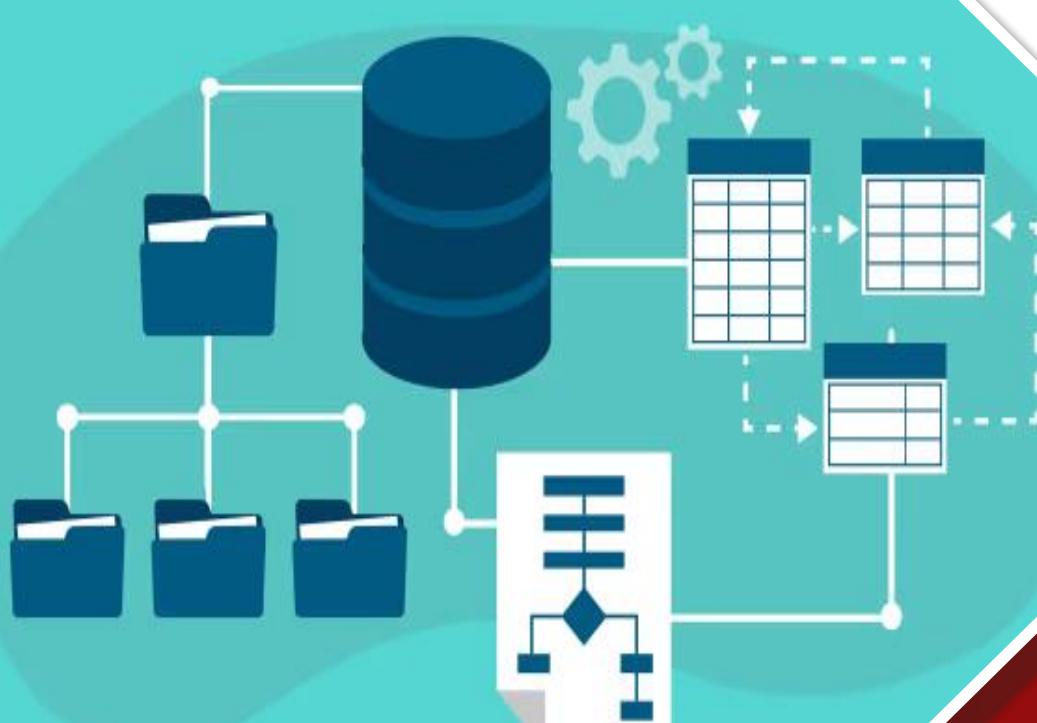


Proyecto Final: Entrega Preliminar**RECETARIO DIGITAL**

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: 19 de Octubre de 2025





Contenido

Test de Autoevalación	3
Introducción.....	4
Diagrama de clases UML	6
Código Fuente	7
Carpeta include	7
basemenu.hpp	7
Category.hpp	9
Configure.hpp	10
date.hpp	11
ingredient.hpp	13
Ingredientmenu.hpp	15
list.hpp	16
menu.hpp	34
name.hpp	36
ownexceptions.hpp	38
proceduremenu.hpp	41
recipe.hpp	42
stringwrapped.hpp	45
Carpeta lib	50
Json.hpp	50
Carpeta src	51
basemenu.cpp	51
date.cpp.....	59
ingredient.cpp	62
ingredientmenu.cpp	65
main.cpp	72
menu.cpp	73
name.cpp.....	96
proceduremenu.cpp.....	98
recipe.cpp.....	104
stringwrapped.cpp	111
Conclusiones.....	125



Test de Autoevalación

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 <i>en formato de texto (sólo si funciona cumpliendo todos los requerimientos)</i>	+25 pts	0 pts	25
Incluí las <i> impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)</i>	+25 pts	0 pts	25
Incluí una <i>portada</i> que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25
Incluí una <i>descripción y conclusiones</i> de mi trabajo	+25 pts	0 pts	25
Suma:			100



Introducción

A continuación, se muestra el desarrollo de la entrega preliminar el proyecto que es realizar un recetario digital, este proyecto reúne conceptos de programación orientada a objetos, estructuras de datos, manejo de memoria y soporte de persistencia en disco, todos aquellos que hemos visto hasta la fecha en el lenguaje de C++.

Para su desarrollo, se implementaron clases que representan los elementos principales del problema, como Receta, Ingrediente y Nombre, junto con listas encargadas de almacenar los distintos objetos, que, dadas las indicaciones, se optó por una lista que contenía apuntadores, de esta manera permitiendo que el arreglo se creé de forma dinámica. En la versión preliminar, se utilizaron arreglos estáticos para almacenar dichos punteros, como el primer parte del semestre, adaptando y mejorando cada vez más la implementación del modelo de la lista para permitir tener mucho control (y siendo lo más eficientes posibles en ello) sobre los elementos que ahí almacena.

Se trató con cuidado el hecho de la inserción ordenada a la lista (no ordenarla después de insertar), y esto se combinó con el ordenamiento, es decir, la lista de los ingredientes, a priori, se ordena mediante un identificador único (ID: esto nos permite tener recetas con nombres repetido y con diferentes procedimientos u autores); pero en el menú de ordenamiento, el usuario puede seleccionar si se van a ordenar por el ID, el nombre de la receta, autor, fecha de creación o cualquiera de sus atributos, esto modifica internamente el orden de los punteros (no de los objetos, así hacemos mucho más laxo el costo computacional de un swapData para el ordenamiento), así que el orden es diferente, por el programa, como un atributo de menú tiene que ser capaz de recordar en base a qué atributo se está ordenando, y según aquello, hacer la inserción ordenada (esto también es equivalente si se carga una base de datos, esta ya tiene que tener un parámetro sobre el cuál esté ordenando, por lo que mediante una función de la lista denominada isSorted() y comparadores explícitos, captamos esto desde que se carga).

