Proyecto Patrón de Diseño DAO

Para este proyecto usaré como base los scripts y el entorno del proyecto anterior sobre la Pokédex.
Para empezar, en el script de crear tablas he añadido la tabla *HABILIDADES,* la cual almacenará el nombre y descripción de las distintas habilidades que poseen los pokémon a la hora de combatir en el videojuego. En la tabla *POKEMON* habilidad pasará a ser clave foránea apuntando a nombre en la otra. Quedando el script con los cambios de esta manera:

```
CREATE TABLE
	HABILIDAD
	(
	NOMBRE	VARCHAR(50) NOT NULL,
	DESCRIPCION VARCHAR(50),
	PRIMARY KEY (NOMBRE)
	)
	COMMENT='Habilidades con su descripción.';

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),
	CONSTRAINT HAB_FOREIGN_KEY FOREIGN KEY (HABILIDAD) REFERENCES
HABILIDAD (NOMBRE)
	)
	COMMENT = 'Pokemon con sus datos.';
```

Ejemplo de inserción en ambas tablas (hay 9 inserciones en la tabla *HABILIDADES*).

```
INSERT INTO HABILIDAD (NOMBRE, DESCRIPCION)
VALUES (
	'INTIMIDACION',
	'AL SALIR AL COMBATE -1 AL ATAQUE ENEMIGO'
  );

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

La tabla HABILIDADES con los datos una vez insertados quedan de la siguiente manera:

*	NOMBRE	DESCRIPCION
1	BROMISTA	ATAQUES DE ESTADO AUMENTA EN 1 SU PRIORIDAD
2	CUERPO AUREO	IMPIDE RECIBIR ATAQUES DE ESTADO
3	FUERZA MENTAL	INMUNE A AMEDRENTARSE
4	INTIMIDACION	AL SALIR AL COMBATE -1 AL ATAQUE ENEMIGO
5	MOMIA	HAB. ENEMIGA ES MOMIA AL RECIBIR CONTACTO
6	PARARRAYOS	ATAQUES ELECTRICOS RECIBIDOS NO DAÑAN, +1 SP.ATK
7	PRESION	ATAQUES ENEMIGOS CONSUMEN EL DOBLE DE PP
8	ROMPEMOLDES	IGNORA HAB. ENEMIGA QUE IMPIDA DAÑO O EFECTO
9	TABLILLA DEBACLE	-25% EL ATAQUE DEL CAMPO

Y la tabla POKEMON:

*	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

Una vez preparada la base de datos el código java sería el siguiente.

Las clases, las cuales están escritas siguiendo el diseño de patrón DAO están organizadas en distintos paquetes para tener una mejor organización,

CLASES USANDO LA TABLA HABILIDADES.

Clase HabilidadInterface.

Esta clase definirá sobre la clase HabilidadBean qué métodos debe implementar, siendo estos dos getter, un setter, y el método getNuevaHabilidad qué devolverá un objeto instanciado de HabilidadBean, a la vez que lo inserta en la base de datos.

src > HabilidadDAO > HabilidadInterface.java > {} HabilidadDAO

```java
package HabilidadDAO;

public interface HabilidadInterface {
    public HabilidadInterface getNuevaHabilidad(String nombre, String descripcion);

    public String getNombre();

    public String getDescripcion();

    public void setDescripcion(String descripcion);
}
```

Clase FactoriaHabilidades.
Esta solo definirá un método sobre la interfaz HabilidadInterface, la cual solo devolverá un objeto de tipo HabilidadBean sin definir datos, esta será la base del objeto DAO.

src > HabilidadDAO > FactoriaHabilidades.java > {} HabilidadDAO

```java
package HabilidadDAO;

public class FactoriaHabilidades {
    public static HabilidadInterface getHabilidadDAO() {
        return new HabilidadBean();
    }
}
```

Clase HabilidadBean.
Esta clase tendrá atributos relacionados con la tabla *HABILIDADES,* con ella podremos instanciar objetos con la información de la tabla y manipular información de la misma con comandos SQL. En la siguiente captura vemos la definición de atributos, constructores y el primer método implementado de la interfaz (getNuevaHabilidad()) que instancia el objeto a la vez que lo inserta en la base de datos.

src > HabilidadDAO > HabilidadBean.java > HabilidadBean

