CUCEI

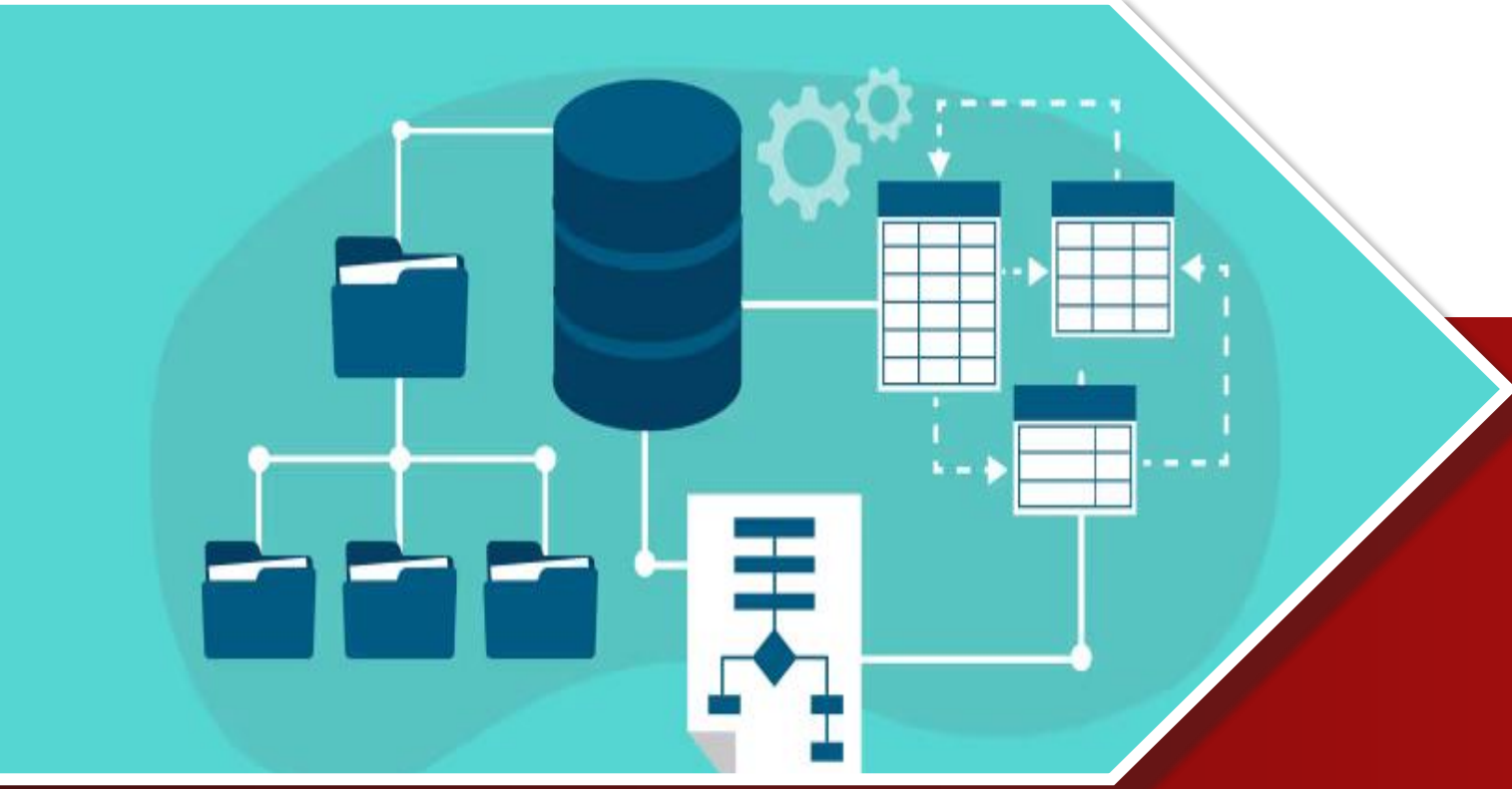
Actividad de Aprendizaje 11

La Lista: Implementación Dinámica Doblemente Ligada

Centro Universitario de Ciencias Exactas e Ingenierías

Materia: Estructuras de Datos

Clave: V0731 ***Sección***: D02



Alumno: Mariscal Rodríguez Omar Jesús

Código: 220858478

Profesor: Gutiérrez Hernández Alfredo

Fecha: 26 de Octubre de 2025

UNIVERSIDAD DE GUADALAJARA

PIENSA Y TRABAJA

Contenido

Test de Autoevaluación

Autoevaluación			
Concepto	***Sí***	***No***	***Acumulado***
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	***-100 pts***	***0 pts***	***0***
Incluí el código fuente ***en formato de texto (sólo si funciona cumpliendo todos los requerimientos)***	***+25 pts***	***0 pts***	***25***
Incluí las ***impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)***	***+25 pts***	***0 pts***	***25***
Incluí una ***portada*** que identifica mi trabajo (nombre, código, materia, fecha, título)	***+25 pts***	***0 pts***	***25***
Incluí una ***descripción y conclusiones*** de mi trabajo	***+25 pts***	***0 pts***	***25***
		Suma:	***100***

Introducción y Abordaje del Problema

El propósito de esta actividad fue trabajar y familiarizarnos con un tipo de lista diferente, una doblemente ligada con una implementación dinámica, con todas las ventajas y responsabilidades que ello requiere. Como se nos daba libertad de uso para elegir el tipo de lista doblemente ligada, se utilizó la lista doblemente ligada con encabezado Dummy.

Abordaje del Problema

Esta actividad nuevamente es una readaptación del código de canciones que ya venimos trabajando desde hace un tiempo, esta vez con una lista doblemente ligada y dinámica, en este caso, utilicé una de encabezado Dummy, además, para hacer un poco más sencillo su manejo sobre todo para la clase Menú y diversas funciones, pensé en colocar un adicional a este encabezado, un dato entero que representara la cantidad de elementos presentes en la lista, esto ahorra mucho en costo computacional sobre todo, ya que hay partes del programa donde se informa la cantidad de elementos, como a la hora de hacer una limpieza de canciones, con una lista dinámica sin encabezados, se tendría que hacer un recorrido de lista con un contador para esta tarea, lo cual es un costo bastante elevado para solo saber el tamaño que tiene la lista, por lo que añadimos este atributo a nuestra clase plantilla de lista, y claro, hay que cuidarlo, sumarle en las adiciones, restarle en las eliminaciones y hacer una correcta copia en los operadores de asignación o en el “add” de la lista.

Fuera de esto, para su suave transición agregaremos un método que devuelve una posición según un entero, para que el menú no tenga que lidiar directamente con esto. Se trabajará también con los ordenamientos, ahora que la lista es doblemente enlazada, los ordenamientos cambiarán de implementación (no de idea central), empezando por el hecho de que no comienzan con un ancla, sino que empiezan con el primer elemento real de la lista, el cual es el siguiente nodo del “Dummy”.

Con el cuidado necesario, la transición no será muy difícil y la compatibilidad con el menú no debería ser un problema más allá de uno que otro “warning” al que habrá que ir poniendo atención.

Programación

El proceso de la codificación (que sería un término más adecuado para esta segunda subsección ya que la programación empieza del planteamiento, no del tecleo del código), se centró sobre todo en la re-implementación de los algoritmos de búsqueda en su gran mayoría, indicar que deben parar al llegar al nodo "Dummy" y hacer que comiencen en el siguiente de este mismo, aunque las mismas cualidades de este tipo de lista doblemente ligada hacen que algoritmos como la adición o eliminación sean bastante naturales y sin tener que considerar tantos casos más que una circularidad inicial, concepto de suma importancia que también se tuvo que cuidar en el "insertSortedData" que se codificó para que esta lista no careciera de herramientas útiles; el algoritmo; el programa ahora advierte del algoritmo de shellSort como uno ineficiente, sigue estando disponible para que el usuario lo seleccione así como la radiodifusora pidió en actualizaciones pasadas, pero como se requiere un acceso aleatorio a la memoria, la complejidad BigO se dispara, y esto se advierte en el programa, que ahora no es una opción tan llamativa como lo era antes.

Para el manejo de nodos y cuidado de la memoria son muy útiles los destructores que aprovechamos para cada nodo individual, y como ya se había adaptado un poco el menú a una lista dinámica, la transición a esta para la clase que interactúa con el usuario fue sumamente suave, tal cual debería ser una actualización.