El manejo de archivos se realiza mediante el formato json, permitiendo mayor compatibilidad, legibilidad mayor mantenibilidad e incluso, posibles integraciones de API's en un formato más estándar; tanto la lectura como la escritura al disco es mediante este formato. Se le agregó también la posibilidad de poder exportar una receta en un



formato diferente como texto plano, así, el usuario puede compartir sus recetas en algo como un .txt y no tener que transcribir o tomar una captura directamente.

Para tener una mejor presentación, y una mayor afinidad con el usuario, el apartado estético se realizó con el formato UTF-8, colores e incluso emojis, de esta manera, la presentación es mucho más agradable a la vista. Para este código de colores, se agregó a ciertas cadenas de caracteres, un código que indica el color en el que se tiene que imprimir, y para no repetir esto a lo largo del código, se realizó una función que inserta directamente este código de color en un texto dado según una elección que el programador haga mediante un parámetro “string”.

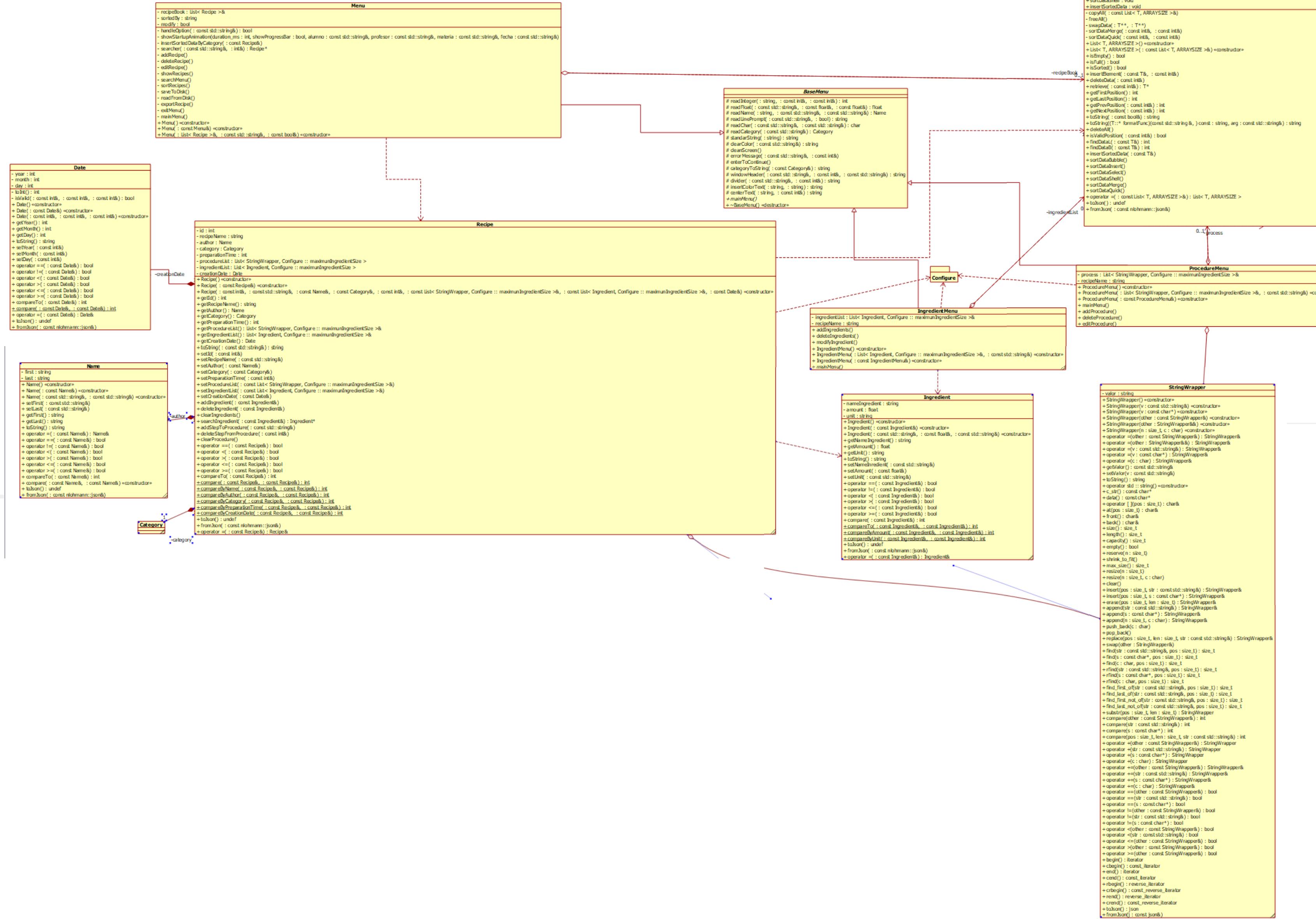
El procedimiento en las recetas se guarda mediante una lista de string’s, pero no string’s cualquiera, ya que si no esto daría error a la hora de invocar un `toString` o un `toJson` por ejemplo, así que se desarrolló un `stringWrapped` que serviría como su reemplazo para agruparlos en esta lista, de esta manera tenemos un control más sencillo en estos.

Considerando el hecho de que íbamos a tener más de 1 menú, tendríamos otro para ingredientes para encapsular la lógica (y en mi caso uno más porque tengo el procedimiento que se guarda como otra lista de `stringWrapped`), se creó una clase virtual pura que servía como base para estos menú’s, su método abstracto era el `mainMenu()`, y sus métodos virtuales eran de repetido uso, como la lectura de datos o la inserción de color, el menú principal, el de ingredientes y el del procedimiento heredan de esto.

Algo que también se cuidó fue el límite de las listas, por defecto, las listas se crean con una capacidad de 1024 espacios, pero considerando que tenemos una lista de recetas, y cada receta en su interior tendría dos listas más, esto podría colapsar en poco tiempo por falta de memoria, así que se crearon 2 static const int en un archivo .hpp configure, sobre el cuál, el programador puede determinar el rango máximo en que las listas almacenan, así, podemos limitar estas listas y evitar este colapso; más adelante, cuando la lista sea dinámica no serán necesarios.

