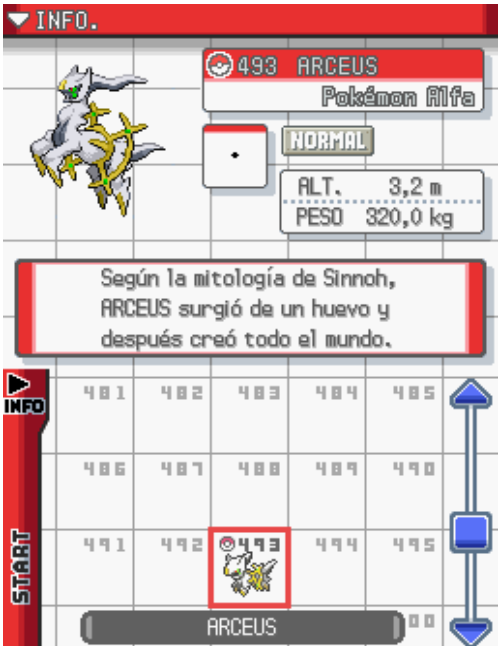
Proyecto individual de JDBC con MySQL.

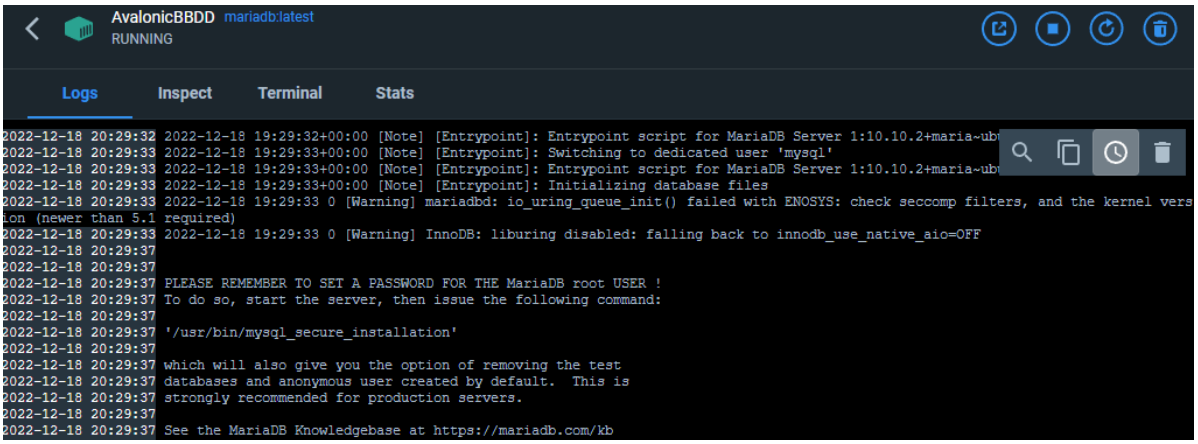
Miguel Ángel Segovia Freeman. DAM2V.

Para este proyecto he decidido simular un prototipo simple de *Pokédex* (solo en consola, sin interfaz gráfica).
Como contexto, es básicamente una enciclopedia digital de las criaturas que pertenecen al famoso videojuego *Pokemon*, y las funcionalidades que tiene en el propio juego son parecidas a las que añado en mi código.