Código Fuente

Carpeta Include

list.hpp

```
#ifndef __LIST_H__
#define __LIST_H__

#include <fstream>
#include <iomanip>
#include <sstream>
#include <string>

#include "ownexceptions.hpp"

using namespace std;

// Definición
template <class T>
class List {
  class Node;

 public:
  typedef Node* Position;

 private:
  int totalElements = 0;
  class Node {
   private:
    T* dataPtr = nullptr;
    Position prev = nullptr;
    Position next = nullptr;

   public:
    Node();
    Node(const T&);

    ~Node();

    T* getDataPtr();
    T& getData();

    Position getPrev() const;
    Position getNext() const;

    void setDataPtr(T*);
    void setData(const T&);
    void setNext(const Position&);
```

```
    void setPrev(const Position&);
};

bool isValidPosition(const Position&) const;
void add(const List<T>&);
void swapData(Position, Position);

Position header = nullptr;

public:
List();
List(const List<T>&);

~List();

bool isEmpty() const;

void insertData(const Position&, const T&);
void insertSortedData(const T&);
void deleteData(const Position&);
int getTotalElements() const;
Position getFirstPos() const;
Position getLastPos() const;
Position getPrevPos(const Position&) const;
Position getNextPos(const Position&) const;
Position getNodeAt(int index) const;

Position findData(const T&) const;
T& retrieve(const Position&);

string toString() const;
string toString(const T&, int(const T&, const T&)) const;

void sortDataBubble();
void sortDataInsert();
void sortDataSelect();
void sortDataShell();

void deleteAll();

List<T>& operator=(const List<T>&);

void sortDataBubble(int(const T&, const T&));
void sortDataInsert(int(const T&, const T&));
void sortDataSelect(int(const T&, const T&));
void sortDataShell(int(const T&, const T&));

template <class X>
```

```
    friend std::ostream& operator<<(std::ostream&, const List<X>&);
  template <class X>
  friend std::istream& operator>>(std::istream&, List<X>&);
};

// Implementación
// Node
template <class T>
List<T>::Node::Node() {}

template <class T>
List<T>::Node::Node(const T& element) : dataPtr(new T(element)) {
  if (dataPtr == nullptr) {
    throw NodeExceptions::NodeExceptionTo("Memoria no disponible, Node");
  }
}

template <class T>
List<T>::Node::~Node() {
  delete this->dataPtr;
}

template <class T>
T* List<T>::Node::getDataPtr() {
  return this->dataPtr;
}

template <class T>
T& List<T>::Node::getData() {
  if (this->dataPtr == nullptr)
    throw NodeExceptions::NodeExceptionTo("Dato Inexistente, getData");
  return *this->dataPtr;
}

template <class T>
typename List<T>::Position List<T>::Node::getPrev() const {
  return this->prev;
}

template <class T>
typename List<T>::Position List<T>::Node::getNext() const {
  return this->next;
}

template <class T>
void List<T>::Node::setDataPtr(T* p) {
  this->dataPtr = p;
}
```

```
template <class T>
void List<T>::Node::setData(const T& e) {
  if (this->dataPtr == nullptr) {
    this->dataPtr = new T(e);
    if (this->dataPtr == nullptr)
      throw NodeExceptions::NodeExceptionTo("Memoria no disponible,
setData");
  } else
    *this->dataPtr = e;
}

template <class T>
void List<T>::Node::setPrev(const typename List<T>::Position& pointer) {
  this->prev = pointer;
}

template <class T>
void List<T>::Node::setNext(const typename List<T>::Position& pointer) {
  this->next = pointer;
}

// Lista

template <class T>
bool List<T>::isValidPosition(const Position& pointer) const {
  Position aux(this->header->getNext());

  while (aux != header) {
    if (aux == pointer) {
      return true;
    }
    aux = aux->getNext();
  }

  return false;
}

template <class T>
void List<T>::add(const List<T>& other) {
  Position aux(other.header->getNext()), newNode;

  while (aux != other.header) {
    if (newNode = new Node(aux->getData()) == newNode == nullptr)
      throw DataContainersExceptions::MemoryDeficiency("Memoria no
Disponible");

    newNode->setPrev(this->header->getPrev());
```

```
      newNode->setNext(this->header);

      this->header->getPrev()->setNext(newNode);
      this->header->setPrev(newNode);
      this->totalElements++;

      aux = aux->getNext();
    }
  }

  template <class T>
  void List<T>::swapData(Position a, Position b) {
    T temp = a->getData();
    a->setData(b->getData());
    b->setData(temp);
  }

  template <class T>
  List<T>::List() : header(new Node), totalElements(0) {
    if (this->header == nullptr)
      throw NodeExceptions::NodeExceptionTo("Memoria no Disponible, List");

    this->header->setPrev(this->header);
    this->header->setNext(this->header);
  }

  template <class T>
  List<T>::List(const List<T>& other) : totalElements(other.totalElements)
  {
    this->add(other);
  }

  template <class T>
  List<T>::~List() {
    this->deleteAll();

    delete this->header;
  }

  template <class T>
  bool List<T>::isEmpty() const {
    return this->header->getNext() == this->header;
  }

  template <class T>
  void List<T>::insertData(const typename List<T>::Position& position,
                           const T& element) {
    if (position != nullptr && !this->isValidPosition(position))
```

```
    throw DataContainersExceptions::InvalidPosition();

  Position newNode(new Node(element));
  if (newNode == nullptr)
    throw DataContainersExceptions::MemoryOverflow("Memoria No
Disponible");

  Position insPos(position == nullptr ? header : position);

  newNode->setPrev(insPos);
  newNode->setNext(insPos->getNext());

  insPos->getNext()->setPrev(newNode);
  insPos->setNext(newNode);
  this->totalElements++;
}

template <class T>
void List<T>::insertSortedData(const T& element) {
  Position newNode(new Node(element));
  if (newNode == nullptr)
    throw DataContainersExceptions::MemoryOverflow("Memoria No
Disponible");

  Position aux(this->header->getNext());

  while (aux != this->header && aux->getData() < newNode->getData())
    aux = aux->getNext();

  newNode->setPrev(aux);
  newNode->setNext(aux->getNext());

  aux->getNext()->setPrev(newNode);
  aux->setNext(newNode);
  this->totalElements++;
}

template <class T>
void List<T>::deleteData(const typename List<T>::Position& position) {
  if (!this->isValidPosition(position))
    throw DataContainersExceptions::InvalidPosition();

  position->getPrev()->setNext(position->getNext());
  position->getNext()->setPrev(position->getPrev());

  delete position;
  this->totalElements--;
}
```

```
template <class T>
int List<T>::getTotalElements() const {
  return this->totalElements;
}

template <class T>
typename List<T>::Position List<T>::getFirstPos() const {
  return this->isEmpty() ? nullptr : this->header->getNext();
}

template <class T>
typename List<T>::Position List<T>::getLastPos() const {
  return this->isEmpty() ? nullptr : this->header->getPrev();
}

template <class T>
typename List<T>::Position List<T>::getPrevPos(
    const typename List<T>::Position& position) const {
  return position == this->header->getNext() || !this-
>isValidPosition(position)
             ? nullptr
             : position->getPrev();
}

template <class T>
typename List<T>::Position List<T>::getNextPos(
    const typename List<T>::Position& position) const {
  return position == this->header->getPrev() || !this-
>isValidPosition(position)
             ? nullptr
             : position->getNext();
}

template <class T>
typename List<T>::Position List<T>::findData(const T& dataSearched) const
{
  Position aux(this->header->getNext());

  while (aux != this->header) {
    if (aux->getData() == dataSearched)
      return aux;
    aux = aux->getNext();
  }

  return nullptr;
}
```

```
template <class T>
T& List<T>::retrieve(const typename List<T>::Position& p) {
  if (!this->isValidPosition(p))
    throw DataContainersExceptions::InvalidPosition();
  return p->getData();
}

template <class T>
string List<T>::toString() const {
  int cont(0);
  ostringstream oss;
  Position aux(this->header->getNext());

  while (aux != this->header) {
    oss << "| " << std::to_string(cont)
        << std::setw(11 - std::to_string(cont).size()) << ""
        << aux->getData().toString() << "\n";
    cont++;
    aux = aux->getNext();
  }

  return oss.str();
}

template <class T>
string List<T>::toString(const T& searched, int cmp(const T&, const T&))
const {
  int cont(0);
  ostringstream oss;
  Position aux(this->header->getNext());

  while (aux != this->header) {
    if (cmp(searched, aux->getData()) == 0) {
      oss << "| " << std::to_string(cont)
          << std::setw(11 - std::to_string(cont).size()) << ""
          << aux->getData().toString() << "\n";
    }
    cont++;
    aux = aux->getNext();
  }
  return oss.str();
}

template <class T>
void List<T>::deleteAll() {
  Position aux;

  while (this->header->getNext() != this->header) {
```

```
    aux = this->header->getNext();

    this->header->setNext(aux->getNext());
    delete aux;
  }

  this->header->setPrev(this->header);
  this->totalElements = 0;
}

template <class T>
List<T>& List<T>::operator=(const List<T>& other) {
  this->deleteAll();
  this->add(other);
  this->totalElements = other.totalElements;

  return *this;
}

template <class T>
void List<T>::sortDataBubble() {
  bool swapped;
  typename List<T>::Position lptr = this->header;
  do {
    swapped = false;
    typename List<T>::Position ptr = this->header->getNext();
    while (ptr->getNext() != lptr) {
      typename List<T>::Position nxt = ptr->getNext();

      if (ptr->getData() > nxt->getData()) {
        this->swapData(ptr, nxt);
        swapped = true;
      }
      ptr = ptr->getNext();
    }

    lptr = ptr;
  } while (swapped);
}

template <class T>
void List<T>::sortDataInsert() {
  if (this->isEmpty())
    return;

  typename List<T>::Position current = this->header->getNext()-
>getNext();
```

```
    while (current != this->header) {
      T key = current->getData();
      typename List<T>::Position mover = current->getPrev();

      while (mover != this->header && mover->getData() > key) {
        mover->getNext()->setData(mover->getData());
        mover = mover->getPrev();
      }

      mover->getNext()->setData(key);
      current = current->getNext();
    }
}

template <class T>
void List<T>::sortDataSelect() {
  if (this->isEmpty())
    return;

  Position current = this->header->getNext();
  while (current != this->header) {
    Position minNode = current;
    Position scanner = current->getNext();

    while (scanner != this->header) {
      if (scanner->getData() < minNode->getData())
        minNode = scanner;

      scanner = scanner->getNext();
    }

    if (minNode != current)
      this->swapData(minNode, current);

    current = current->getNext();
  }
}

template <class T>
void List<T>::sortDataShell() {
  int series[] = {4181, 2584, 1597, 987, 610, 377, 233, 144, 89, 55,
                  34,   21,   13,   8,   5,   3,   2,   1,   0};

  int size = 0;
  Position temp = this->getFirstPos();
  while (temp != nullptr) {
    size++;
    temp = temp->getNext();
```

```
  }

  int pos = 0;
  int gap = series[pos];

  while (gap > size) {
    gap = series[++pos];
  }

  while (gap > 0) {
    for (int i = gap; i < size; i++) {
      Position nodeI = this->getNodeAt(i);
      T tempData = nodeI->getData();

      int j = i;
      while (j >= gap) {
        Position nodeJMinusGap = this->getNodeAt(j - gap);

        if (nodeJMinusGap->getData() > tempData) {
          Position nodeJ = this->getNodeAt(j);
          nodeJ->setData(nodeJMinusGap->getData());
          j -= gap;
        } else {
          break;
        }
      }

      Position nodeJ = this->getNodeAt(j);
      nodeJ->setData(tempData);
    }

    gap = series[++pos];
  }
}

template <class T>
void List<T>::sortDataBubble(int cmp(const T&, const T&)) {
  if (this->isEmpty() || this->header->getNext()->getNext() == this-
>header)
    return;

  bool swapped;
  typename List<T>::Position lptr = this->header;
  do {
    swapped = false;
    typename List<T>::Position ptr = this->header->getNext();
    while (ptr->getNext() != lptr) {
      typename List<T>::Position nxt = ptr->getNext();
```

```
      if (cmp(ptr->getData(), nxt->getData()) > 0) {
        this->swapData(ptr, nxt);
        swapped = true;
      }
      ptr = ptr->getNext();
    }
    lptr = ptr;
  } while (swapped);
}

template <class T>
void List<T>::sortDataInsert(int cmp(const T&, const T&)) {
  if (this->isEmpty() || this->header->getNext()->getNext() == this-
>header)
    return;

  typename List<T>::Position current = this->header->getNext()-
>getNext();

  while (current != this->header) {
    T key = current->getData();
    typename List<T>::Position mover = current->getPrev();

    while (mover != this->header && cmp(mover->getData(), key) > 0) {
      mover->getNext()->setData(mover->getData());
      mover = mover->getPrev();
    }

    mover->getNext()->setData(key);
    current = current->getNext();
  }
}

template <class T>
void List<T>::sortDataSelect(int cmp(const T&, const T&)) {
  if (this->isEmpty() || this->header->getNext()->getNext() == this-
>header)
    return;

  Position current = this->header->getNext();
  while (current != this->header) {
    Position minNode = current;
    Position scanner = current->getNext();

    while (scanner != this->header) {
      if (cmp(scanner->getData(), minNode->getData()) < 0)
        minNode = scanner;
```

```
      scanner = scanner->getNext();
    }

    if (minNode != current)
      this->swapData(minNode, current);

    current = current->getNext();
  }
}

template <class T>
void List<T>::sortDataShell(int cmp(const T&, const T&)) {
  int series[] = {4181, 2584, 1597, 987, 610, 377, 233, 144, 89, 55,
                  34,   21,   13,   8,   5,   3,   2,   1,   0};

  int size = 0;
  Position temp = this->getFirstPos();
  while (temp != nullptr) {
    size++;
    temp = temp->getNext();
  }

  int pos = 0;
  int gap = series[pos];

  while (gap > size) {
    gap = series[++pos];
  }

  while (gap > 0) {
    for (int i = gap; i < size; i++) {
      Position nodeI = this->getNodeAt(i);
      T tempData = nodeI->getData();

      int j = i;
      while (j >= gap) {
        Position nodeJMinusGap = this->getNodeAt(j - gap);

        if (cmp(nodeJMinusGap->getData(), tempData) > 0) {
          Position nodeJ = this->getNodeAt(j);
          nodeJ->setData(nodeJMinusGap->getData());
          j -= gap;
        } else {
          break;
        }
      }

      Position nodeJ = this->getNodeAt(j);
```

```
        nodeJ->setData(tempData);
      }

      gap = series[++pos];
    }
  }

  template <class T>
  typename List<T>::Position List<T>::getNodeAt(int index) const {
    if (this->totalElements < index || index < 0)
      throw DataContainersExceptions::InvalidPosition();

    Position current = this->header->getNext();
    int count = 0;

    while (current != this->header && count < index) {
      current = current->getNext();
      count++;
    }

    return current;
  }

  template <class X>
  std::ostream& operator<<(std::ostream& os, const List<X>& list) {
    typename List<X>::Position aux = list.header->getNext();

    while (aux != list.header) {
      os << aux->getData();
      aux = aux->getNext();
    }

    return os;
  }

  template <class X>
  std::istream& operator>>(std::istream& is, List<X>& list) {
    X element;
    try {
      while (is >> element) {
        list.insertSortedData(element);
      }
    } catch (const std::invalid_argument& ex) {
    }

    return is;
  }
  #endif  // __LIST_H__
```