En conjunto, este proyecto permitió aplicar de manera práctica diversos conocimientos sobre estructuras de datos, recursividad y orientación a objetos, consolidando las bases necesarias para continuar con la versión final, en la que se reemplazarán los arreglos dinámicos por listas doblemente y simplemente ligadas, según corresponda.

Diagrama de clases UML





Código Fuente

Carpeta include

basemenu.hpp

```
#ifndef __BASEMENU_H__
#define __BASEMENU_H__


#include <string>

#include "category.hpp"
#include "configure.hpp"
#include "name.hpp"
#include "ownexceptions.hpp"


class BaseMenu {
protected:
    virtual int readInteger(std::string, const int&, const int&);
    virtual float readFloat(const std::string&, const float&, const float&);
    virtual Name readName(std::string,
                          const std::string& = "Ingrese el Nombre:",
                          const std::string& = "Ingrese el Apellido");
    virtual std::string readLinePrompt(const std::string&, bool = false);
    virtual char readChar(const std::string&, const std::string&);
    virtual Category readCategory(const std::string&);
    virtual std::string standarString(std::string);
    virtual std::string clearColor(const std::string&);

    virtual void cleanScreen();
    virtual void errorMessage(const std::string&,
                            const int& =
Configure::predeterminatedSizeWindows);

// Métodos Auxiliares
    virtual void enterToContinue();
    virtual std::string categoryToString(const Category&) const;
    virtual std::string windowHeader(
        const std::string&,
        const int& = Configure::predeterminatedSizeWindows,
        const std::string& = "=") const;
    virtual std::string divider(
        const std::string& = "=",
        const int& = Configure::predeterminatedSizeWindows) const;
```



```
virtual std::string insertColorText(std::string, std::string);
virtual std::string centerText(
    std::string,
    const int& = Configure::predeterminedSizeWindows);

public:
    virtual void mainMenu() = 0;
    virtual ~BaseMenu();
};

#endif // __BASEMENU_H__
```



Category.hpp

```
#ifndef __CATEGORY_H__
#define __CATEGORY_H__

#include <string>

enum Category {
    DESAYUNO = 1,
    COMIDA = 2,
    CENA = 3,
    NAVIDEÑO = 4,
};

#endif // __CATEGORY_H__
```



Configure.hpp

```
#ifndef __CONFIGURE_H__
#define __CONFIGURE_H__


namespace Configure {
    static const int predeterminedSizeWindows =
        60; // Controla el tamaño de los títulos y lo centrado de los textos de
             // forma Global
    // Variables para la Gestión de Memoria Estática a fin de no abarcar
    cantidades
    // excesivas de la misma
    static const int maximunIngredientSize =
        20; // Cantidad Máxima de Ingredientes para Una receta
    static const int maximunProcedureSize =
        20; // Cantidad Máxima de Pasos para una Receta
} // namespace Configure
#endif // __CONFIGURE_H__
```



date.hpp

```
#ifndef __DATE_H__
#define __DATE_H__


#include <chrono>
#include <fstream>
#include <iostream>

#include "../lib/nlohmann/json.hpp"

#include "ownexceptions.hpp"


class Date {
private:
    int year;
    int month;
    int day;

    int toInt() const;
    bool isValid(const int&, const int&, const int&) const;

public:
    Date();
    Date(const Date&);

    Date(const int&, const int&, const int&);

    int getYear() const;
    int getMonth() const;
    int getDay() const;

    std::string toString() const;

    void setYear(const int&);
    void setMonth(const int&);
    void setDay(const int&);

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

    int compareTo(const Date&) const;
```



```
static int compare(const Date&, const Date&);

Date& operator=(const Date&);

nlohmann::json toJson() const;
void fromJson(const nlohmann::json&);

};

#endif // __DATE_H__
```



ingredient.hpp

```
#ifndef __INGREDIENT_H__
#define __INGREDIENT_H__


#include <windows.h>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>

#include "../lib/nlohmann/json.hpp"

#include "ownexceptions.hpp"


class Ingredient {
private:
    std::string nameIngredient;
    float amount;
    std::string unit;

public:
    Ingredient();
    Ingredient(const Ingredient&);
    /// @brief
    /// @param nameIngredient
    /// @param amount
    /// @param unit
    Ingredient(const std::string&, const float&, const std::string&);

    std::string getNameIngredient() const;
    float getAmount() const;
    std::string getUnit() const;

    std::string toString() const;

    void setNameInredient(const std::string&);
    void setAmount(const float&);
    void setUnit(const std::string&);

    bool operator==(const Ingredient&) const;
    bool operator!=(const Ingredient&) const;
    bool operator<(const Ingredient&) const;
```



```
bool operator>(const Ingredient&) const;
bool operator<=(const Ingredient&) const;
bool operator>=(const Ingredient&) const;

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

static int compareByAmount(const Ingredient&, const Ingredient&);
static int compareByUnit(const Ingredient&, const Ingredient&);

nlohmann::json toJson() const;
void fromJson(const nlohmann::json&);

Ingredient& operator=(const Ingredient&);

};

#endif // __INGREDIENT_H__
```