```java
package HabilidadDAO;

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

public class HabilidadBean implements HabilidadInterface {
    private String nombre;
    private String descripcion;
    private Connection conexion;

    public HabilidadBean() {
    }

    public HabilidadBean(String nombre, String descripcion) {
        this.nombre = nombre;
        this.descripcion = descripcion;
    }

    @Override
    public HabilidadInterface getNuevaHabilidad(String nombre, String descripcion) {

        conexion = getConexion();
        try (Statement st = conexion.createStatement()) {
            st.execute(
                    "INSERT INTO HABILIDAD (NOMBRE, DESCRIPCION) VALUES('" + nombre.toUpperCase() + "','" +
                            descripcion.toUpperCase() + "')");
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return new HabilidadBean(nombre, descripcion);
    }
```

En esta segunda captura se muestran los métodos get y set los cuales usa la cláusula UPDATE para actualizar los datos en la tabla cada vez que se modifica el objeto. Y por último el método getConexion() que se usa para conectar a la base de datos.

```
34
35      @Override
36      public String getNombre() {
37          return this.nombre;
38      }
39
40      @Override
41      public String getDescripcion() {
42          return this.descripcion;
43      }
44
45      @Override
46      public void setDescripcion(String descripcion) {
47          conexion = getConexion();
48          try (Statement st = conexion.createStatement()) {
49              st.execute(
50                      "UPDATE HABILIDAD SET DESCRIPCION'" + descripcion +
51                              "'WHERE NOMBRE='" + this.nombre + "'");
52              st.close();
53              conexion.close();
54          } catch (SQLException e) {
55              e.printStackTrace();
56          }
57          this.descripcion = descripcion;
58      }
59
60      private Connection getConexion() {
61          try {
62              Class.forName(className: "com.mysql.cj.jdbc.Driver");
63              Connection con = DriverManager.getConnection(
64                      url: "jdbc:mysql://localhost/Pokedex", user: "root", password: "shiav1");
65              return con;
66          } catch (SQLException | ClassNotFoundException e) {
67              e.printStackTrace();
68              return null;
69          }
70      }
71  }
```

Clase AplicacionHabilidad

Aquí probamos lo programado anteriormente, instanciamos un objeto DAO y lo usamos para insertar la habilidad “Poder Solar” en la base de datos. Además, tenemos un objeto el cual podemos modificar, añadiendo una descripción por ejemplo.

```
src > HabilidadDAO > J AplicacionHabilidad.java > AplicacionHabilidad
1   package HabilidadDAO;
2
3   public class AplicacionHabilidad {
        Run | Debug
4       public static void main(String[] args) {
5           HabilidadInterface objetoDAO = FactoriaHabilidades.getHabilidadDAO();
6
7           HabilidadInterface hab1 = objetoDAO.getNuevaHabilidad(nombre: "Poder Solar", descripcion: " ");
8
9           System.out.println("INSERTADO SIN DESCRIPCIÓN\n"+hab1.getNombre()+"\n"+hab1.getDescripcion());
10
11          hab1.setDescripcion(descripcion: "1.5x de Atk bajo el sol, 1/8 de HP menos por turno");
12
13          System.out.println("Descripcion modificada: "+hab1.getDescripcion());
14      }
15  }
```

Resultado de la ejecución en la terminal.

```
TERMINAL
PS C:\Users\migue\Documents\GitHub\AD-2223\Java\Prácticas\PokedexDAO\pokedexDAO>  c:; cd 'c:\Users\migue\Documents\GitHub\AD-2223\Java\Prácticas\PokedexDAO\pokedexDAO'; & 'C:\Program Files\Java\jdk1.8.0_311\bin\java.exe' '-cp' 'C:\Users\migue\AppData\Local\Temp\cp_b8h5mudekv3ble5t2c1c5g254.jar' 'HabilidadDAO.AplicacionHabilidad'
INSERTADO SIN DESCRIPCIÓN
Poder Solar