menu.cpp

```
#ifndef __MENU_H__
#define __MENU_H__

#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>

#include "list.hpp"
#include "name.hpp"
#include "song.hpp"

class Menu {
 private:
  List<Song>& songList;

  void enterToContinue();
  int readInteger(std::string, const int&, const int&);
  Name readName(std::string);
  std::string readLinePrompt(const std::string&, bool = false);
  char readChar(const std::string&, const char*);

  bool handleOption(const std::string&);
  std::string windowHeader(const int&, const std::string&) const;
  std::string songTable(const int& = 10,
                        const int& = 35,
                        const int& = 30,
                        const int& = 25) const;

  void noDataMessage();

  void mainMenu();
  void insertSong();
  void deleteSong(const int&);
  void deleteAllSongs();
  void editSong(const int&);
  void exitProgram();

  void searchMenu();
  void searchBySongName();
  void searchByIntepreter();

  void sortMenu();
  void sortBySongName();
  void sortByInterpreter();
  void sortByRanking();
```

```
    void saveToDisk();
    void readFromDisk();

  public:
    Menu();
    Menu(const Menu&);
    Menu(List<Song>&);
};

#endif  // __MENU_H__
```

name.hpp

```
#ifndef __NAME_H__
#define __NAME_H__

#include <fstream>
#include <iostream>
#include <string>

#include "ownexceptions.hpp"

class Name {
 private:
  std::string first;
  std::string last;

 public:
  Name();
  Name(const Name&);
  Name(const std::string&, const std::string&);

  // Interfaz
  // Setter's
  void setFirst(const std::string&);
  void setLast(const std::string&);

  // Getter's
  std::string getFirst() const;
  std::string getLast() const;

  std::string toString() const;

  Name& operator=(const Name&);

  bool operator==(const Name&) const;
  bool operator!=(const Name&) const;
  bool operator<(const Name&) const;
  bool operator>(const Name&) const;
  bool operator<=(const Name&) const;
  bool operator>=(const Name&) const;

  int compareTo(const Name&) const;
  int static compare(const Name&, const Name&);

  friend std::ostream& operator<<(std::ostream&, const Name&);
  friend std::istream& operator>>(std::istream&, Name&);
};
#endif  // __NAME_H__
```

ownexceptions.hpp

```
#ifndef __OWNEXCEPTIONS_H__
#define __OWNEXCEPTIONS_H__

#include <stdexcept>
#include <string>

namespace DataContainersExceptions {
class MemoryDeficiency : public std::runtime_error {
 public:
  explicit MemoryDeficiency(const std::string& msg = "Insuficiencia de
Memoria")
      : std::runtime_error(msg) {}
};

class MemoryOverflow : public std::runtime_error {
 public:
  explicit MemoryOverflow(const std::string& msg = "Desbordamiento de
Memoria")
      : std::runtime_error(msg) {}
};

class InvalidPosition : public std::runtime_error {
 public:
  explicit InvalidPosition(
      const std::string& msg = "La posicion Ingresada es Invalida")
      : std::runtime_error(msg) {}
};
}  // namespace DataContainersExceptions

namespace InputExceptions {
class InvalidOption : public std::runtime_error {
 public:
  explicit InvalidOption(
      const std::string& msg = "La opcion ingresada esta fuera de rango")
      : runtime_error(msg) {}
};

class EmptyString : public std::runtime_error {
 public:
  explicit EmptyString(
      const std::string& msg = "El string no puede estar vacio")
      : runtime_error(msg) {}
};

class OperationCanceledException : public std::runtime_error {
```

```
 public:
  explicit OperationCanceledException(
      const std::string msg = "Operacion Cancelada")
      : runtime_error(msg) {}
};

class NumberNotPositive : public std::runtime_error {
 public:
  explicit NumberNotPositive(
      const std::string& msg = "Esta entrada debe ser mayor a 0")
      : runtime_error(msg) {}
};
}  // namespace InputExceptions

namespace DateExceptions {
class InvalidDate : public std::runtime_error {
 public:
  explicit InvalidDate(const std::string& msg = "La fecha es invalida")
      : runtime_error(msg) {}
};
}  // namespace DateExceptions

namespace RecipeExceptions {
class RepeatedIngredient : public std::runtime_error {
 public:
  explicit RepeatedIngredient(
      const std::string& msg = "El ingrediente ya esta registrado en la
lista.")
      : runtime_error(msg) {}
};

class NonExistenIngredient : public std::runtime_error {
 public:
  explicit NonExistenIngredient(
      const std::string& msg = "El ingrediente no existe en la lista.")
      : runtime_error(msg) {}
};
}  // namespace RecipeExceptions

namespace MenuExceptions {
class InvalidInsertCategory : std::runtime_error {
 public:
  explicit InvalidInsertCategory(
      const std::string& msg = "La categoria de inserción es inválida")
      : runtime_error(msg) {}
};
}  // namespace MenuExceptions
```

```
 namespace NodeExceptions {
class NodeExceptionTo : std::runtime_error {
 public:
  explicit NodeExceptionTo(const std::string& msg = "Excepcion del Nodo")
      : runtime_error(msg) {}
};
}  // namespace NodeExceptions

#endif  // __OWNEXCEPTIONS_H__
```

song.hpp

```
#ifndef __SONG_H__
#define __SONG_H__

#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>

#include "name.hpp"

class Song {
 private:
  int ranking;
  std::string songName;
  Name author;
  Name interpreter;
  std::string mp3Name;

 public:
  Song();
  Song(const Song&);

  /// @brief
  /// @param  Ranking
  /// @param  NombreCancion
  /// @param  NombreAutor
  /// @param  NombreInterprete
  /// @param  NombreMP3
  Song(const int&,
       const std::string&,
       const Name&,
       const Name&,
       const std::string&);

  // Interfaz:
  // Setter's
  void setRanking(const int&);
  void setSongName(const std::string&);
  void setAuthor(const Name&);
  void setInterpreter(const Name&);
  void setMp3Name(const std::string&);

  // Getter's
  int getRanking() const;
  std::string getSongName() const;
  Name getAuthor() const;
```

```
    Name getInterpreter() const;
    std::string getMp3Name() const;

    /// @brief Función toString para la lsit.hpp
    /// @param  widthRanking
    /// @param  widthSongName
    /// @param  widthName
    /// @param  widthMP3
    std::string toString(const int& = 10,
                         const int& = 35,
                         const int& = 30,
                         const int& = 25) const;  // Para impresiones en
list.hpp

    /// @brief Función de 1 sola canción
    /// @param widthBorder
    /// @return
    std::string toStringOnly(
        const int& = 60) const;  // Para impresiones de solo 1 canción

    Song& operator=(const Song&);

    // Operadores Relacionales que utilizan el ranking como compardor
    bool operator==(const Song&) const;
    bool operator!=(const Song&) const;
    bool operator<(const Song&) const;
    bool operator>(const Song&) const;
    bool operator<=(const Song&) const;
    bool operator>=(const Song&) const;

    int compareTo(const Song&) const;
    static int compare(const Song&, const Song&);

    static int compareBySongName(const Song&, const Song&);
    static int compareByAutor(const Song&, const Song&);
    static int compareByInterpreter(const Song&, const Song&);
    static int compareByMP3Name(const Song&, const Song&);

    friend std::ostream& operator<<(std::ostream&, const Song&);
    friend std::istream& operator>>(std::istream&, Song&);
};
#endif  // __SONG_H__
```