Ingredientmenu.hpp

```
#ifndef __INGREDIENTMENU_H__
#define __INGREDIENTMENU_H__


#include <sstream>
#include <string>

#include "basemenu.hpp"
#include "ingredient.hpp"
#include "list.hpp"

class IngredientMenu : protected BaseMenu {
private:
    List<Ingredient, Configure::maximunIngredientSize>& ingredientList;
    std::string recipeName;

public:
    void addIngredients();
    void deleteIngredients();
    void modifyIngredient();

    IngredientMenu();
    IngredientMenu(List<Ingredient, Configure::maximunIngredientSize>&,
                  const std::string&);
    IngredientMenu(const IngredientMenu&);

    void mainMenu();
};

#endif // __INGREDIENTMENU_H__
```



list.hpp

```
#ifndef __LIST_H__
#define __LIST_H__


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

#include "../lib/nlohmann/json.hpp"

#include "ownexceptions.hpp"

template <class T, int ARRSIZE = 1024>
class List {
private:
    T** data;
    int last;

    void copyAll(const List<T, ARRSIZE>&);
    void freeAll();

    void swapData(T**, T**);

    void sortDataMerge(const int&, const int&);
    void sortDataMerge(const int&, const int&, int(const T&, const T&));

    void sortDataQuick(const int&, const int&);
    void sortDataQuick(const int&, const int&, int(const T&, const T&));

public:
    List<T, ARRSIZE>();
    List<T, ARRSIZE>(const List<T, ARRSIZE>&);
    ~List<T, ARRSIZE>();

    bool isEmpty() const;
    bool isFull() const;
    bool isSorted() const;

    void insertElement(const T&, const int&);
    void deleteData(const int&);
    T* retrieve(const int&);

    // Getter's
```



```
int getFirstPosition() const;
int getLastPosition() const;

int getPrevPosition(const int&) const;
int getNextPosition(const int&) const;

// toString Completo
std::string toString(const bool& = false) const;
// toString por categoría
std::string toString(const T&, int(const T&, const T&)) const;
// toString para diferentes impresiones:
std::string toString(std::string (T::*formatFunc)(const std::string&)
const,
                     const std::string& arg);

void deleteAll();

bool isValidPosition(const int&) const;

int findDataL(const T&);
int findDataB(const T&);

void insertSortedData(const T&);

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

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

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

// Métodos Extras al Modelo
nlohmann::json toJson() const;
void fromJson(const nlohmann::json&);

bool isSorted(int(const T&, const T&)) const;

int findDataL(const T&, int(const T&, const T&));
```



```
int findDataB(const T&, int(const T&, const 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&));
void sortDataMerge(int(const T&, const T&));
void sortDataQuick(int(const T&, const T&));

void insertSortedData(const T&, int(const T&, const T&));
};

template <class T, int ARRSIZE>
List<T, ARRSIZE>::List() : last(-1) {
    this->data = new T*[ARRSIZE];
    for (int i = 0; i < ARRSIZE; i++)
        this->data[i] = nullptr;
}

template <class T, int ARRSIZE>
List<T, ARRSIZE>::~List() {
    freeAll();
    delete[] data;
    data = nullptr;
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::copyAll(const List<T, ARRSIZE>& other) {
    this->last = other.last;

    this->freeAll();
    for (int i = 0; i <= other.last; i++) {
        this->data[i] = new T(*other.data[i]);
    }

    for (int i = other.last + 1; i < ARRSIZE; i++) {
        if (this->data[i])
            delete this->data[i];
        this->data[i] = nullptr;
    }
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::freeAll() {
    for (int i = 0; i <= this->last; i++) {
```



```
        if (this->data[i] != nullptr)
            delete this->data[i];
        this->data[i] = nullptr;
    }

template <class T, int ARYSIZE>
bool List<T, ARYSIZE>::isValidPosition(const int& position) const {
    return !(position > last || position < 0);
}

template <class T, int ARYSIZE>
List<T, ARYSIZE>::List(const List<T, ARYSIZE>& other) {
    this->data = new T*[ARYSIZE];

    for (int i = 0; i < ARYSIZE; i++)
        this->data[i] = nullptr;

    copyAll(other);
}

template <class T, int ARYSIZE>
bool List<T, ARYSIZE>::isEmpty() const {
    return this->last == -1;
}

template <class T, int ARYSIZE>
bool List<T, ARYSIZE>::isFull() const {
    return this->last == (ARYSIZE - 1);
}

template <class T, int ARYSIZE>
bool List<T, ARYSIZE>::isSorted() const {
    for (int i = 0; i < this->last; i++)
        if (*this->data[i] > *this->data[i + 1])
            return false;

    return true;
}

template <class T, int ARYSIZE>
bool List<T, ARYSIZE>::isSorted(int cmp(const T&, const T&)) const {
    for (int i = 0; i < this->last; i++)
        if (cmp(*this->data[i], *this->data[i + 1]) > 0)
            return false;
```



```
        return true;
    }

    // Inserción en el Punto de Interés
    template <class T, int ARYSIZE>
    void List<T, ARYSIZE>::insertElement(const T& newData, const int&
    position) {
        if (isFull())
            throw DataContainersExceptions::MemoryDeficiency(
                "Lista Llena, InsertElement(List)");

        if (!isValidPosition(position) && position != last + 1)
            throw DataContainersExceptions::InvalidPosition(
                "Posicion Invalida, InsertElement(List)");

        for (int i = last; i >= position; i--)
            this->data[i + 1] = this->data[i];

        this->data[position] = new T(newData);
        last++;
    }

    template <class T, int ARYSIZE>
    void List<T, ARYSIZE>::deleteData(const int& position) {
        if (!isValidPosition(position))
            throw DataContainersExceptions::InvalidPosition(
                "Poscion Invalida, delteData(List)");

        // Liberar la Memoria de la posicion indicada:
        delete this->data[position];
        this->data[position] = nullptr;

        // Recorrer la Lista
        for (int i = position; i < last; i++)
            this->data[i] = this->data[i + 1];
        last--;
    }

    template <class T, int ARYSIZE>
    T* List<T, ARYSIZE>::retrieve(const int& position) {
        if (!isValidPosition(position))
            throw DataContainersExceptions::InvalidPosition(
                "Posicion Invalida, retrieve(List)");
        return data[position];
```



```
}

template <class T, int ARYSIZE>
int List<T, ARYSIZE>::getFirstPosition() const {