Descripcion modificada: 1.5x de Atk bajo el sol, 1/8 de HP menos por turno
```

CLASES USANDO LA TABLA POKEMON.

Clase PokemonInterface.

Esta clase definirá sobre la clase PokemonBean qué métodos debe implementar.
Los getter que devolverán el campo deseado.

src > PokemonDAO > PokemonInterface.java > {} Po

```java
package PokemonDAO;

import java.util.ArrayList;

public interface PokemonInterface {
    // GETTERS
    public int getEntrada();

    public String getNombre();

    public String getTipo1();

    public String getTipo2();

    public String getHabilidad();

    public String getRegion();

    public float getAltura();

    public float getPeso();
```

Los setter que modifican tanto el objeto como la base de datos.

```java

    // SETTERS
    public void setNombre(String nombre);

    public void setTipo1(String tipo1);

    public void setTipo2(String tipo2);

    public void setHabilidad(String habilidad);

    public void setRegion(String region);

    public void setAltura(float altura);

    public void setPeso(float peso);

```

Los métodos de búsqueda que devolverán el objeto/objetos que se desee, un borrado y por último un constructor básico para instanciar e insertar.

```
38      // BUSQUEDAS
39      public PokemonBean getPokemonPorNumeroEntrada(int entrada);
40
41      public ArrayList<PokemonBean> getPokemonPorTipo(String tipo);
42
43      public ArrayList<PokemonBean> getPokemonPorHabilidad(String habilidad);
44
45      // BORRADO
46      public void delete();
47
48      // CONSTRUCTOR
49      public PokemonInterface getNuevoPokemon(
50              int entrada, String nombre, String habilidad, String tipo1, String tipo2, String region, float altura,
51              float peso);
52  }
```

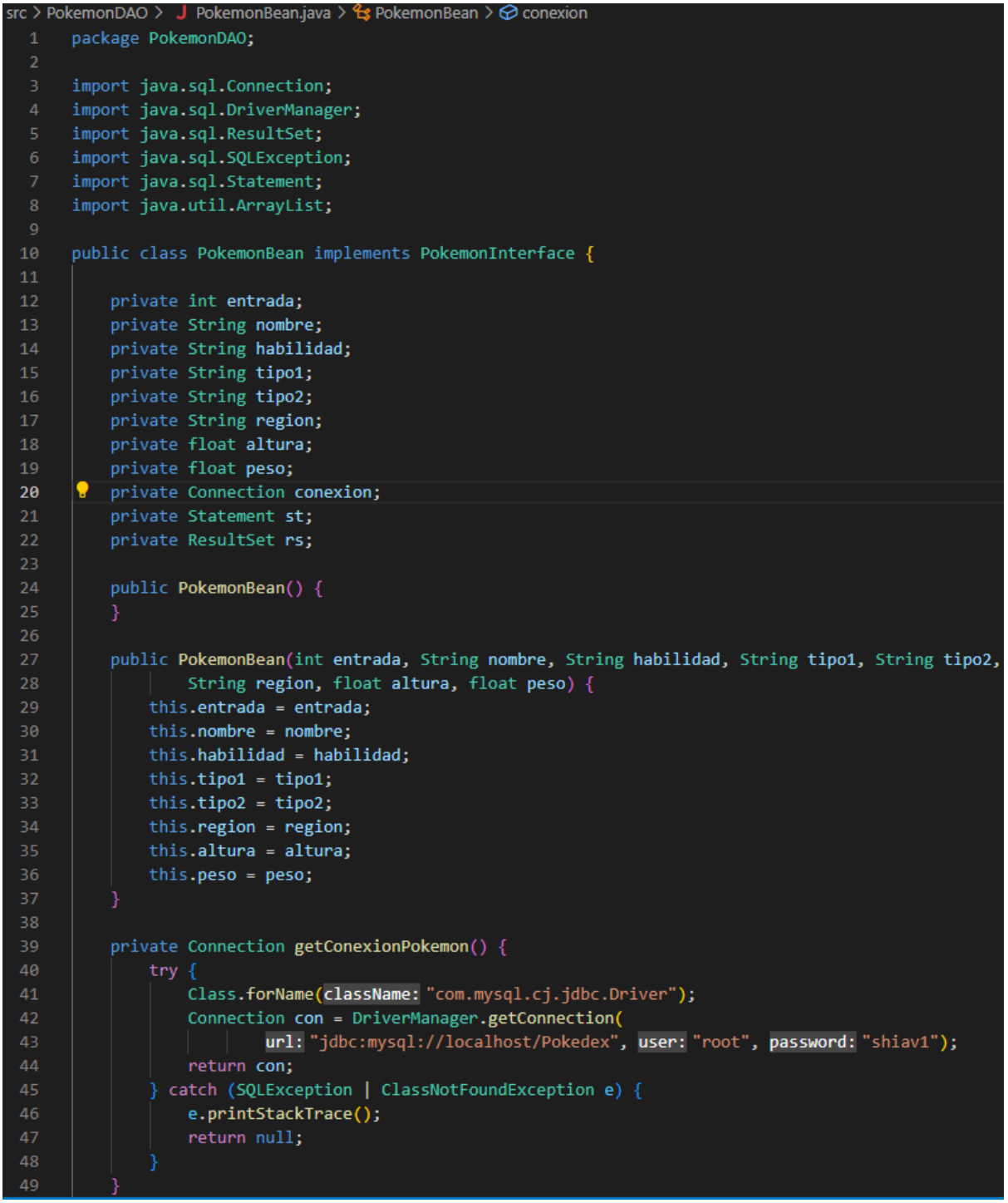
Clase FactoriaPokemons

Esta solo definirá un método sobre la interfaz PokemonInterface, la cual solo devolverá un objeto de tipo PokemonBean sin definir datos, esta será la base del objeto DAO.