Carpeta src

main.cpp

```
#include "menu.hpp"

int main() {
  new Menu(*new List<Song>);

  return 0;
}
```

menu.cpp

```
#include "menu.hpp"

using namespace std;

Menu::Menu() : songList(*new List<Song>) {
  mainMenu();
}

Menu::Menu(const Menu& other) : songList(other.songList) {
  mainMenu();
}

Menu::Menu(List<Song>& s) : songList(s) {
  mainMenu();
}

void Menu::enterToContinue() {
  cout << "[Enter] para continuar..." << endl;
  getchar();
}

int Menu::readInteger(string oss,
                      const int& lowerLimit,
                      const int& upperLimit) {
  string aux("");
  int result;
  while (true) {
    try {
      system("CLS");
      cout << oss;
      getline(cin, aux);
      result = stoi(aux);

      if (result > upperLimit || result < lowerLimit)
        throw InputExceptions::InvalidOption("Numero Fuera de Rango");
      break;
    } catch (const std::invalid_argument& ex) {
      system("CLS");
      cout << "Entrada invalida" << endl;
      cout << "Intente nuevamente" << endl;
      enterToContinue();
    } catch (const InputExceptions::InvalidOption& msg) {
      system("CLS");
      cout << msg.what() << endl;
      enterToContinue();
    }
  }
```

```
  return result;
}

Name Menu::readName(string prompt) {
  Name result;
  result.setFirst(readLinePrompt(prompt));
  prompt += result.getFirst() + "\n";
  result.setLast(readLinePrompt(prompt + "Ingrese el Apellido: "));

  return result;
}

string Menu::readLinePrompt(const string& prompt, bool allowEmpty) {
  string result;
  while (true) {
    system("CLS");
    cout << prompt;
    getline(cin, result);
    if (!allowEmpty && result.empty()) {
      system("CLS");
      cout << "No puede estar vacio.\nIntentelo nuevamente." << endl;
      enterToContinue();
      continue;
    }
    return result;
  }
}

char Menu::readChar(const std::string& prompt, const char* posibilities)
{
  char result, comparation;

  while (true) {
    int i = 0;
    system("CLS");
    cout << prompt;
    cin >> result;

    result = toupper(result);
    do {
      comparation = *(posibilities + i);
      if (result == comparation)
        return result;
      i++;
    } while (comparation != '\0');

    system("CLS");
```

```
      cout << "Opcion Invalida" << endl;
      cout << "Intentelo Nuevamente" << endl;
      system("PAUSE");
    }
}

string Menu::windowHeader(const int& widthBorder, const string& prompt)
const {
  ostringstream oss;

  oss << left << setfill('=') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  // Título de Ventana
  oss << setw(widthBorder / 2 - (prompt.size() / 2)) << "| " << prompt
      << setw((widthBorder / 2) - (prompt.size() / 2) - 2) << "" << "|"
<< endl;
  oss << setfill('-') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  return oss.str();
}

bool Menu::handleOption(const std::string& promt) {
  string response;

  system("CLS");
  cout << promt;
  getline(cin, response);

  if (response.empty())
    return true;

  // Hacer las letras mayúsculas
  char option =
      static_cast<char>(std::toupper(static_cast<unsigned
char>(response[0])));

  // buscar primer dígito después de la letra (saltando espacios)
  std::size_t pos = 1;
  while (pos < response.size() &&
         std::isspace(static_cast<unsigned char>(response[pos])))
    ++pos;

  bool hasNumber = false;
  int index = -1;
  if (pos < response.size() &&
      std::isdigit(static_cast<unsigned char>(response[pos]))) {
```

```
    std::size_t start = pos;
    std::size_t end = start;
    while (end < response.size() &&
           std::isdigit(static_cast<unsigned char>(response[end])))
      ++end;
    std::string numstr = response.substr(start, end - start);
    try {
      index = std::stoi(numstr);
      hasNumber = true;
    } catch (...) {
      hasNumber = false;
    }
  }

  switch (option) {
    case 'A':
      this->insertSong();
      break;

    case 'B':
      if (!hasNumber) {
        system("CLS");
        std::cout << "Falta numero de posicion. Ej: B2\n";
        this->enterToContinue();
        break;
      }
      if (this->songList.getTotalElements() <= index || index < 0) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
        this->enterToContinue();
        break;
      }
      this->editSong(index);
      break;

    case 'C':
      if (!hasNumber) {
        system("CLS");
        std::cout << "Falta numero de posicion. Ej: C12\n";
        this->enterToContinue();
        break;
      }
      if (this->songList.getTotalElements() <= index || index < 0) {
        system("CLS");
        std::cout << "Posicion de lista invalida\n";
        this->enterToContinue();
        break;
      }
```

```
    this->deleteSong(index);
    break;

  case 'D':
    if (!hasNumber) {
      system("CLS");
      std::cout << "Falta numero de posiciin. Ej: D12\n";
      this->enterToContinue();
      break;
    }
    if (this->songList.getTotalElements() <= index || index < 0) {
      system("CLS");
      std::cout << "Posicion de lista invalida\n";
      this->enterToContinue();
      break;
    }
    system("CLS");
    {
      List<Song>::Position pos = this->songList.getNodeAt(index);
      if (pos != nullptr) {
        Song& s = this->songList.retrieve(pos);
        std::cout << s.toStringOnly();
      } else {
        std::cout << "Cancion no encontrada\n";
      }
    }
    this->enterToContinue();
    break;
  case 'E':
    this->deleteAllSongs();
    break;
  case 'F':
    this->saveToDisk();
    break;
  case 'G':
    this->readFromDisk();
    break;
  case 'H':
    this->searchMenu();
    break;
  case 'I':
    this->sortMenu();
    break;
  case 'J':
    this->exitProgram();
    return false;

  default:
```

```
        system("CLS");
        std::cout << "Comando invalido\nIntentelo nuevamente.\n";
        enterToContinue();
        break;
    }  // switch

    return true;
}

std::string Menu::songTable(const int& widthRanking,
                            const int& widthSongName,
                            const int& widthName,
                            const int& widthMP3) const {
    ostringstream oss;

    int widthBorder =
        widthRanking + widthSongName + (widthName * 2) + widthMP3 + 16;

    oss << windowHeader(widthBorder, "LISTA DE EXITOS");
    oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking"
        << setw(widthRanking - 7) << "| " << "Nombre de la Cancion"
        << setw(widthSongName - 20) << "| " << "Nombre del Artista"
        << setw(widthName - 18) << "| " << "Nombre del Interprete"
        << setw(widthName - 21) << "| " << "Nombre del MP3" <<
setw(widthMP3 - 14)
        << "|" << endl;

    oss << setfill('-');
    oss << setw(widthBorder) << " ";
    oss << setfill(' ') << endl;

    oss << this->songList.toString();

    oss << setfill('-');
    oss << setw(widthBorder) << " ";
    oss << setfill(' ');
    oss << endl;

    return oss.str();
}

void Menu::noDataMessage() {
    cout << "+-------------------------------------------------------+" <<
endl;
    cout << "+            No hay Canciones Registradas Aun           +" <<
endl;
    cout << "+              Regresando al Menu...                    +" <<
endl;
```

```
    cout << "+-------------------------------------------------------+" <<
endl;
  this->enterToContinue();
}

void Menu::mainMenu() {
  ostringstream oss;
  bool running = true;

  while (running) {
    system("CLS");

    // Limpiar el ostringstream
    oss.str("");
    oss.clear();

    oss << this->songTable();
    oss << "Opciones: \n";
    oss << "[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] "
           "Eliminar "
           "una Cancion. [D<n>]  Mostrar Detalles de
Cancion.  [E]    Eliminar "
           "Todas las Canciones. \n\n"
           "[F] Guardar la Database   [G] Leer del Disco   "
           "[H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.\n\n";
    oss << "Seleccione un Comando: ";

    running = handleOption(oss.str());
  }
}

void Menu::insertSong() {
  int widthBorder = 100;
  Song newSong;
  string myString("");
  int myInt(0);
  Name myName;
  ostringstream oss;

  do {
    system("CLS");
    // Linea Exterior
    oss << windowHeader(widthBorder, "INSERTAR EXITO");

    oss << "Ingrese el Nombre de la Cancion: ";
    myString = this->readLinePrompt(oss.str(), false);
    newSong.setSongName(myString);
    oss << newSong.getSongName() << endl;
```

```
oss << "Ingrese el Ranking de la Cancion: ";

while (true) {
  try {
    system("CLS");
    myInt = readInteger(oss.str(), 1, 9999999);
    newSong.setRanking(myInt);
    if (songList.findData(newSong) != nullptr)
      throw std::invalid_argument("Ranking ya utilizado");
    break;
  } catch (const InputExceptions::InvalidOption& msg) {
    system("CLS");
    cout << msg.what() << endl;
    enterToContinue();
  } catch (const std::invalid_argument& msg) {
    system("CLS");
    cout << msg.what() << endl;
    enterToContinue();
  }
}

oss << newSong.getRanking() << endl;
oss << "Ingrese el Nombre del Autor: ";
myName = readName(oss.str());
oss << myName.getFirst() << endl;
oss << "Ingrese el Apellido: " << myName.getLast() << endl;

newSong.setAuthor(myName);

oss << "Ingrese el Nombre del Interprete: ";
myName = readName(oss.str());
oss << myName.getFirst() << endl;
oss << "Ingrese el Apellido: " << myName.getLast() << endl;

newSong.setInterpreter(myName);

oss << "Ingrese el nombre del Archivo MP3: ";
myString = this->readLinePrompt(oss.str(), false);
newSong.setMp3Name(myString);
oss << newSong.getMp3Name() << endl;

if (songList.isEmpty())
  songList.insertData(nullptr, newSong);
else {
  oss << "Ingrese la posicion en la lista que tendra la cancion: ";
  while (true) {
    try {
```

```
            myInt = readInteger(oss.str(), 0, 3000);
            songList.insertData(this->songList.getNodeAt(myInt), newSong);
            oss << myInt << endl;
            break;
          } catch (const DataContainersExceptions::InvalidPosition& msg) {
            system("CLS");
            cout << msg.what() << endl;
            cout << "Intente Nuevamente." << endl;
            enterToContinue();
          }
        }
      }

      oss << "Cancion Agregada con Exito!." << endl;
      oss << "Desea Agregar Otra Cancion? (1. Si / 2. No): ";

      myInt = readInteger(oss.str(), 1, 2);

      oss.str("");
      oss.clear();
    } while (myInt != 2);
}

void Menu::deleteSong(const int& position) {
    system("CLS");
    ostringstream oss;
    int response;

    List<Song>::Position pos = songList.getNodeAt(position);
    if (pos == nullptr) {
      cout << "Posicion invalida" << endl;
      enterToContinue();
      return;
    }

    Song& target = songList.retrieve(pos);
    oss << target.toStringOnly();
    oss << "Esta seguro que desea eliminar esta cancion? (1. Si/ 2. No): ";
    response = readInteger(oss.str(), 1, 2);

    if (response == 1) {
      songList.deleteData(pos);
      oss << endl << "Cancion Eliminada con Exito!" << endl;
    } else {
      oss << endl << "Operacion Cancelada" << endl;
    }

    system("CLS");
```

```
    cout << oss.str();
  enterToContinue();
}

void Menu::deleteAllSongs() {
  system("CLS");
  if (this->songList.isEmpty()) {
    cout << "Aun no hay canciones para eliminar" << endl;
    enterToContinue();
    return;
  }

  ostringstream oss;
  int widhtBorder = 50;

  oss << windowHeader(widhtBorder, "ELIMINAR TODAS LAS CANCIONES");

  oss << "Esta seguro que desea eliminar las "
      << this->songList.getTotalElements() << " canciones? (1. Si/ 2.
No): ";
  int response = readInteger(oss.str(), 1, 2);
  system("CLS");
  if (response == 1) {
    songList.deleteAll();
    cout << "Canciones eliminadas con Exito!" << endl;
    cout << "Base de Datos Vacia." << endl;
  } else {
    cout << "Operacion Cancelada." << endl;
  }
  enterToContinue();
}

void Menu::editSong(const int& position) {
  ostringstream oss;

  List<Song>::Position pos = songList.getNodeAt(position);
  if (pos == nullptr) {
    system("CLS");
    cout << "Posicion invalida" << endl;
    enterToContinue();
    return;
  }

  Song& target = songList.retrieve(pos);
  int editOption, newRanking;
  string dataString;
  Name newName;
  Song ver;
```

```
    oss << target.toStringOnly();
    oss << "5 Salir\n";

    editOption = readInteger(
        oss.str() + "Elige el atributo que quieras cambiar (1-5): ", 1, 5);

    switch (editOption) {
      case 1:
        oss << "Ingrese el Nuevo Ranking de la Cancion: ";
        newRanking = readInteger(oss.str(), 1, 9999999);
        ver.setRanking(newRanking);
        if (this->songList.findData(ver) != nullptr) {
          system("CLS");
          cout << "El ranking ya esta ocupado" << endl;
          enterToContinue();
          break;
        }
        target.setRanking(newRanking);
        cout << "Cambio hecho con Exito!";
        break;
      case 2:
        oss << "Ingrese el nuevo nombre de la cancion: ";
        dataString = readLinePrompt(oss.str());
        target.setSongName(dataString);
        cout << "Cambio hecho con Exito!";
        enterToContinue();
        break;
      case 3:
        oss << "Ingrese el nuevo autor de la cancion: ";
        newName = readName(oss.str());
        target.setAuthor(newName);
        cout << "Cambio hecho con Exito!";
        break;
      case 4:
        oss << "Ingrese el nuevo interprete de la cancion: ";
        newName = readName(oss.str());
        target.setInterpreter(newName);
        cout << "Cambio hecho con Exito!";
        break;
      case 5:
        return;
      default:
        break;
    }
    enterToContinue();
}
```

```
void Menu::exitProgram() {
  system("CLS");
  int response;
  ostringstream oss;
  if (!this->songList.isEmpty()) {
    oss << windowHeader(50, "SALIR SIN GUARDAR?");
    response = readInteger(
        oss.str() +
            "Desea Guardar las canciones antes de Salir? (1. Si/ 2. No):
",
        1, 2);
    if (response == 1)
      saveToDisk();
  }

  system("CLS");
  std::cout << "Saliendo del Programa.\nTenga un Lindo Dia :D\n";
  enterToContinue();
}

void Menu::searchMenu() {
  system("CLS");
  if (this->songList.isEmpty()) {
    this->noDataMessage();
    return;
  }

  ostringstream oss;
  char op;
  oss << windowHeader(50, "BUSCAR CANCION");
  oss << "Existen un total de: " << this->songList.getTotalElements()
      << " registradas." << endl;
  oss << "A continuacion se muestran las opciones de busqueda: " << endl;
  oss << "[A] Buscar por Nombre de Cancion" << endl
      << "[B] Buscar por Nombre del Inteprete" << endl
      << "[R] Regresar." << endl
      << "Seleccione una Opcion: ";

  while (op != 'R') {
    system("CLS");
    cout << oss.str();
    cin >> op;
    cin.ignore();

    op = toupper(op);

    switch (op) {
      case 'A':
```

```
          this->searchBySongName();
          break;
        case 'B':
          this->searchByIntepreter();
          break;
        case 'R':
          system("CLS");
          cout << "Regresando...";
          enterToContinue();
          break;
        default:
          system("CLS");
          cout << "Opcion invalida" << endl;
          cout << "Intentelo nuevamente" << endl;
          enterToContinue();
          break;
      }
    }
}

void Menu::searchBySongName() {
  ostringstream oss;
  string songName, songsResults;
  int repeat;

  do {
    List<Song> songWithTheName;
    Song searchedSong;

    oss.str("");
    oss.clear();
    system("CLS");
    oss << windowHeader(50, "BUSCAR POR NOMBRE DE CANCION");
    oss << "Ingrese el nombre de la cancion que quiera buscar: ";

    songName = readLinePrompt(oss.str(), false);
    searchedSong.setSongName(songName);

    songsResults =
        this->songList.toString(searchedSong, Song::compareBySongName);

    oss.str("");
    oss.clear();

    if (songsResults.empty()) {
      oss << "\nNo existe un registro de una cancion llamada: " <<
songName
          << endl;
```

```
    } else {
      oss << windowHeader(146, "LISTA DE EXITOS");
      oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking" <<
setw(3)
          << "| " << "Nombre de la Cancion" << setw(15) << "| "
          << "Nombre del Artista" << setw(12) << "| " << "Nombre del
Interprete"
          << setw(9) << "| " << "Nombre del MP3" << setw(11) << "|" <<
endl;

      oss << setfill('-');
      oss << setw(146) << " ";
      oss << setfill(' ') << endl;
      oss << songsResults;

      oss << setfill('-');
      oss << setw(146) << "";
      oss << setfill(' ') << endl;
    }

    oss << "Desea Realizar Otra Busqueda?: (1.Si / 2.No): ";
    repeat = readInteger(oss.str(), 1, 2);

  } while (repeat != 2);

  system("CLS");
  cout << "Regresando..." << endl;
  enterToContinue();
}

void Menu::searchByIntepreter() {
  ostringstream oss;
  Name searchedName;
  int repeat;

  do {
    List<Song> songsOfInterpreter;
    Song searchedSong;

    oss.str("");
    oss.clear();
    system("CLS");
    oss << windowHeader(70, "BUSCAR POR INTERPRETE DE LA CANCION");

    oss << "Ingrese el Nombre del Interprete: ";
    searchedName = readName(oss.str());
    searchedSong.setInterpreter(searchedName);
```

```
    string songsResults;
    songsResults =
        this->songList.toString(searchedSong,
Song::compareByInterpreter);

    oss.str("");
    oss.clear();

    if (songsResults.empty()) {
      oss << "\nNo existe un registro de una cancion del interprete: "
          << searchedName.toString() << endl;
    } else {
      oss << windowHeader(146, "LISTA DE EXITOS");
      oss << "| " << "# En Lista" << setw(3) << "| " << "Ranking" <<
setw(3)
          << "| " << "Nombre de la Cancion" << setw(15) << "| "
          << "Nombre del Artista" << setw(12) << "| " << "Nombre del
Interprete"
          << setw(9) << "| " << "Nombre del MP3" << setw(11) << "|" <<
endl;

      oss << setfill('-');
      oss << setw(146) << " ";
      oss << setfill(' ') << endl;
      oss << songsResults;

      oss << setfill('-');
      oss << setw(146) << "";
      oss << setfill(' ') << endl;
    }

    oss << "Desea Realizar Otra Busqueda?: (1.Si / 2.No): ";
    repeat = readInteger(oss.str(), 1, 2);

  } while (repeat != 2);

  system("CLS");
  cout << "Regresando..." << endl;
  enterToContinue();
}

void Menu::sortMenu() {
  system("CLS");
  if (this->songList.isEmpty()) {
    this->noDataMessage();
    return;
  }
```

```
    if (this->songList.getTotalElements() == 0) {
      cout << "Solo hay 1 cancion registrada." << endl;
      cout << "No se requiere Ordenamiento." << endl;
      enterToContinue();
      return;
    }

    ostringstream oss;
    char op;
    oss << windowHeader(50, "ORDENAR EXITOS");
    oss << "Existen un total de: " << this->songList.getTotalElements()
        << " registradas." << endl;
    oss << "A continuacion se muestran las opciones sobre las cuales
ordenar las "
           "canciones: "
        << endl;
    oss << "[A] Ordenar por Nombre de Cancion" << endl
        << "[B] Ordenar por Nombre del Inteprete" << endl
        << "[C] Ordenar por Numero de Ranking" << endl
        << "[R] Regresar." << endl
        << "Seleccione una Opcion: ";

    while (op != 'R') {
      char charsValid[] = {'A', 'B', 'C', 'R'};
      op = readChar(oss.str(), charsValid);
      cin.ignore();

      switch (op) {
        case 'A':
          this->sortBySongName();
          break;
        case 'B':
          this->sortByInterpreter();
          break;
        case 'C':
          this->sortByRanking();
          break;
        case 'R':
          system("CLS");
          cout << "Regresando...";
          enterToContinue();
          break;
      }
    }
}

void Menu::sortBySongName() {
    system("CLS");
```

```
 ostringstream oss;
  char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

  oss << windowHeader(100, "Ordenar por Nombre de la Cancion");
  oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
      << endl;
  oss << "[A] Ordenamiento por Burbuja" << endl;
  oss << "[B] Ordenamiento por InsertSort" << endl;
  oss << "[C] Ordenamiento por SelectSort" << endl;
  oss << "[D] Ordenamiento por ShellSort (Ordenamiento Sumamente
Ineficiente "
         "en Implementacion Dinamica)"
      << endl;
  oss << "[E] Regresar" << endl << endl;
  oss << "Ingrese una Opcion: ";

  op = this->readChar(oss.str(), validOptions);
  cin.ignore();

  oss << op << endl;
  switch (op) {
    case 'A':
      this->songList.sortDataBubble(Song::compareBySongName);
      oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
      break;
    case 'B':
      this->songList.sortDataInsert(Song::compareBySongName);
      oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
      break;
    case 'C':
      this->songList.sortDataSelect(Song::compareBySongName);
      oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
      break;
    case 'D':
      this->songList.sortDataShell(Song::compareBySongName);
      oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
      break;
    case 'E':
      system("CLS");
      cout << "Regresando..." << endl;
      enterToContinue();
      return;
      break;
  }

  system("CLS");
  oss << endl << "Regresando..." << endl << endl;
```

```
    cout << oss.str();
  enterToContinue();
}

void Menu::sortByInterpreter() {
  system("CLS");
  ostringstream oss;
  char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

  oss << windowHeader(100, "Ordenar por Interprete de la Cancion");
  oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
      << endl;
  oss << "[A] Ordenamiento por Burbuja" << endl;
  oss << "[B] Ordenamiento por InsertSort" << endl;
  oss << "[C] Ordenamiento por SelectSort" << endl;
  oss << "[D] Ordenamiento por ShellSort (Ordenamiento Sumamente
Ineficiente "
         "en Implementacion Dinamica)"
      << endl;
  oss << "[E] Regresar" << endl << endl;
  oss << "Ingrese una Opcion: ";

  op = this->readChar(oss.str(), validOptions);
  cin.ignore();

  oss << op << endl;
  switch (op) {
    case 'A':
      this->songList.sortDataBubble(Song::compareByInterpreter);
      oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
      break;
    case 'B':
      this->songList.sortDataInsert(Song::compareByInterpreter);
      oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
      break;
    case 'C':
      this->songList.sortDataSelect(Song::compareByInterpreter);
      oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
      break;
    case 'D':
      this->songList.sortDataShell(Song::compareByInterpreter);
      oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
      break;
    case 'E':
      system("CLS");
      cout << "Regresando..." << endl;
      enterToContinue();
```

```
        return;
      break;
    }

    system("CLS");
    oss << endl << "Regresando..." << endl << endl;
    cout << oss.str();
    enterToContinue();
}

void Menu::sortByRanking() {
    system("CLS");
    ostringstream oss;
    char op, validOptions[5] = {'A', 'B', 'C', 'D', 'E'};

    oss << windowHeader(100, "Ordenar por Ranking de la Cancion");
    oss << "A continuacion, eliga un algoritmo de ordenamiento para la
lista: "
        << endl;
    oss << "[A] Ordenamiento por Burbuja" << endl;
    oss << "[B] Ordenamiento por InsertSort" << endl;
    oss << "[C] Ordenamiento por SelectSort" << endl;
    oss << "[D] Ordenamiento por ShellSort (Ordenamiento Sumamente
Ineficiente "
           "en Implementacion Dinamica)"
        << endl;
    oss << "[E] Regresar" << endl << endl;
    oss << "Ingrese una Opcion: ";

    op = this->readChar(oss.str(), validOptions);
    cin.ignore();

    oss << op << endl;
    switch (op) {
      case 'A':
        this->songList.sortDataBubble();
        oss << "Ordenamiento por Burbuja hecho correctamente!." << endl;
        break;
      case 'B':
        this->songList.sortDataInsert();
        oss << "Ordenamiento por InserSor hecho correctamente!" << endl;
        break;
      case 'C':
        this->songList.sortDataSelect();
        oss << "Ordenamiento por DataSelect hecho correctamente!" << endl;
        break;
      case 'D':
        this->songList.sortDataShell();
```

```
        oss << "Ordenamiento por ShellSort hecho correctamente!" << endl;
        break;
      case 'E':
        system("CLS");
        cout << "Regresando..." << endl;
        enterToContinue();
        return;
        break;
    }

    system("CLS");
    oss << endl << "Regresando..." << endl << endl;
    cout << oss.str();
    enterToContinue();
}

void Menu::saveToDisk() {
    system("CLS");
    ostringstream oss;

    if (this->songList.isEmpty()) {
      cout << "+-----------------------------------------------------+"
<< endl;
      cout << "+            No hay Canciones Registradas Aun         +"
<< endl;
      cout << "+              Regresando al Menu...                  +"
<< endl;
      cout << "+-----------------------------------------------------+"
<< endl;
      enterToContinue();
      return;
    }

    int widthBorder = 50;
    string fileName("");
    ofstream file;

    oss << windowHeader(widthBorder, "GUARDAR DATABASE");

    oss << "Ingrese el Nombre que Tendra el Archivo: ";
    fileName = readLinePrompt(oss.str());

    file.open(fileName, ios_base::trunc);

    if (!file.is_open())
      oss << "No se permite la creacion de archivos." << endl;
    else {
      file << this->songList;
```

```
    oss << "Database guardada con Exito!" << endl;
  }

  system("CLS");
  cout << oss.str();
  enterToContinue();
}

void Menu::readFromDisk() {
  system("CLS");
  ostringstream oss;
  int widthBorder = 100;
  ifstream file;
  string fileName;

  oss << windowHeader(widthBorder, "LEER ARCHIVO");

  oss << "Tenga en Cuenta que los Archivos se Sobreescribiran" << endl;
  oss << "Ingrese el Nombre del Archivo a Cargar sus Datos: ";

  fileName = readLinePrompt(oss.str());
  oss << fileName << endl;

  file.open(fileName);

  if (!file.is_open())
    oss << "El archivo no existe o no pudo ser abierto" << endl;
  else {
    this->songList.deleteAll();
    file >> this->songList;
    oss << "Archivos Cargados Con Exito!" << endl;
  }

  oss << setfill('=') << setw(widthBorder) << "" << endl;
  oss << setfill(' ');

  system("CLS");
  cout << oss.str();
  enterToContinue();
}
```