    return isEmpty() ? -1 : 0;
}

template <class T, int ARYSIZE>
int List<T, ARYSIZE>::getLastPosition() const {
    return this->last;
}

template <class T, int ARYSIZE>
int List<T, ARYSIZE>::getPrevPosition(const int& position) const {
    return (!isValidPosition(position) || position == 0) ? -1 : (position -
1);
}

template <class T, int ARYSIZE>
int List<T, ARYSIZE>::getNextPosition(const int& position) const {
    return (!isValidPosition(position) || position == last) ? -1 : (position +
1);
}

template <class T, int ARYSIZE>
std::string List<T, ARYSIZE>::toString(const bool& numbered) const {
    std::ostringstream oss;
    for (int i = 0; i <= this->last; i++) {
        if (numbered)
            oss << i + 1 << ". ";
        oss << this->data[i]->toString() << std::endl;
    }
    return oss.str();
}

template <class T, int ARYSIZE>
std::string List<T, ARYSIZE>::toString(const T& search,
                                         int cmp(const T&, const T&)) const
{
    std::ostringstream oss;
    for (int i = 0; i <= this->last; i++) {
        if (cmp(search, *this->data[i]) == 0)
            oss << this->data[i]->toString() << std::endl;
    }
}
```



```
        return oss.str();
    }

template <typename T, int ARRSIZE>
std::string List<T, ARRSIZE>::toString(
    std::string (T::*formatFunc)(const std::string&) const,
    const std::string& arg) {
    std::ostringstream oss;
    for (int i = 0; i < this->last; i++) {
        oss << (this->data[i]->*formatFunc)(arg) << "\n";
    }
    return oss.str();
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::deleteAll() {
    this->freeAll();
    this->last = -1;
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::insertSortedData(const T& newData) {
    int i(0);

    while ((i <= this->last) && (newData > *this->data[i]))
        i++;

    this->insertElement(newData, i);
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::insertSortedData(const T& newData,
                                         int cmp(const T&, const T&)) {
    int i(0);

    while ((i <= this->last) && cmp(newData, *this->data[i]) > 0)
        i++;

    this->insertElement(newData, i);
}

template <class T, int ARRSIZE>
int List<T, ARRSIZE>::findDataL(const T& searchedData) {
    for (int i = 0; i <= this->last; i++)
```



```
        if (*this->data[i] == searchData)
            return i;
        return -1;
    }

template <class T, int ARYSIZE>
int List<T, ARYSIZE>::findDataB(const T& searchData) {
    int i(0), j(this->last), middle;

    while (i <= j) {
        middle = (i + j) / 2;
        if (*this->data[middle] == searchData)
            return middle;
        if (searchData < *this->data[middle])
            j = middle - 1;
        else
            i = middle + 1;
    }
    return -1;
}

template <class T, int ARYSIZE>
void List<T, ARYSIZE>::swapData(T** a, T** b) {
    T* aux = *a;
    *a = *b;
    *b = aux;
}

template <class T, int ARYSIZE>
void List<T, ARYSIZE>::sortDataBubble() {
    int i(this->last), j;
    bool flag;

    do {
        flag = false;
        j = 0;

        while (j < i) {
            if (*this->data[j] > *this->data[j + 1]) {
                swapData(&this->data[j], &this->data[j + 1]);
                flag = true;
            }
            j++;
        }
    }
```



```
i--;
} while (flag);
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataInsert() {
    int i(1), j;
    T* aux;

    while (i <= this->last) {
        aux = this->data[i];

        j = i;
        while (j > 0 && *aux < *this->data[j - 1]) {
            this->data[j] = this->data[j - 1];

            j--;
        }

        if (i != j) {
            this->data[j] = aux;
        }

        i++;
    }
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataSelect() {
    int i(0), j, menor;

    while (i <= this->last) {
        menor = i;

        j = i + 1;

        while (j <= this->last) {
            if (*this->data[j] < *this->data[menor]) {
                menor = j;
            }
            j++;
        }

        if (i != menor) {
            this->swapData(&this->data[i], &this->data[menor]);
        }
    }
}
```



```
        }
        i++;
    }
}

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

    while (dif > 0) {
        i = dif;
        while (i <= this->last) {
            j = i;

            while (j >= dif && *this->data[j - dif] > *this->data[j]) {
                this->swapData(&this->data[j - dif], &this->data[j]);
                j -= dif;
            }

            i++;
        }

        dif = series[++pos];
    }
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataMerge() {
    this->sortDataMerge(0, this->last);
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataMerge(int cmp(const T&, const T&)) {
    this->sortDataMerge(0, this->last, cmp);
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataMerge(const int& leftEdge,
                                         const int& rightEdge) {
    // Criterio de Paro
    if (leftEdge >= rightEdge)
        return;
```



```
int m((leftEdge + rightEdge) / 2);

// Divide y Vencerás
this->sortDataMerge(leftEdge, m);
this->sortDataMerge(m + 1, rightEdge);

// Intercalación
static T temp[ARRAYSIZE];

for (int n(leftEdge); n <= rightEdge; n++)
    temp[n] = this->data[n];

int i(leftEdge), j(m + 1), x(leftEdge);

while (i <= m && j <= rightEdge) {
    while (i <= m && *temp[i] <= *temp[j])
        this->data[x++] = temp[i++];

    if (i <= m)
        while (j <= rightEdge && *temp[j] <= *temp[i])
            this->data[x++] = temp[j++];
}

while (i <= m)
    this->data[x++] = temp[i++];
while (j <= m)
    this->data[x++] = temp[j++];
}

template <class T, int ARYSIZE>
List<T, ARYSIZE> List<T, ARYSIZE>::operator=(

    const List<T, ARYSIZE>& other) {
    copyAll(other);
    return *this;
}

template <class T, int ARYSIZE>
void List<T, ARYSIZE>::sortDataQuick() {
    this->sortDataQuick(0, this->last);
}

template <class T, int ARYSIZE>
void List<T, ARYSIZE>::sortDataQuick(int cmp(const T&, const T&)) {
    this->sortDataQuick(0, this->last, cmp);
}
```



```
template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataQuick(const int& leftEdge,
                                         const int& rightEdge) {
    // Criterio de paro
    if (leftEdge >= rightEdge)
        return;

    if (leftEdge == rightEdge + 1) {
        if (*this->data[leftEdge] > *this->data[rightEdge])