```
src > PokemonDAO > J FactoriaPokemons.java > {} PokemonDAO
1   package PokemonDAO;
2
3   public class FactoriaPokemons {
4       public static PokemonInterface getPokemonDAO(){
5           return new PokemonBean();
6       }
7   }
8
```

Clase PokemonBean

Esta clase tendrá atributos relacionados con la tabla *POKEMON,* con ella podremos instanciar objetos con la información de la tabla y manipular información de la misma con comandos SQL. En la siguiente captura vemos la definición de atributos, constructores y el método que conecta con la base de datos.



A continuación se muestran los métodos implementados, los get devolverán el valor pedido y los set modificaron tanto el objeto como la base de datos.

```java
    public int getEntrada() {
        return this.entrada;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("UPDATE POKEMON SET NOMBRE = '" +
                    nombre.toUpperCase() + "' WHERE ENTRADA = " + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        this.nombre = nombre;
    }

    public String getHabilidad() {
        return this.habilidad;
    }

    public void setHabilidad(String habilidad) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("UPDATE POKEMON SET HABILIDAD = '" +
                    habilidad.toUpperCase() + "' WHERE ENTRADA = " + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        this.habilidad = habilidad;
    }

    public String getTipo1() {
        return this.tipo1;
    }

```

```java
    public void setTipo1(String tipo1) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("UPDATE POKEMON SET TIPO_PRINCIPAL = '" +
                    tipo1.toUpperCase() + "' WHERE ENTRADA = " + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        this.tipo1 = tipo1;
    }

    public String getTipo2() {
        return this.tipo2;
    }

    public void setTipo2(String tipo2) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("UPDATE POKEMON SET TIPO_SECUNDARIO = '" +
                    tipo2.toUpperCase() + "' WHERE ENTRADA = " + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        this.tipo2 = tipo2;
    }

    public String getRegion() {
        return this.region;
    }

    public void setRegion(String region) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("UPDATE POKEMON SET REGION = '" +
                    region.toUpperCase() + "' WHERE ENTRADA = " + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        this.region = region;
    }
```