name.cpp

```
#include "name.hpp"

Name::Name() : first("default"), last("default") {}

Name::Name(const Name& other) : first(other.first), last(other.last) {}

Name::Name(const std::string& f, const std::string& l) : first(f),
last(l) {}

void Name::setFirst(const std::string& first) {
  if (first.empty())
    throw InputExceptions::EmptyString(
        "Nombre no puede estar vacío, setFirst(Name)");
  this->first = first;
}

void Name::setLast(const std::string& last) {
  if (last.empty())
    throw InputExceptions::EmptyString(
        "Apellido no puede estar vacío, setLast(Name)");
  this->last = last;
}

std::string Name::getFirst() const {
  return this->first;
}

std::string Name::getLast() const {
  return this->last;
}

std::string Name::toString() const {
  return this->first + " " + this->last;
}

Name& Name::operator=(const Name& other) {
  this->first = other.first;
  this->last = other.last;

  return *this;
}

bool Name::operator==(const Name& other) const {
  return this->toString() == other.toString();
}

bool Name::operator!=(const Name& other) const {
```

```
    return !(*this == other);
}

bool Name::operator<(const Name& other) const {
  return this->toString() < other.toString();
}

bool Name::operator>(const Name& other) const {
  return this->toString() > other.toString();
}

bool Name::operator<=(const Name& other) const {
  return (*this < other) || (*this == other);
}

bool Name::operator>=(const Name& other) const {
  return (*this > other) || (*this == other);
}

int Name::compareTo(const Name& other) const {
  return this->toString().compare(other.toString());
}

int Name::compare(const Name& nameA, const Name& nameB) {
  return nameA.toString().compare(nameB.toString());
}

std::ostream& operator<<(std::ostream& os, const Name& name) {
  os << name.first << "," << name.last;

  return os;
}

std::istream& operator>>(std::istream& is, Name& name) {
  std::string dataString;
  getline(is, dataString, ',');
  name.first = dataString;
  getline(is, dataString, ',');
  name.last = dataString;

  return is;
}
```

