2000

EJERCICIO 2

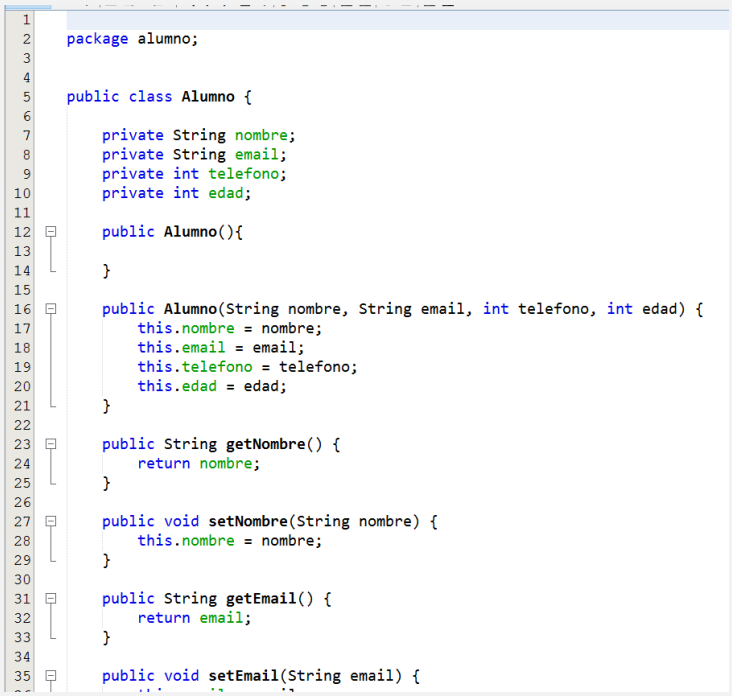
Para esta actividad he creado un proyecto en Netbeans llamado **GuardarDatos**. El proyecto se compone de una clase Alumno , una clase Métodos y un formulario **frmGuardarDatos**.



FORMULARIO GUARDAR DATOS

El formulario está compuesto de varias etiquetas, campos de texto para nombre, email , teléfono y spinner para la edad , todos ellos los utilizaremos para ir introduciendo los datos de los alumnos.

También tenemos el botón guardar, que nos guardará los datos en un fichero **"alumnos.dat"** , el botón listar que nos listará los datos de los alumnos en una tabla, y el botón salir que cerrará la aplicación.



```java
package alumno;


public class Alumno {

    private String nombre;
    private String email;
    private int telefono;
    private int edad;

    public Alumno(){

    }

    public Alumno(String nombre, String email, int telefono, int edad) {
        this.nombre = nombre;
        this.email = email;
        this.telefono = telefono;
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
```

La clase Alumno se utilizará para recoger los datos del alumno

```java
    private void jButton_GuardarActionPerformed(java.awt.event.ActionEvent evt) {

        String nombre = txtNombre.getText();
        String email = txtEmail.getText();
        int telefono = Integer.parseInt(txtTelefono.getText().toString());
        int edad = Integer.parseInt(jComboBox_Edad.getSelectedItem().toString());

        alumno.setNombre(nombre);
        alumno.setEmail(email);
        alumno.setTelefono(telefono);
        alumno.setEdad(edad);
        metodos.guardar(alumno);
        metodos.guardarArchivo(alumno);


    }
```

PROCEDIMIENTO GUARDAR

El botón **Guardar** recoge los datos del alumno y mediante los métodos guardar y **guardararchivo** se guardar los datos en un fichero

```java
//PROCEDIMIENTO PARA GUARDAR EN UN ARCHIVO
public void guardarArchivo(Alumno alumno) {
    try {
        FileWriter fw = new FileWriter("alumnos.dat", true);
        BufferedWriter bw = new BufferedWriter(fw);
        PrintWriter pw = new PrintWriter(bw);
        pw.print(alumno.getNombre());
        pw.print("|" + alumno.getEmail());
        pw.print("|" + alumno.getTelefono());
        pw.println("|" + alumno.getEdad());
        pw.close();

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
}
```

PROCEDIMIENTO GUARDARARCHIVO

El procedimiento **guardarArchivo** recoge como parámetro alumnos.

El objeto **FileWriter** para crear el fichero físico en el disco duro "alumnos.dat".

El objeto **BufferedWriter** añade un buffer intermedio asi vamos escribiendo y se guardaran los datos cuando haya bastantes datos para hacer una escritura eficiente

El objeto **PrintWriter** nos permite imprimir representaciones formateadas de una salida de stream de texto.

Al añadir el carácter " | " separaremos cada registro en el fichero.

```java
//PROCEDIMIENTO PARA MOSTRAR LOS DATOS EN UN jTable
public DefaultTableModel listaAlumnos() {

    //CREAMOS UN VECTOR QUE CONTENGA NOMBRE , EMAIL, TELEFONO ,EDAD
    Vector cabeceras = new Vector();
    cabeceras.addElement("Nombre");
    cabeceras.addElement("Email");
    cabeceras.addElement("Telefono");
    cabeceras.addElement("Edad");

    // CREAMOS UN MODELO DE TABLA PARA AGREGAR EL VECTOR, EN LA UBICACION 0
    DefaultTableModel mdlTabla = new DefaultTableModel(cabeceras, 0);

    try {
        FileReader fr = new FileReader("alumnos.dat");
        BufferedReader br = new BufferedReader(fr);
        String d;
        while ((d = br.readLine()) != null) {
            StringTokenizer dato = new StringTokenizer(d, "|");
            Vector x = new Vector();
            while (dato.hasMoreTokens()) {
                x.addElement(dato.nextToken());
            }
            mdlTabla.addRow(x);
            //contadorPrincipal++;
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e);
    }
    return mdlTabla;
}
```

PROCEDIMIENTO PARA MOSTRAR LOS DATOS DE LOS ALUMNOS EN UN JTABLE.

Creamos un vector y añadimos las cabeceras. Creamos un modelo de tabla para agregar el vector. Los objetos **Filereader** y **BufferedReader** leerán el fichero "**alumnos.dat**" que guardaremos en un String, recorremos el String mediante un bucle while. Dentro del bucle creamos un objeto **StringTokenizer** que recibe el String y el carácter " | " . Creamos el objeto **Vector** . Otro bucle **While** que recorre el objeto **STringTokenizer** añadiendo al Vector cada dato que encuentra separado por el " | ".

Después el dato se añade a la tabla y finalmente devuelve la tabla.

EJERCICIO 3

```
    // PROCEDIMIENTO PARA CONTAR LOS REGISTROS
    public static int contador() {
         contadorPrincipal=0;
        try {
            FileReader fr = new FileReader("alumnos.dat");
            BufferedReader br = new BufferedReader(fr);
            String d;
            while ((d = br.readLine()) != null) {
                contadorPrincipal++;
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
        }
        return contadorPrincipal;
    }

}
```

PROCEDIMIENTO CONTADOR PARA CONTAR LOS REGISTROS

En esta ocasión mediante el objeto **FileReader**,**BufferedReader** y un bucle vamos a leer el fichero "alumnos.dat" para contar los registros y los vamos añadir a una variable **contadorPrincipal** que retornaremos al final y mostraremos en la interfaz gráfica con el número de alumnos que tiene el fichero.

```
    private void jButton_MostrarListaActionPerformed(java.awt.event.ActionEvent evt) {

        jTable1.setModel(metodos.listaAlumnos());
        txtCantidad.setText(Metodos.contador() + "");
    }
```