            this->swapData(&this->data[leftEdge], &this->data[rightEdge]);
        return;
    }

    // Separación
    int i(leftEdge), j(rightEdge);

    while (i < j) {
        while (i < j && *this->data[i] <= *this->data[rightEdge])
            i++;

        while (i < j && *this->data[j] >= *this->data[rightEdge])
            j--;

        if (i != j)
            this->swapData(&this->data[i], &this->data[j]);
    }

    if (i != rightEdge)
        this->swapData(&this->data[i], &this->data[rightEdge]);

    // Divide y Vencerás
    this->sortDataQuick(leftEdge, i - 1);
    this->sortDataQuick(i + 1, rightEdge);
}

template <class X>
std::ostream& operator<<(std::ostream& os, const List<X>& list) {
    int i = 0;
    while (i <= list.last)
        os << *list.data[i++] << "," << std::endl;

    return os;
}
```



```
template <class X>
std::istream& operator>>(std::istream& is, List<X>& list) {
    X obj;

    try {
        while (is >> obj) {
            if (!list.isFull())
                list.data[++list.last] = new X(obj);
        }
    } catch (const std::invalid_argument& ex) {
    }
    return is;
}

// Extras al Modelo de la Lista:

template <class T, int ARRSIZE>
nlohmann::json List<T, ARRSIZE>::toJson() const {
    nlohmann::json j;
    j["data"] = nlohmann::json::array();
    for (int i = 0; i <= this->last; i++) {
        j["data"].push_back(this->data[i]->toJson());
    }
    return j;
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::fromJson(const nlohmann::json& js) {
    this->deleteAll();
    T* obj(new T);
    for (const auto& data : js) {
        obj->fromJson(data);
        this->insertElement(*obj, this->last + 1);
    }
}

template <class T, int ARRSIZE>
int List<T, ARRSIZE>::findDataL(const T& searchData,
                                  int cmp(const T&, const T&)) {
    for (int i = 0; i <= this->last; i++)
        if (cmp(searchData, *this->data[i]) == 0)
            return i;

    return -1;
}
```



```
template <class T, int ARRSIZE>
int List<T, ARRSIZE>::findDataB(const T& searchedData,
                                  int cmp(const T&, const T&)) {
    int i(0), j(this->last), middle;

    while (i <= j) {
        middle = (i + j) / 2;

        if (cmp(searchedData, *this->data[middle]) == 0)
            return middle;
        if (cmp(searchedData, *this->data[middle]) < 0)
            j = middle - 1;
        else
            i = middle + 1;
    }

    return -1;
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataBubble(int cmp(const T&, const T&)) {
    int i(this->last), j;
    bool flag;

    do {
        flag = false;
        j = 0;

        while (j < i) {
            if (cmp(*this->data[j], *this->data[j + 1]) > 0) {
                swapData(&this->data[j], &this->data[j + 1]);
                flag = true;
            }
            j++;
        }

        i--;
    } while (flag);
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataInsert(int cmp(const T&, const T&)) {
    int i(1), j;
    T* aux;
```



```
while (i <= this->last) {  
    aux = this->data[i];  
  
    j = i;  
    while (j > 0 && cmp(*aux, *this->data[j - 1]) < 0) {  
        this->data[j] = this->data[j - 1];  
  
        j--;  
    }  
  
    if (i != j) {  
        this->data[j] = aux;  
    }  
  
    i++;  
}  
}  
  
template <class T, int ARRSIZE>  
void List<T, ARRSIZE>::sortDataSelect(int cmp(const T&, const T&)) {  
    int i(0), j, menor;  
  
    while (i <= this->last) {  
        menor = i;  
  
        j = i + 1;  
  
        while (j <= this->last) {  
            if (cmp(*this->data[j], *this->data[menor]) < 0) {  
                menor = j;  
            }  
            j++;  
        }  
  
        if (i != menor) {  
            this->swapData(&this->data[i], &this->data[menor]);  
        }  
        i++;  
    }  
}  
  
template <class T, int ARRSIZE>  
void List<T, ARRSIZE>::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 pos(0), dif(series[pos]), i, j;  
  
while (dif > 0) {  
    i = dif;  
    while (i <= this->last) {  
        j = i;  
  
        while (j >= dif && cmp(*this->data[j - dif], *this->data[j]) > 0) {  
            this->swapData(&this->data[j - dif], &this->data[j]);  
            j -= dif;  
        }  
  
        i++;  
    }  
  
    dif = series[++pos];  
}  
}  
  
template <class T, int ARRSIZE>  
void List<T, ARRSIZE>::sortDataMerge(const int& leftEdge,  
                                      const int& rightEdge,  
                                      int cmp(const T&, const T&)) {  
    // Criterio de Paro  
    if (leftEdge >= rightEdge)  
        return;  
  
    int m((leftEdge + rightEdge) / 2);  
  
    // Divide y Vencerás  
    this->sortDataMerge(leftEdge, m, cmp);  
    this->sortDataMerge(m + 1, rightEdge, cmp);  
  
    // Intercalación  
    static T temp[ARRSIZE];  
  
    for (int n(leftEdge); n <= rightEdge; n++)  
        temp[n] = this->data[n];  
  
    int i(leftEdge), j(m + 1), x(leftEdge);  
  
    while (i <= m && j <= rightEdge) {  
        while (i <= m && cmp(*temp[i], *temp[j]) <= 0)  
            this->data[x++] = temp[i++];
```



```
if (i <= m)
    while (j <= rightEdge && cmp(*temp[j], *temp[i]) <= 0)
        this->data[x++] = temp[j++];
}

while (i <= m)
    this->data[x++] = temp[i++];
while (j <= m)
    this->data[x++] = temp[j++];
}

template <class T, int ARRSIZE>
void List<T, ARRSIZE>::sortDataQuick(const int& leftEdge,
                                         const int& rightEdge,
                                         int cmp(const T&, const T&)) {
// Criterio de paro
if (leftEdge >= rightEdge)
    return;

if (leftEdge == rightEdge + 1) {
    if (cmp(*this->data[leftEdge], *this->data[rightEdge]) > 0)
        this->swapData(&this->data[leftEdge], &this->data[rightEdge]);
    return;
}

// Separación
int i(leftEdge), j(rightEdge);

while (i < j) {
    while (i < j && cmp(*this->data[i], *this->data[rightEdge]) <= 0)
        i++;

    while (i < j && cmp(*this->data[j], *this->data[rightEdge]) >= 0)
        j--;

    if (i != j)
        this->swapData(&this->data[i], &this->data[j]);
}

if (i != rightEdge)
    this->swapData(&this->data[i], &this->data[rightEdge]);

// Divide y Vencerás
this->sortDataQuick(leftEdge, i - 1, cmp);
```



```
this->sortDataQuick(i + 1, rightEdge, cmp);  
}  
  
#endif // __LIST_H__
```