song.cpp

```
#include "song.hpp"

using namespace std;

Song::Song()
    : ranking(-1),
      songName("default"),
      author(),
      interpreter(),
      mp3Name("default") {}

Song::Song(const Song& other)
    : ranking(other.ranking),
      songName(other.songName),
      author(other.author),
      interpreter(other.interpreter),
      mp3Name(other.mp3Name) {}

Song::Song(const int& r,
           const std::string& n,
           const Name& a,
           const Name& i,
           const std::string& m)
    : ranking(r), songName(n), author(a), interpreter(i), mp3Name(m) {}

void Song::setRanking(const int& ranking) {
  if (ranking <= 0)
    throw InputExceptions::InvalidOption("El ranking debe ser positivo");
  this->ranking = ranking;
}

void Song::setSongName(const std::string& songName) {
  if (songName.empty())
    throw InputExceptions::EmptyString("El nombre no puede estar
vacio.");
  this->songName = songName;
}

void Song::setAuthor(const Name& author) {
  this->author = author;  // Name tiene sus propias validaciones
}

void Song::setInterpreter(const Name& interpreter) {
  this->interpreter = interpreter;
}

void Song::setMp3Name(const std::string& mp3Name) {
```

```
    if (mp3Name.empty())
      throw InputExceptions::EmptyString("El nombre no puede estar vacio");
    this->mp3Name = mp3Name;
}

int Song::getRanking() const {
    return this->ranking;
}

std::string Song::getSongName() const {
    return this->songName;
}

Name Song::getAuthor() const {
    return this->author;
}

Name Song::getInterpreter() const {
    return this->interpreter;
}

std::string Song::getMp3Name() const {
    return this->mp3Name;
}

std::string Song::toStringOnly(const int& widthBorder) const {
    ostringstream oss;

    oss << setfill('=') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    oss << "| " << setw((widthBorder / 2) + 10) << "INFORMACION DE LA
CANCION"
        << setw((widthBorder / 2) - 12) << "|" << endl;

    oss << setfill('-') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    oss << "Posicion en el Ranking: " << ranking << endl;
    oss << "Nombre de la Cancion: " << songName << endl;
    oss << "Nombre del Autor: " << author.toString() << endl;
    oss << "Nombre del Inteprete: " << interpreter.toString() << endl;
    oss << "Nombre del Archivo MP3: " << mp3Name << endl;

    oss << endl << setfill('=') << setw(widthBorder) << "" << endl;
    oss << setfill(' ');

    return oss.str();
```

```
}

std::string Song::toString(const int& widthRanking,
                           const int& widthSongName,
                           const int& widthName,
                           const int& widthMP3) const {
  ostringstream oss;

  oss << "| " << this->ranking
      << setw(widthRanking - to_string(this->ranking).size()) << "| "
      << this->songName << setw(widthSongName - this->songName.size()) <<
"| "
      << this->author.toString()
      << setw(widthName - this->author.toString().size()) << "| "
      << this->interpreter.toString()
      << setw(widthName - this->interpreter.toString().size()) << "| "
      << this->mp3Name << setw(widthMP3 - this->mp3Name.size()) << "|";

  return oss.str();
}

Song& Song::operator=(const Song& other) {
  this->ranking = other.ranking;
  this->songName = other.songName;
  this->author = other.author;
  this->interpreter = other.interpreter;
  this->mp3Name = other.mp3Name;

  return *this;
}

bool Song::operator==(const Song& other) const {
  return this->ranking == other.ranking;
}

bool Song::operator!=(const Song& other) const {
  return !(*this == other);
}

bool Song::operator<(const Song& other) const {
  return this->ranking < other.ranking;
}

bool Song::operator>(const Song& other) const {
  return this->ranking > other.ranking;
}

bool Song::operator<=(const Song& other) const {
```

```
    return !(*this > other);
}

bool Song::operator>=(const Song& other) const {
  return !(*this < other);
}

int Song::compareTo(const Song& other) const {
  return this->ranking - other.ranking;
}

int Song::compare(const Song& songA, const Song& songB) {
  return songA.ranking - songB.ranking;
}

int Song::compareBySongName(const Song& songA, const Song& songB) {
  return songA.songName.compare(songB.songName);
}

int Song::compareByAutor(const Song& songA, const Song& songB) {
  return songA.author.compareTo(songB.author);
}

int Song::compareByInterpreter(const Song& songA, const Song& songB) {
  return songA.interpreter.compareTo(songB.interpreter);
}

int Song::compareByMP3Name(const Song& songA, const Song& songB) {
  return songA.mp3Name.compare(songB.mp3Name);
}

std::ostream& operator<<(std::ostream& os, const Song& song) {
  os << song.ranking << "," << song.songName << "," << song.author << ","
     << song.interpreter << "," << song.mp3Name;

  return os;
}
std::istream& operator>>(std::istream& is, Song& song) {
  string dataString;
  getline(is, dataString, ',');
  song.ranking = stoi(dataString);
  getline(is, song.songName, ',');
  is >> song.author;
  is >> song.interpreter;
  getline(is, song.mp3Name);

  return is;
}
```

Ejecución del Programa

La ejecución de este programa para el usuario no cambia, así que las tomas de pantalla serán similares a tareas anteriores, pero muestran que el programa funciona adecuadamente con las modificaciones hechas en la lista.

La ejecución del menú principal al iniciar el programa

```
================================================================================================================================
|                                                    LISTA DE EXITOS                                                           |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
--------------------------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

Agregar una canción sigue teniendo éxito:

```
================================================================================================================================
|                                                  INSERTAR EXITO                                                              |
--------------------------------------------------------------------------------------------------------------------------------
Ingrese el Nombre de la Cancion: Campeones del Mundo
Ingrese el Ranking de la Cancion: 1
Ingrese el Nombre del Autor: José
Ingrese el Apellido: Madero
Ingrese el Nombre del Interprete: José
Ingrese el Apellido: Madero
Ingrese el nombre del Archivo MP3: CampMunJMV.mp3
Cancion Agregada con Exito!.
Desea Agregar Otra Cancion? (1. Si / 2. No): |
```

Y su reflejo en el menú principal sí se ve:

```
================================================================================================================================
|                                                    LISTA DE EXITOS                                                           |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
--------------------------------------------------------------------------------------------------------------------------------
| 0          | 1       | Campeones del Mundo             | José Madero                 | José Madero                 | CampMunJMV.mp3          |
--------------------------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

Se sigue siendo compatible el mismo formato de archivo para lectura que veníamos manejando con anterioridad:

```
================================================================================================================================
|                                                   LEER ARCHIVO                                                               |
--------------------------------------------------------------------------------------------------------------------------------
Tenga en Cuenta que los Archivos se Sobreescribiran
Ingrese el Nombre del Archivo a Cargar sus Datos: database.csv
Archivos Cargados Con Exito!
================================================================================================================================
[Enter] para continuar...
```

Que sí se reflejan correctamente:

LISTA DE EXITOS

# En Lista	Ranking	Nombre de la Cancion	Nombre del Artista	Nombre del Interprete	Nombre del MP3
0	235	Karmadame	León Larregui	Zoé Zoé	karmadame_zoe.mp3,
1	302	Luna	León Larregui	Zoé Zoé	luna_zoe.mp3,
2	315	Labios Rotos	León Larregui	Zoé Zoé	labiosrotos_zoe.mp3,
3	456	Narcisista por Excelencia	José Madero	José Madero	narc_ex_jm.mp3,
4	789	El Duelo	León Larregui	Zoé Zoé	elduelo_zoe.mp3,
5	904	Los Malaventurados No Lloran	José Madero	José Madero	malav_jm.mp3,
6	1023	Mercedes	José Madero	José Madero	mercedes_jm.mp3,
7	1222	Rayo de Luz	José Madero	José Madero	rayoluz_jm.mp3,
8	1421	Fiebre	León Larregui	Zoé Zoé	fiebre_zoe.mp3,
9	1502	No Hay Mal Que Dure	León Larregui	Zoé Zoé	nomal_dure_zoe.mp3,
10	1551	Love	León Larregui	Zoé Zoé	love_zoe.mp3,
11	1582	Nada Personal	Luis Hernández	Luis Hernández	nada_pers.mp3,
12	1673	Día de Mayo	José Madero	José Madero	diademayo_jm.mp3,
13	1678	Tokyo	Laura Pergolizzi	LP Pergolizzi	tokyo_lp.mp3,
14	1742	Monster (Zoé cover)	León Larregui	Zoé Zoé	monster_zoe.mp3,
15	1755	Hablemos del Campo	José Madero	José Madero	habcamp_jm.mp3,
16	1760	Earned It	Abel Tesfaye	The Weeknd	earned_tw.mp3,
17	1764	Save Your Tears	Abel Tesfaye	The Weeknd	save_tw.mp3,
18	1876	Quince Mil Días	José Madero	José Madero	quincemil_jm.mp3,
19	1888	Monster	Abel Tesfaye	The Weeknd	monster_tw.mp3,
20	1901	Gasolina (cover)	Laura Pergolizzi	LP Pergolizzi	gasolina_lp.mp3,
21	1977	Belong To The World (cover)	Laura Pergolizzi	LP Pergolizzi	belong_lp.mp3,
22	1987	Lunes 28	José Madero	José Madero	lunes28_jm.mp3,
23	1988	Callaíta (cover)	Abel Tesfaye	The Weeknd	callaita_tw.mp3,
24	1989	Lost On You	Laura Pergolizzi	LP Pergolizzi	lost_lp.mp3,
25	2022	Midnight City (cover)	Laura Pergolizzi	LP Pergolizzi	midnight_lp.mp3,
26	2099	Gardenias 87	José Madero	José Madero	gardenias_jm.mp3,
27	2111	Strange	Laura Pergolizzi	LP Pergolizzi	strange_lp.mp3,
28	2115	Nueve Vidas	José Madero	José Madero	nuevevidas_jm.mp3,
29	2122	The Hills	Abel Tesfaye	The Weeknd	hills_tw.mp3,
30	2177	Hit Me Hard	The Weeknd	The Weeknd	hitme_tw.mp3,
31	2203	Campeones del Mundo	José Madero	José Madero	campeones_jm.mp3,
32	2228	Home	The Weeknd	The Weeknd	home_tw.mp3,
33	2244	Popular	León Larregui	Zoé Zoé	popular_zoe.mp3,
34	2250	In Your Eyes	Abel Tesfaye	The Weeknd	inyoureyes_tw.mp3,
35	2307	What You Need	The Weeknd	The Weeknd	whatneed_tw.mp3,

Editar cualquier canción sigue funcionando:

```
==========================================================
|                INFORMACION DE LA CANCION                |
----------------------------------------------------------
Posicion en el Ranking: 1755
Nombre de la Cancion: Hablemos del Campo
Nombre del Autor: José Madero
Nombre del Inteprete: José Madero
Nombre del Archivo MP3: habcamp_jm.mp3,

==========================================================
5 Salir
Elige el atributo que quieras cambiar (1-5):
```

```
==========================================================
|                INFORMACION DE LA CANCION                |
----------------------------------------------------------
Posicion en el Ranking: 1755
Nombre de la Cancion: Hablemos del Campo
Nombre del Autor: José Madero
Nombre del Inteprete: José Madero
Nombre del Archivo MP3: habcamp_jm.mp3,

==========================================================
5 Salir
Ingrese el Nuevo Ranking de la Cancion: 5
Cambio hecho con Exito![Enter] para continuar...
```

Que sí se reflejan en el menú:

```
34 | 5    | Hablemos del Campo | José Madero | José Madero | habcamp_jm.mp3,
35 | 1987 | Lunes 28           | José Madero | José Madero | lunes28_jm.mp3,
36 | 2244 | Popular            | León Larregui | Zoé Zoé   | popular_zoe.mp3,
37 | 315  | Labios Rotos       | León Larregui | Zoé Zoé   | labiosrotos_zoe.mp3,
38 | 2228 | Home               | The Weeknd  | The Weeknd  | home_tw.mp3,
39 | 1502 | No Hay Mal Que Dure | León Larregui | Zoé Zoé | nomal_dure_zoe.mp3,
```

Mostrar la información de una canción sigue mostrándose como debería:

```
============================================================
|                  INFORMACION DE LA CANCION                |
------------------------------------------------------------
Posicion en el Ranking: 2420
Nombre de la Cancion: Dafne
Nombre del Autor: José Madero
Nombre del Inteprete: José Madero
Nombre del Archivo MP3: dafne_jm.mp3,

============================================================
[Enter] para continuar...
```

El guardado al disco:

```
================================================
|                GUARDAR DATABASE              |
------------------------------------------------
Ingrese el Nombre que Tendra el Archivo: Database guardada con Exito!
[Enter] para continuar...
```