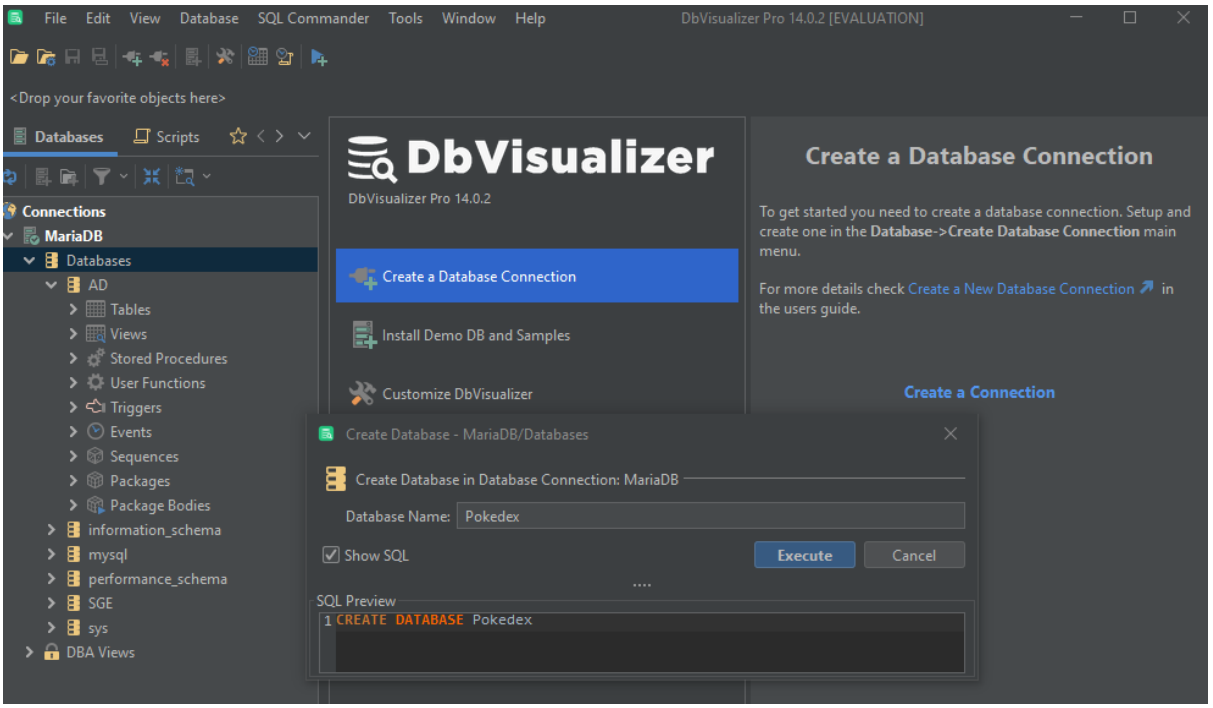
(Captura real del videojuego, cada pokémon está identificado por un número de entrada que es inamovible, Arceus por ejemplo siempre ha sido el nº493)

El entorno de trabajo usado incluye un docker con una imagen de MariaDB, un gestor de bases de datos llamado DBVisualizer para crear la base de datos y cargar los scripts, y Visual Studio Code para escribir el código java.



(Logs del docker con la base de datos corriendo).

Para empezar he creado la base de datos *Pokédex* con DBVisualizer (que previamente ha sido conectada la BBDD de MariaDB).



A continuación he cargado los siguiente scripts (ficheros adjuntos en el proyecto).

Script usado para crear la tabla principal POKEMON:

```
CREATE TABLE POKEMON (
        ENTRADA INT(50) NOT NULL,
        NOMBRE VARCHAR(50) NOT NULL,
        TIPO_PRINCIPAL VARCHAR(50),
        TIPO_SECUNDARIO VARCHAR(50),
        HABILIDAD VARCHAR(50),
        REGION VARCHAR(50),
        ALTURA FLOAT,
        PESO FLOAT,
        CONSTRAINT PRIMARY KEY (ENTRADA)
) COMMENT = 'Aquí se guardará información de distintos pokemones.'
```

Sentencia de ejemplo usada para insertar registros (el script incluye 11 inserciones):

```
INSERT INTO POKEMON
        (ENTRADA, NOMBRE, TIPO_PRINCIPAL, TIPO_SECUNDARIO, HABILIDAD,
        REGION, ALTURA, PESO)
VALUES
        (25, 'PIKACHU', 'ELECTRICO', NULL, 'PARARRAYOS', 'KANTO', 0.4, 6.0);
```

Los datos una vez insertados quedarían de la siguiente manera:

*	ENTRADA	NOMBRE	TIPO_PRINCIPAL	TIPO_SECUNDARIO	HABILIDAD	REGION	ALTURA	PESO
1	25	PIKACHU	ELECTRICO	(null)	PARARRAYOS	KANTO	0.4	6.0
2	249	LUGIA	PSIQUICO	VOLADOR	PRESION	JOHTO	5.2	216.0
3	373	SALAMENCE	DRAGON	VOLADOR	INTIMIDACION	HOENN	1.5	102.6
4	448	LUCARIO	LUCHA	ACERO	FUERZA MENTAL	SINNOH	1.2	54.0
5	530	EXCADRILL	TIERRA	ACERO	ROMPEMOLDES	UNOVA	0.7	40.4
6	563	COFAGRIGUS	FANTASMA	(null)	MOMIA	UNOVA	1.7	76.5
7	727	INCINEROAR	FUEGO	SINIESTRO	INTIMIDACION	ALOLA	1.8	83.0
8	823	CORVIKNIGHT	VOLADOR	ACERO	PRESION	GALAR	2.2	75.0
9	861	GRIMMSNARL	SINIESTRO	HADA	BROMISTA	GALAR	1.5	61.0
10	1000	GOLDENGHO	ACERO	FANTASMA	CUERPO AUREO	PALDEA	1.2	30.0
11	1001	WO-CHIEN	SINIESTRO	PLANTA	TABLILLA DEBACLE	PALDEA	1.5	74.2

Ya configurada la base de datos procedo a mostrar el código escrito en Java, con ejemplos de uso y anotaciones/explicaciones.

Clase Pokemons.

Clase que gestionará la base de datos, desde la conexión hasta los métodos que utilizan Querys/Transacciones.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Pokemons {
    // Declaración de objetos que usaremos para manejar la base de datos
    private Connection conexion;
    private Statement st;
    private ResultSet rs;

    // Constructor del objeto Pokemons
    public Pokemons() throws ClassNotFoundException,
            SQLException, InstantiationException, IllegalAccessException {
        // Driver de mysql
        Class.forName(className: "com.mysql.cj.jdbc.Driver").newInstance();
        conectar();
    }
```

En el constructor se establece el driver **"com.mysql.cj.jdbc.Driver"** con el método forName.

```java
    // Método que conecta a la base de datos
    public void conectar() throws SQLException {
        // Establezco ruta, usuario y contraseña
        conexion = DriverManager.getConnection(url: "jdbc:mysql://localhost:3306/Pokedex", user: "root", password: "shiav1");
        // Mantengo esto del código de ejemplo ya que permite hacer que el cursor sea
        // bidireccional y que el objeto se actualizable
        st = conexion.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
        System.out.println("\nConexion realizada con exito.\n "
                + "\nAccediendo a la Pokedex");
    }
```

Conectar() utilizará la URL: **"jdbc:mysql://localhost:3306/Pokedex"** con el usuario “root” y la contraseña “shiav1”.

Al objeto de tipo statement se le pasarán dos constantes (TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE) que como indican las anotaciones nos da más libertad a la hora de trabajar.
Si la conexión se establece nos aparecerá un mensaje que nos lo confirmará como en la siguiente captura tomada del programa en ejecución.

```
Conexion realizada con exito.

Accediendo a la Pokedex
```

Cerrar conexión cerrará los recursos abiertos solo en el caso de que se hayan usado.
Los siguientes métodos se pueden explicar con las anotaciones del código.

```java
    // Método que cierra los objetos de tipo sql cuando se termina de usar la app
    public void cerrarConexion() throws SQLException {
        if (rs != null)
            rs.close();
        if (st != null)
            st.close();
        if (conexion != null)
            conexion.close();
    }

    // A partir de aquí se definen métodos que usan la BBDD con querys

    // Método que muestra todos los datos de la base de datos
    public void mostrarPokedex() throws SQLException {
        rs = st.executeQuery(sql: "SELECT * FROM POKEMON");
        while (rs.next())
            mostrarFilaActual();
    }

    // Método que muestra los datos de la posición del cursor rs
    public void mostrarFilaActual() throws SQLException {
        int nColumnas = rs.getMetaData().getColumnCount();
        for (int i = 1; i <= nColumnas; ++i) {
            System.out.print(rs.getString(i) + " ");
        }
        System.out.println();
    }

```

Métodos que usan transacciones (INSERT, DELETE) mediante sentencias, además de un método para filtrar según los tipos de pokemon.