```
144
145     public float getAltura() {
146         return this.altura;
147     }
148
149     public void setAltura(float altura) {
150         conexion = getConexionPokemon();
151         try {
152             st = conexion.createStatement();
153             st.execute("UPDATE POKEMON SET ALTURA = " +
154                     String.valueOf(altura) + " WHERE ENTRADA = " + this.entrada);
155             st.close();
156             conexion.close();
157         } catch (SQLException e) {
158             e.printStackTrace();
159         }
160         this.altura = altura;
161     }
162
163     public float getPeso() {
164         return this.peso;
165     }
166
167     public void setPeso(float peso) {
168         conexion = getConexionPokemon();
169         try {
170             st = conexion.createStatement();
171             st.execute("UPDATE POKEMON SET PESO = " +
172                     String.valueOf(peso) + " WHERE ENTRADA = " + this.entrada);
173             st.close();
174             conexion.close();
175         } catch (SQLException e) {
176             e.printStackTrace();
177         }
178         this.peso = peso;
179     }
```

Se implementan los métodos de búsqueda que devolverán el objeto buscado, o un ArrayList dependiendo del método, pasaremos la condición por parámetro.

Búsqueda de un pokemon particular por su número de entrada.

```java

    @Override
    public PokemonBean getPokemonPorNumeroEntrada(int entradaB) {
        conexion = getConexionPokemon();
        PokemonBean pokemon = new PokemonBean();
        try {
            st = conexion.createStatement();
            rs = st.executeQuery("SELECT * FROM POKEMON WHERE ENTRADA = " + entradaB);
            while (rs.next()) {
                pokemon.entrada = rs.getInt(columnLabel: "ENTRADA");
                pokemon.nombre = rs.getString(columnLabel: "NOMBRE");
                pokemon.habilidad = rs.getString(columnLabel: "HABILIDAD");
                pokemon.tipo1 = rs.getString(columnLabel: "TIPO_PRINCIPAL");
                pokemon.tipo2 = rs.getString(columnLabel: "TIPO_SECUNDARIO");
                pokemon.region = rs.getString(columnLabel: "REGION");
                pokemon.altura = rs.getFloat(columnLabel: "ALTURA");
                pokemon.peso = rs.getFloat(columnLabel: "PESO");
            }
            rs.close();
            st.close();
            conexion.close();
            return pokemon;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

```

Búsqueda de pokemons de un determinado tipo.

```java
    @Override
    public ArrayList<PokemonBean> getPokemonPorTipo(String tipoB) {
        conexion = getConexionPokemon();
        PokemonBean pokemon;
        ArrayList<PokemonBean> pokemons = new ArrayList<>();
        try {
            st = conexion.createStatement();
            rs = st.executeQuery("SELECT * FROM POKEMON WHERE TIPO_PRINCIPAL LIKE '" + tipoB.toUpperCase() +
                    "' OR TIPO_SECUNDARIO LIKE '" + tipoB.toUpperCase() + "' ");
            while (rs.next()) {
                pokemon = new PokemonBean();
                pokemon.entrada = rs.getInt(columnLabel: "ENTRADA");
                pokemon.nombre = rs.getString(columnLabel: "NOMBRE");
                pokemon.habilidad = rs.getString(columnLabel: "HABILIDAD");
                pokemon.tipo1 = rs.getString(columnLabel: "TIPO_PRINCIPAL");
                pokemon.tipo2 = rs.getString(columnLabel: "TIPO_SECUNDARIO");
                pokemon.region = rs.getString(columnLabel: "REGION");
                pokemon.altura = rs.getFloat(columnLabel: "ALTURA");
                pokemon.peso = rs.getFloat(columnLabel: "PESO");
                pokemons.add(pokemon);
            }
            rs.close();
            st.close();
            conexion.close();
            return pokemons;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