La búsqueda:

```
================================================
|                 BUSCAR CANCION               |
------------------------------------------------
Existen un total de: 45 registradas.
A continuacion se muestran las opciones de busqueda:
[A] Buscar por Nombre de Cancion
[B] Buscar por Nombre del Inteprete
[R] Regresar.
Seleccione una Opcion:
```

```
================================================================================
|                    BUSCAR POR INTERPRETE DE LA CANCION                       |
--------------------------------------------------------------------------------
Ingrese el Nombre del Interprete: LP
Ingrese el Apellido: Pergolizzi
```

```
================================================================================================================================================
|                                                                LISTA DE EXITOS                                                               |
------------------------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
------------------------------------------------------------------------------------------------------------------------------------------------
| 10         | 1901    | Gasolina (cover)                | Laura Pergolizzi            | LP Pergolizzi               | gasolina_lp.mp3,        |
| 11         | 1977    | Belong To The World (cover)     | Laura Pergolizzi            | LP Pergolizzi               | belong_lp.mp3,          |
| 16         | 1678    | Tokyo                           | Laura Pergolizzi            | LP Pergolizzi               | tokyo_lp.mp3,           |
| 17         | 2111    | Strange                         | Laura Pergolizzi            | LP Pergolizzi               | strange_lp.mp3,         |
| 18         | 1989    | Lost On You                     | Laura Pergolizzi            | LP Pergolizzi               | lost_lp.mp3,            |
| 19         | 2022    | Midnight City (cover)           | Laura Pergolizzi            | LP Pergolizzi               | midnight_lp.mp3,        |
------------------------------------------------------------------------------------------------------------------------------------------------
Desea Realizar Otra Busqueda?: (1.Si / 2.No):
```

El ordenamiento:

```
================================================
|                 ORDENAR EXITOS               |
------------------------------------------------
Existen un total de: 45 registradas.
A continuacion se muestran las opciones sobre las cuales ordenar las canciones:
[A] Ordenar por Nombre de Cancion
[B] Ordenar por Nombre del Inteprete
[C] Ordenar por Numero de Ranking
[R] Regresar.
Seleccione una Opcion:
```

```
================================================================================================================================================
|                                                    Ordenar por Nombre de la Cancion                                                          |
------------------------------------------------------------------------------------------------------------------------------------------------
A continuacion, eliga un algoritmo de ordenamiento para la lista:
[A] Ordenamiento por Burbuja
[B] Ordenamiento por InsertSort
[C] Ordenamiento por SelectSort
[D] Ordenamiento por ShellSort (Ordenamiento Sumamente Ineficiente en Implementacion Dinamica)
[E] Regresar

Ingrese una Opcion:
```

Que en el menú si se ven ordenados, en este caso por nombre de la canción:

```
================================================================================================================================================
|                                                                LISTA DE EXITOS                                                               |
------------------------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete       | Nombre del MP3          |
------------------------------------------------------------------------------------------------------------------------------------------------
| 8          | 2999    | After Hours                     | Abel Tesfaye                | The Weeknd                  | afterh_tw.mp3,          |
| 1          | 1977    | Belong To The World (cover)     | Laura Pergolizzi            | LP Pergolizzi               | belong_lp.mp3,          |
| 2          | 2345    | Blinding Lights                 | Abel Tesfaye                | The Weeknd                  | blindin_tw.mp3,         |
| 3          | 1988    | Callaita (cover)                | Abel Tesfaye                | The Weeknd                  | callaita_tw.mp3,        |
| 4          | 2203    | Campeones del Mundo             | José Madero                 | José Madero                 | campeones_jm.mp3,       |
| 5          | 2428    | Dafne                           | José Madero                 | José Madero                 | dafne_jm.mp3,           |
| 6          | 1673    | Día de Mayo                     | José Madero                 | José Madero                 | diademayo_jm.mp3,       |
| 7          | 1760    | Earned It                       | Abel Tesfaye                | The Weeknd                  | earned_tw.mp3,          |
| 8          | 789     | El Duelo                        | León Larregui               | Zoé Zoé                     | elduelo_zoe.mp3,        |
| 9          | 1421    | Fiebre                          | León Larregui               | Zoé Zoé                     | fiebre_zoe.mp3,         |
| 10         | 2433    | Friends                         | Abel Tesfaye                | The Weeknd                  | friends_tw.mp3,         |
| 11         | 2320    | Gardenias                       | José Madero                 | José Madero                 | gardenias2_jm.mp3,      |
| 12         | 2099    | Gardenias 87                    | José Madero                 | José Madero                 | gardenias_jm.mp3,       |
| 13         | 1901    | Gasolina (cover)                | Laura Pergolizzi            | LP Pergolizzi               | gasolina_lp.mp3,        |
| 14         | 1       | Hablemos del Campo              | José Madero                 | José Madero                 | habcamp_jm.mp3,         |
| 15         | 2410    | Heartless                       | Abel Tesfaye                | The Weeknd                  | heartless_tw.mp3,       |
| 16         | 2177    | Hit Me Hard                     | The Weeknd                  | The Weeknd                  | hitme_tw.mp3,           |
| 17         | 2228    | Home                            | The Weeknd                  | The Weeknd                  | home_tw.mp3,            |
| 18         | 2250    | In Your Eyes                    | Abel Tesfaye                | The Weeknd                  | inyoureyes_tw.mp3,      |
| 19         | 235     | Karmadame                       | León Larregui               | Zoé Zoé                     | karmadame_zoe.mp3,      |
| 20         | 315     | Labios Rotos                    | León Larregui               | Zoé Zoé                     | labiosrotos_zoe.mp3,    |
| 21         | 2860    | Liminal (Zoé)                   | León Larregui               | Zoé Zoé                     | liminal_zoe.mp3,        |
| 22         | 906     | Los Malaventurados No Lloran    | José Madero                 | José Madero                 | malav_jm.mp3,           |
| 23         | 1989    | Lost On You                     | Laura Pergolizzi            | LP Pergolizzi               | lost_lp.mp3,            |
| 24         | 1551    | Love                            | León Larregui               | Zoé Zoé                     | love_zoe.mp3,           |
| 25         | 302     | Luna                            | León Larregui               | Zoé Zoé                     | luna_zoe.mp3,           |
| 26         | 1987    | Lunes 28                        | José Madero                 | José Madero                 | lunes28_jm.mp3,         |
| 27         | 1023    | Mercedes                        | José Madero                 | José Madero                 | mercedes_jm.mp3,        |
| 28         | 2022    | Midnight City (cover)           | Laura Pergolizzi            | LP Pergolizzi               | midnight_lp.mp3,        |
| 29         | 1888    | Monster                         | Abel Tesfaye                | The Weeknd                  | monster_tw.mp3,         |
| 30         | 1762    | Monster (Zoé cover)             | León Larregui               | Zoé Zoé                     | monster_zoe.mp3,        |
| 31         | 1582    | Nada Personal                   | Luis Hernández              | Luis Hernández              | nada_pers.mp3,          |
| 32         | 456     | Narcisista por Excelencia       | José Madero                 | José Madero                 | narc_ex_jm.mp3,         |
| 33         | 1502    | No Hay Mal Que Dure             | León Larregui               | Zoé Zoé                     | nomal_dure_zoe.mp3,     |
| 34         | 2115    | Nuevo Vidas                     | José Madero                 | José Madero                 | nuevovidas_jm.mp3,      |
| 35         | 2248    | Popular                         | León Larregui               | Zoé Zoé                     | popular_zoe.mp3,        |
```

La eliminación completa:

```
================================================
|          ELIMINAR TODAS LAS CANCIONES        |
------------------------------------------------
Esta seguro que desea eliminar las 45 canciones? (1. Si/ 2. No): 1
```

```
================================================================================================================================
|                                                    LISTA DE EXITOS                                                           |
--------------------------------------------------------------------------------------------------------------------------------
| # En Lista | Ranking | Nombre de la Cancion            | Nombre del Artista          | Nombre del Interprete      | Nombre del MP3         |
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Opciones:
[A] Agregar una Cancion.  [B<n>] Editar una Cancion [C<n>] Eliminar una Cancion. [D<n>]  Mostrar Detalles de Cancion.  [E]    Eliminar Todas las Canciones.

[F] Guardar la Database   [G] Leer del Disco   [H] Buscar una Cancion    [I] Ordenar Lista  [J] Salir.

Seleccione un Comando:
```

En general, el programa tiene el comportamiento esperado

Conclusiones

Todo el mundo de las listas es bastante más basto de lo que un inicio podíamos pensar, ya hemos pasado por las implementaciones estáticas y ahora que estamos en las dinámicas con todas las propiedades que ello tiene, tenemos otros submundos en los cuales estamos adentrándonos, una lista doblemente ligada tiene facilidades importantes, el mero hecho de poder acceder de manera inmediata al elemento anterior sin tener que realizar otro recorrido de lista es importante, sobre todo en conjuntos de datos muy grandes, donde cada uno de estos recorridos es sumamente costoso, y la idea de un encabezado tipo "Dummy", en un inicio pensaba que era un poco inusual, pensaría que sería mejor simplemente tener un atributo al primer elemento y que ese sea la referencia del "inicio" (si se le puede llamar así a una lista circular), pero al ver como simplifica el trabajo y como no aisla los casos para operaciones fundamentales sino que es una generalización que se adapta sorprendentemente bien, y usualmente las implementaciones más simples son las más eficaces, parece seguir el principio de diseño "KISS" y parece que se adecua bastante bien.

Sin embargo, claro está, no podemos universalizar solo un tipo de lista para la solución informática de todo tipo de problemas, conocer varias alternativas y como programadores que inician su proceso desde el planteamiento y no desde el tecleo, debemos saber elegir cuál lista es la que mejor puede aprovecharse o la que más conviene utilizar para cada tipo de problemas, en este caso, la adición de un atributo al encabezado que indique la cantidad de elementos para no hacer un recorrido (ya que no disponemos de un atributo "last" como en las estáticas), fue útil; en otros casos, tener un atributo por ejemplo, para el último elemento y evitar más recorridos. E incluso, se me ocurre que podría ser interesante tener en el encabezado un tipo de dato adicional que correlacione un número de índice con la dirección de memoria que se almacene, para gozar de las bondades de la memoria dinámica y hacer que no sea tan tontería un algoritmo shellSort eficiente para listas dinámicas o muchas cosas más, no hay límite para lo que podamos conseguir con las listas y es tan emocionante pensar en todas estas cosas.