```java
    // Método que buscará según el nombre recibido
    public void mostrarBusqueda(String busqueda) throws SQLException {
        rs = st.executeQuery("SELECT * FROM POKEMON WHERE NOMBRE LIKE '" + busqueda.toUpperCase() + "'");
        if (!rs.next()) {
            System.out.println(x: "No se encuentra el pokemon con ese nombre");
        } else {
            mostrarFilaActual();
        }
    }
    //Método de de inserción
    public void insertarPokemon(int entrada, String nombre, String tipo1, String tipo2, String habilidad, String region,
            float altura, float peso) throws SQLException {
        st.executeUpdate(
                "INSERT INTO POKEMON VALUES (" + entrada + ", '" + nombre + "', '" + tipo1 + "', '" + tipo2 +
                        "', '" + habilidad + "', '" + region + "', " + altura + ", " + peso + ")");
    }
    //Método de eliminación
    public void borrarPokemon(int entrada) throws SQLException {
        st.executeUpdate("DELETE FROM " + "POKEMON" +
                " WHERE ENTRADA = " + entrada);
    }
    //Este método mostrará por pantalla los registros que cumplan el tipo especificado en el parámetro
    public void filtrarPortipos(String tipo) throws SQLException {
        rs = st.executeQuery("SELECT * FROM POKEMON WHERE TIPO_PRINCIPAL LIKE '"+tipo.toUpperCase()+
        "' OR TIPO_SECUNDARIO LIKE '"+tipo.toUpperCase()+"' ;");
        if (!rs.next()) {
            System.out.println(x: "No hay almacenados pokemon con ese tipo");
        } else {
            mostrarFilaActual();
            while(rs.next()){
                mostrarFilaActual();
            };
        }
    }
```

Métodos que usan sentencias de ordenación usando ORDER BY.

```java
    //Método que ordena pro orden de entrada de mayor a menor
    public void ultimoAprimero() throws SQLException {
        rs = st.executeQuery(sql: "SELECT * FROM POKEMON ORDER BY ENTRADA DESC");
        while (rs.next())
            mostrarFilaActual();
    }
    //Ordenación alfabetica
    public void alfabeticamente() throws SQLException {
        rs = st.executeQuery(sql: "SELECT * FROM POKEMON ORDER BY NOMBRE");
        while (rs.next())
            mostrarFilaActual();
    }
    //Ordenación de más pequeño al más alto
    public void altura() throws SQLException {
        rs = st.executeQuery(sql: "SELECT * FROM POKEMON ORDER BY ALTURA");
        while (rs.next())
            mostrarFilaActual();
    }
    //Ordenación del más ligero al más pesado
    public void peso() throws SQLException {
        rs = st.executeQuery(sql: "SELECT * FROM POKEMON ORDER BY PESO");
        while (rs.next())
            mostrarFilaActual();
    }
}
```

Clase Pokedex.

Clase principal, aquí se escribe el esqueleto del programa y se usarán los métodos programados previamente.

```java
import java.sql.SQLException;
import java.util.Scanner;

//Clase principal
public class Pokedex {
    // Objeto de tipo Pokemons para poder trabajar con la base de datos
    private static Pokemons BD;
    // Objeto de tipo scanner para poder recibir inputs del usuario
    private static Scanner sc;
```

Método main, se le mostrará al usuario un menú (con el método menú mostrado más adelante) donde tiene que elegir una acción, y según su elección el switch ejecutará un método distinto. Si pulsa 7, saldrá del programa.

```java
    public static void main(String args[]) {
        // Variable que determinará la acción del usuario
        int opcion = 0;
        sc = new Scanner(System.in);
        try {
            BD = new Pokemons();
            // Opciones del menú
            String[] opciones = { "Pokedex",
                    "Buscar pokemon",
                    "Insertar pokemon",
                    "Borrar pokemon",
                    "Filtrar por tipos",
                    "Ordenar",
                    "Salir." };
            do {
                // Llamada al método menu que forma un menú con las opciones pasadas por
                // parámetro.
                // según la decición del usuario se realizará una acción sobre la base de datos
                switch (opcion = menu(opciones, opciones.length)) {
                    case 1:
                        BD.mostrarPokedex();
                        break;
                    case 2:
                        buscarPokemon();
                        break;
                    case 3:
                        insertarPokemon();
                        break;
                    case 4:
                        borrarPokemon();
                        break;
                    case 5:
                        filtrarPortipos();
                        break;
                    case 6:
                        ordenar();
                        break;
                }
            } while (opcion != 7);
```

Manejo de excepciones. Siempre se ejecutará el método cerrarConexion().