menu.hpp

```
#ifndef __MENU_H__
#define __MENU_H__


#include <windows.h>
#include <chrono>
#include <fstream>
#include <iomanip>
#include <sstream>
#include <string>
#include <thread>

#include "basemenu.hpp"
#include "configure.hpp"
#include "ingredientMenu.hpp"
#include "list.hpp"
#include "ownexceptions.hpp"
#include "proceduremenu.hpp"
#include "recipe.hpp"


class Menu : protected BaseMenu {
private:
    List<Recipe>& recipeBook;
    std::string sortedBy = "id";
    bool modify = false;

    bool handleOption(const std::string&);

    void showStartupAnimation(
        int duration_ms = 2000,
        bool showProgressBar = true,
        const std::string& alumno = "Omar Mariscal",
        const std::string& profesor = "Alfredo Gutiérrez",
        const std::string& materia = " Estructuras de Datos",
        const std::string& fecha = Date().toString()) const;

    void insertSortedDataByCategory(const Recipe&);

    // Pantallas
    // Pantallas Intermedias
    Recipe* searcher(const std::string&, int&);

    // Pantallas Principales
    void addRecipe();
    void deleteRecipe();
```



```
void editRecipe();
void showRecipes();
void searchMenu();
void sortRecipes();
void saveToDisk();
void readFromDisk();
void exportRecipe();
void exitMenu();
void MainMenu() override;

public:
// Constructores
Menu();
Menu(const Menu&);
Menu(List<Recipe>&, const std::string&, const bool&);
};

#endif // __MENU_H__
```