Búsqueda de pokemons según su habilidad.

```java

    @Override
    public ArrayList<PokemonBean> getPokemonPorHabilidad(String habilidad) {
        conexion = getConexionPokemon();
        PokemonBean pokemon;
        ArrayList<PokemonBean> pokemons = new ArrayList<>();
        try {
            st = conexion.createStatement();
            rs = st.executeQuery("SELECT * FROM POKEMON WHERE HABILIDAD LIKE '" + habilidad.toUpperCase() + "' ;");
            while (rs.next()) {
                pokemon = new PokemonBean();
                pokemon.entrada = rs.getInt(columnLabel: "ENTRADA");
                pokemon.nombre = rs.getString(columnLabel: "NOMBRE");
                pokemon.habilidad = rs.getString(columnLabel: "HABILIDAD");
                pokemon.tipo1 = rs.getString(columnLabel: "TIPO_PRINCIPAL");
                pokemon.tipo2 = rs.getString(columnLabel: "TIPO_SECUNDARIO");
                pokemon.region = rs.getString(columnLabel: "REGION");
                pokemon.altura = rs.getFloat(columnLabel: "ALTURA");
                pokemon.peso = rs.getFloat(columnLabel: "PESO");
                pokemons.add(pokemon);
            }
            rs.close();
            st.close();
            conexion.close();
            return pokemons;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

Por último, tenemos los métodos de borrado y el que inserta/crea el objeto.

```java

    @Override
    public void delete() {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute("DELETE FROM POKEMON WHERE ENTRADA =" + this.entrada);
            st.close();
            conexion.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public PokemonInterface getNuevoPokemon(int entrada, String nombre, String habilidad, String tipo1, String tipo2,
            String region, float altura, float peso) {
        conexion = getConexionPokemon();
        try {
            st = conexion.createStatement();
            st.execute(
                    "INSERT INTO POKEMON VALUES (" + entrada + ", '" + nombre.toUpperCase() + "', '"
                            + tipo1.toUpperCase() + "', '" + tipo2.toUpperCase() +
                            "', '" + habilidad.toUpperCase() + "', '" + region.toUpperCase() + "', " + altura + ", "
                            + peso + ")");
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return new PokemonBean(entrada, nombre, habilidad, tipo1, tipo2, region, altura, peso);
    }
}
```

Clase AplicacionPokemon.

Aquí probamos que los métodos programados funcionan.

Usaré de ejemplo a “Charizard”, le insertaré, modificaré y borraré. También recuperaré a los pokémon de tipo acero.

src > PokemonDAO > AplicacionPokemon.java > AplicacionPokemon > main(String[])

```java
package PokemonDAO;

import java.util.ArrayList;

public class AplicacionPokemon {
    Run | Debug
    public static void main(String[] args) {
        PokemonInterface objetoDAO = FactoriaPokemons.getPokemonDAO();

        PokemonInterface poke1 = objetoDAO.getNuevoPokemon(
                entrada: 6, nombre: "CHARIZARD", habilidad: "PODER SOLAR", tipo1: "FUEGO", tipo2: " ", region: "KANTO", altura: 1.70f, peso: 95.00f);

        System.out.println(poke1.getNombre() + " insertado");

        poke1.setTipo2(tipo2: "volador");

        System.out.println("Añadido el tipo "+poke1.getTipo2()+" a "+poke1.getNombre());

        ArrayList<PokemonBean> pokes = objetoDAO.getPokemonPorTipo(tipo: "acero");

        for (PokemonBean pokemonBean : pokes) {
            System.out.println(pokemonBean.getNombre());
        }

        poke1.delete();

        //Busco la entrada 6 que es la que acabo de borrar
        System.out.println(objetoDAO.getPokemonPorNumeroEntrada(entrada: 6).getNombre());
    }
}
```

Resultado de la consola, todo funciona a la perfección:

```
TERMINAL
PS C:\Users\migue\Documents\GitHub\AD-2223\Java\Prácticas\PokedexDAO\pokedexDAO>  c:; cd 'c:\Users\migue\Documents\GitHub\AD-2223\Java\Prácticas\PokedexDAO\pokedexDAO'; & 'C:\Program Files\Java\jdk1.8.0_311\bin\java.exe' '-cp' 'C:\Users\migue\AppData\Local\Temp\cp_b8h5mudekv3ble5t2c1c5g254.jar' 'PokemonDAO.AplicacionPokemon'
CHARIZARD insertado
Añadido el tipo volador a CHARIZARD
LUCARIO
EXCADRILL
CORVIKNIGHT
GOLDENGHO
null
PS C:\Users\migue\Documents\GitHub\AD-2223\Java\Prácticas\PokedexDAO\pokedexDAO>
```