```
50          } catch (ClassNotFoundException e) {
51              System.out.println(e.getMessage());
52          } catch (InstantiationException e) {
53              System.out.print(e.getMessage());
54          } catch (IllegalAccessException e) {
55              System.out.print(e.getMessage());
56          } catch (SQLException e) {
57              System.out.print(e.getMessage());
58          } finally {
59              try {
60                  BD.cerrarConexion();
61              } catch (SQLException ignorada) {
62              }
63          }
64      }
```

Resultado del código con el menú (diseño prestado del código de ejemplo).

```
------------------------------------

    1. Pokedex
    2. Buscar pokemon
    3. Insertar pokemon
    4. Borrar pokemon
    5. Filtrar por tipos
    6. Ordenar
    7. Salir.

------------------------------------
```

Si pulsamos 1 tendremos el siguiente resultado ya que el método estaba definido en la clase Pokemons.

```
Opci n (1 - 7): 1
25 PIKACHU ELECTRICO null PARARRAYOS KANTO 0.4 6.0
249 LUGIA PSIQUICO VOLADOR PRESION JOHTO 5.2 216.0
373 SALAMENCE DRAGON VOLADOR INTIMIDACION HOENN 1.5 102.6
448 LUCARIO LUCHA ACERO FUERZA MENTAL SINNOH 1.2 54.0
530 EXCADRILL TIERRA ACERO ROMPEMOLDES UNOVA 0.7 40.4
563 COFAGRIGUS FANTASMA null MOMIA UNOVA 1.7 76.5
727 INCINEROAR FUEGO SINIESTRO INTIMIDACION ALOLA 1.8 83.0
823 CORVIKNIGHT VOLADOR ACERO PRESION GALAR 2.2 75.0
861 GRIMMSNARL SINIESTRO HADA BROMISTA GALAR 1.5 61.0
1000 GOLDENGHO ACERO FANTASMA CUERPO AUREO PALDEA 1.2 30.0
1001 WO-CHIEN SINIESTRO PLANTA TABLILLA DEBACLE PALDEA 1.5 74.2


---------------------------------

    1. Pokedex
    2. Buscar pokemon
    3. Insertar pokemon
    4. Borrar pokemon
    5. Filtrar por tipos
    6. Ordenar
    7. Salir.
```

A continuación tenemos un método de búsqueda simple y otro que se encarga de la recogida de datos que serán insertados en el método insertarPokemon().

```java
    // Método que pide por teclado el nombre del pokemon a buscar y hará la bsuqueda
    // con el objeto de tipo pokemon
    public static void buscarPokemon() throws SQLException {
        System.out.println(x: "\nIntroduce el nombre del pokemon a buscar:");
        System.out.print(s: "> ");
        String busqueda = sc.nextLine();

        BD.mostrarBusqueda(busqueda);
    }

    // Se pedirá por teclado los datos para hacer una inserción
    public static void insertarPokemon() throws SQLException {
        int entrada;
        String nombre, tipo1, tipo2, habilidad, region;
        float altura, peso;

        System.out.println(x: "Entrada: ");
        entrada = sc.nextInt();
        sc.nextLine();
        System.out.println(x: "Nombre: ");
        nombre = sc.nextLine();
        System.out.println(x: "Tipo principal: ");
        tipo1 = sc.nextLine();
        System.out.println(x: "Tipo secundario: ");
        tipo2 = sc.nextLine();
        System.out.println(x: "Habilidad: ");
        habilidad = sc.nextLine();
        System.out.println(x: "Region: ");
        region = sc.nextLine();
        System.out.println(x: "Altura: ");
        altura = sc.nextFloat();
        System.out.println(x: "Peso: ");
        peso = sc.nextFloat();
        BD.insertarPokemon(entrada, nombre, tipo1, tipo2, habilidad, region, altura, peso);
        sc.nextLine();
    }
```

Partiendo del menú anterior, si pulsamos 2 ocurrirá lo siguiente.

```
Opci n (1 - 7): 2

Introduce el nombre del pokemon a buscar:
> salamence
373 SALAMENCE DRAGON VOLADOR INTIMIDACION HOENN 1.5 102.6
```

En el caso de que no se encuentre registrado el pokemon:

```
Opci n (1 - 7): 2

Introduce el nombre del pokemon a buscar:
> arceus
No se encuentra el pokemon con ese nombre
```

Si pulsamos 3 insertamos de la siguiente forma.

```
Opci n (1 - 7): 3
Entrada:
478
Nombre:
Froslass
Tipo principal:
hielo
Tipo secundario:
ghost
Habilidad:
manto niveo
Region:
sinnoh
Altura:
4,03
Peso:
58,6
```

Comprobamos que se ha insertado correctamente.

```
Opci n (1 - 7): 1
25 PIKACHU ELECTRICO null PARARRAYOS KANTO 0.4 6.0
249 LUGIA PSIQUICO VOLADOR PRESION JOHTO 5.2 216.0
373 SALAMENCE DRAGON VOLADOR INTIMIDACION HOENN 1.5 102.6
448 LUCARIO LUCHA ACERO FUERZA MENTAL SINNOH 1.2 54.0
478 FROSLASS HIELO GHOST MANTO NIVEO SINNOH 4.03 58.6
530 EXCADRILL TIERRA ACERO ROMPEMOLDES UNOVA 0.7 40.4
563 COFAGRIGUS FANTASMA null MOMIA UNOVA 1.7 76.5
727 INCINEROAR FUEGO SINIESTRO INTIMIDACION ALOLA 1.8 83.0
823 CORVIKNIGHT VOLADOR ACERO PRESION GALAR 2.2 75.0
861 GRIMMSNARL SINIESTRO HADA BROMISTA GALAR 1.5 61.0
1000 GOLDENGHO ACERO FANTASMA CUERPO AUREO PALDEA 1.2 30.0
1001 WO-CHIEN SINIESTRO PLANTA TABLILLA DEBACLE PALDEA 1.5 74.2
```

Métodos que invocan los que ya definimos anteriormente para eliminar y filtrar pasando los datos recogidos por teclado.

```
103     // Método que realizará una eliminación según la entrada de la pokedex
104     // proporcionada
105     public static void borrarPokemon() throws SQLException {
106         int entrada;
107         System.out.println(x: "Entrada: ");
108         entrada = sc.nextInt();
109         sc.nextLine();
110         BD.borrarPokemon(entrada);
111     }
112
113     // Se pide por teclado el tipo para filtrar y se hará la llamada al método del
114     // mismo nombre
115     public static void filtrarPortipos() throws SQLException {
116         System.out.println(x: "\nIntroduce el nombre del tipo: ");
117         System.out.print(s: "> ");
118         System.out.println(x: "Mostrando pokemons de tipo ");
119         String tipo = sc.nextLine();
120         BD.filtrarPortipos(tipo);
121     }
```

Pulsamos la opción 4, la cual borra.

```
Opci n (1 - 7): 4
Entrada:
478
```

Se borró correctamente.

```
Opci n (1 - 7): 1
25 PIKACHU ELECTRICO null PARARRAYOS KANTO 0.4 6.0
249 LUGIA PSIQUICO VOLADOR PRESION JOHTO 5.2 216.0
373 SALAMENCE DRAGON VOLADOR INTIMIDACION HOENN 1.5 102.6
448 LUCARIO LUCHA ACERO FUERZA MENTAL SINNOH 1.2 54.0
530 EXCADRILL TIERRA ACERO ROMPEMOLDES UNOVA 0.7 40.4
563 COFAGRIGUS FANTASMA null MOMIA UNOVA 1.7 76.5
727 INCINEROAR FUEGO SINIESTRO INTIMIDACION ALOLA 1.8 83.0
823 CORVIKNIGHT VOLADOR ACERO PRESION GALAR 2.2 75.0
861 GRIMMSNARL SINIESTRO HADA BROMISTA GALAR 1.5 61.0
1000 GOLDENGHO ACERO FANTASMA CUERPO AUREO PALDEA 1.2 30.0
1001 WO-CHIEN SINIESTRO PLANTA TABLILLA DEBACLE PALDEA 1.5 74.2
```

Pulsamos la 5 ahora, pasando como parámetro el tipo “acero”.

```
Opci n (1 - 7): 5

Introduce el nombre del tipo:
> Mostrando pokemons de tipo
acero
448 LUCARIO LUCHA ACERO FUERZA MENTAL SINNOH 1.2 54.0
530 EXCADRILL TIERRA ACERO ROMPEMOLDES UNOVA 0.7 40.4
823 CORVIKNIGHT VOLADOR ACERO PRESION GALAR 2.2 75.0
1000 GOLDENGHO ACERO FANTASMA CUERPO AUREO PALDEA 1.2 30.0
```

En el caso de que no estén registrados:

```
Opci n (1 - 7): 5

Introduce el nombre del tipo:
> Mostrando pokemons de tipo
hielo
No hay almacenados pokemon con ese tipo
```

Por último, el método ordenar llamará a un submenú que nos indicará cómo queremos los datos ordenados y debajo el método que crea los menús(inspirados por el código de ejemplo).

```
123     // El método invocará un submenú donde el usuario deberá indicar como quiere que
124     // se ordenen los registros
125     public static void ordenar() throws SQLException {
126         String[] opciones = { "De último a primero",
127                 "Alfabeticamente ",
128                 "Por altura",
129                 "Por peso",
130                 "Salir." };
131         int opcion = 0;
132         do {
133             // Las opciones están deficnidas en la clase Pokemons
134             switch (opcion = menu(opciones, opciones.length)) {
135                 case 1:
136                     BD.ultimoAprimero();
137                     break;
138                 case 2:
139                     BD.alfabeticamente();
140                     break;
141                 case 3:
142                     BD.altura();
143                     break;
144                 case 4:
145                     BD.peso();
146                     break;
147             }
148         } while (opcion != 5);
149     }
150 
151     // Método que construye un menú, el usuario meterá la opción deseada por
152     // tecleado con el objeto escaner
153     public static int menu(String[] opciones, int numOpciones) {
154         int i = 0, opcion = 0;
155 
156         System.out.println(x: "\n_____\n");
157         for (i = 1; i <= numOpciones; ++i) {
158             System.out.print("    " + i + ". " + opciones[i - 1] + "\n");
159         }
160         System.out.println(x: "_____\n");
161         do {
162             System.out.print("\nOpci n (1 - " + numOpciones + "): ");
163             opcion = sc.nextInt();
164             sc.nextLine();
165         } while (opcion < 1 || opcion > numOpciones);
166         return opcion;
167     }
168 }
```

Vamos a ordenar por peso en este ejemplo, pulsamos 6, y después 4. Aparecerán primero los más ligeros.

```
Opci n (1 - 5): 4
25 PIKACHU ELECTRICO null PARARRAYOS KANTO 0.4 6.0
1000 GOLDENGHO ACERO FANTASMA CUERPO AUREO PALDEA 1.2 30.0
530 EXCADRILL TIERRA ACERO ROMPEMOLDES UNOVA 0.7 40.4
448 LUCARIO LUCHA ACERO FUERZA MENTAL SINNOH 1.2 54.0
861 GRIMMSNARL SINIESTRO HADA BROMISTA GALAR 1.5 61.0
1001 WO-CHIEN SINIESTRO PLANTA TABLILLA DEBACLE PALDEA 1.5 74.2
823 CORVIKNIGHT VOLADOR ACERO PRESION GALAR 2.2 75.0
563 COFAGRIGUS FANTASMA null MOMIA UNOVA 1.7 76.5
727 INCINEROAR FUEGO SINIESTRO INTIMIDACION ALOLA 1.8 83.0
373 SALAMENCE DRAGON VOLADOR INTIMIDACION HOENN 1.5 102.6
249 LUGIA PSIQUICO VOLADOR PRESION JOHTO 5.2 216.0
```