name.hpp

```
#ifndef __NAME_H__
#define __NAME_H__

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

#include "../lib/nlohmann/json.hpp"

#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&);

nlohmann::json toJson() const;
void fromJson(const nlohmann::json&);

};

#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 categoría de inserción es inválida")
: runtime_error(msg) {}
};

} // namespace MenuExceptions

#endif // __OWNEXCEPTIONS_H__
```



proceduremenu.hpp

```
#ifndef __PROCEDUREMENU_H__
#define __PROCEDUREMENU_H__


#include <sstream>
#include <string>

#include "basemenu.hpp"
#include "configure.hpp"
#include "list.hpp"
#include "ownexceptions.hpp"
#include "stringwrapped.hpp"


class ProcedureMenu : protected BaseMenu {
private:
    List<StringWrapper, Configure::maximumIngredientSize>& process;
    std::string recipeName;

public:
    ProcedureMenu();
    ProcedureMenu(List<StringWrapper, Configure::maximumIngredientSize> &,
                 const std::string& );
    ProcedureMenu(const ProcedureMenu&);

    void mainMenu();
    void addProcedure();
    void deleteProcedure();
    void editProcedure();
};

#endif // __PROCEDUREMENU_H__
```



recipe.hpp

```
#ifndef __RECIPE_H__
#define __RECIPE_H__


#include <windows.h>
#include <iostream>
#include <sstream>
#include <string>

#include "../lib/nlohmann/json.hpp"

#include "category.hpp"
#include "configure.hpp"
#include "date.hpp"
#include "ingredient.hpp"
#include "list.hpp"
#include "name.hpp"
#include "stringwrapped.hpp"


class Recipe {
private:
    int id;
    std::string recipeName;
    Name author;
    Category category;
    int preparationTime;
    List<StringWrapper, Configure::maximunIngredientSize> procedureList;
    List<Ingredient, Configure::maximunIngredientSize> ingredientList;
    Date creationDate;

public:
    Recipe();
    Recipe(const Recipe&);
    Recipe(const int&,
           const std::string&,
           const Name&,
           const Category&,
           const int&,
           const List<StringWrapper, Configure::maximunIngredientSize>&,
           const List<Ingredient, Configure::maximunIngredientSize>&,
           const Date&);

    int getId() const;
    std::string getRecipeName() const;
```



```
Name getAuthor() const;
Category getCategory() const;
int getPreparationTime() const;
List<StringWrapper, Configure::maximunIngredientSize>& getProcedureList();
List<Ingredient, Configure::maximunIngredientSize>& getIngredientList();
Date getCreationDate() const;

std::string toString(const std::string& = "full") const;

// Setter's
void setId(const int&);
void setRecipeName(const std::string&);
void setAuthor(const Name&);
void setCategory(const Category&);
void setPreparationTime(const int&);
void setProcedureList(
    const List<StringWrapper, Configure::maximunIngredientSize>&);

void setIngredientList(
    const List<Ingredient, Configure::maximunIngredientSize>&);

void setCreationDate(const Date&);

// Algoritmicos
// Control de Ingredientes
void addIngredient(const Ingredient&);
void deleteIngredient(const Ingredient&);
void clearIngredients();
Ingredient* searchIngredient(const Ingredient&);

// Control de Procedimiento
void addStepToProcedure(const std::string&);
void deleteStepFromProcedure(const int&);
void clearProcedure();

bool operator==(const Recipe&) const;
bool operator<(const Recipe&) const;
bool operator>(const Recipe&) const;
bool operator<=(const Recipe&) const;
bool operator>=(const Recipe&) const;

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

static int compareByName(const Recipe&, const Recipe&);
static int compareByAuthor(const Recipe&, const Recipe&);
static int compareByCategory(const Recipe&, const Recipe&);
```



```
static int compareByPreparationTime(const Recipe&, const Recipe&);  
static int compareByCreationDate(const Recipe&, const Recipe&);  
  
nlohmann::json toJson() const;  
void fromJson(const nlohmann::json&);  
  
Recipe& operator=(const Recipe&);  
};  
  
#endif // __RECIPE_H__
```



```
stringwrapped.hpp
// StringWrapper.hpp

#ifndef __STRINGWRAPPER_H__
#define __STRINGWRAPPER_H__

#include <string>
#include <iostream>

#include <../lib/nlohmann/json.hpp>

using json = nlohmann::json;

class StringWrapper {
private:
    std::string valor;

public:
    // ===== CONSTRUCTORES =====
    StringWrapper();
    StringWrapper(const std::string& v);
    StringWrapper(const char* v);
    StringWrapper(const StringWrapper& other);
    StringWrapper(StringWrapper&& other) noexcept;
    StringWrapper(size_t n, char c);

    // ===== OPERADORES DE ASIGNACIÓN =====
    StringWrapper& operator=(const StringWrapper& other);
    StringWrapper& operator=(StringWrapper&& other) noexcept;
    StringWrapper& operator=(const std::string& v);
    StringWrapper& operator=(const char* v);
    StringWrapper& operator=(char c);

    // ===== GETTERS Y SETTERS (inline) =====
    const std::string& getValor() const { return valor; }
    std::string& getValor() { return valor; }
    void setValor(const std::string& v) { valor = v; }
    std::string toString() const { return valor; }

    // ===== CONVERSIÓN (inline) =====
    operator std::string() const { return valor; }
    const char* c_str() const { return valor.c_str(); }
    const char* data() const { return valor.data(); }
```



```
// ===== ACCESO A ELEMENTOS (inline)
=====
char& operator[](size_t pos) { return valor[pos]; }
const char& operator[](size_t pos) const { return valor[pos]; }
char& at(size_t pos) { return valor.at(pos); }
const char& at(size_t pos) const { return valor.at(pos); }
char& front() { return valor.front(); }
const char& front() const { return valor.front(); }
char& back() { return valor.back(); }
const char& back() const { return valor.back(); }

// ===== CAPACIDAD (inline) =====
size_t size() const { return valor.size(); }
size_t length() const { return valor.length(); }
size_t capacity() const { return valor.capacity(); }
bool empty() const { return valor.empty(); }
void reserve(size_t n) { valor.reserve(n); }
void shrink_to_fit() { valor.shrink_to_fit(); }
size_t max_size() const { return valor.max_size(); }
void resize(size_t n) { valor.resize(n); }
void resize(size_t n, char c) { valor.resize(n, c); }

// ===== MODIFICADORES =====
void clear();
StringWrapper& insert(size_t pos, const std::string& str);
StringWrapper& insert(size_t pos, const char* s);
StringWrapper& erase(size_t pos = 0, size_t len = std::string::npos);
StringWrapper& append(const std::string& str);
StringWrapper& append(const char* s);
StringWrapper& append(size_t n, char c);
void push_back(char c);
void pop_back();
StringWrapper& replace(size_t pos, size_t len, const std::string& str);
void swap(StringWrapper& other);

// ===== OPERACIONES DE STRING (inline)
=====
size_t find(const std::string& str, size_t pos = 0) const { return
valor.find(str, pos); }
size_t find(const char* s, size_t pos = 0) const { return valor.find(s,
pos); }
size_t find(char c, size_t pos = 0) const { return valor.find(c, pos); }
size_t rfind(const std::string& str, size_t pos = std::string::npos)
const { return valor.rfind(str, pos); }
```



```
size_t rfind(const char* s, size_t pos = std::string::npos) const {
    return valor.rfind(s, pos); }
size_t rfind(char c, size_t pos = std::string::npos) const { return
    valor.rfind(c, pos); }
size_t find_first_of(const std::string& str, size_t pos = 0) const {
    return valor.find_first_of(str, pos); }
size_t find_last_of(const std::string& str, size_t pos =
    std::string::npos) const { return valor.find_last_of(str, pos); }
size_t find_first_not_of(const std::string& str, size_t pos = 0) const {
    return valor.find_first_not_of(str, pos); }
size_t find_last_not_of(const std::string& str, size_t pos =
    std::string::npos) const { return valor.find_last_not_of(str, pos); }
StringWrapper substr(size_t pos = 0, size_t len = std::string::npos)
const { return StringWrapper(valor.substr(pos, len)); }
int compare(const StringWrapper& other) const { return
    valor.compare(other.valor); }
int compare(const std::string& str) const { return valor.compare(str); }
int compare(const char* s) const { return valor.compare(s); }
int compare(size_t pos, size_t len, const std::string& str) const {
    return valor.compare(pos, len, str); }

// ===== OPERADORES (la mayoría son inline)
=====
StringWrapper operator+(const StringWrapper& other) const { return
    StringWrapper(valor + other.valor); }
StringWrapper operator+(const std::string& str) const { return
    StringWrapper(valor + str); }
StringWrapper operator+(const char* s) const { return
    StringWrapper(valor + s); }
StringWrapper operator+(char c) const { return StringWrapper(valor + c);
}
StringWrapper& operator+=(const StringWrapper& other);
StringWrapper& operator+=(const std::string& str);
StringWrapper& operator+=(const char* s);
StringWrapper& operator+=(char c);

bool operator==(const StringWrapper& other) const { return valor ==
    other.valor; }
bool operator==(const std::string& str) const { return valor == str; }
bool operator==(const char* s) const { return valor == s; }
bool operator!=(const StringWrapper& other) const { return valor !=
    other.valor; }
bool operator!=(const std::string& str) const { return valor != str; }
bool operator!=(const char* s) const { return valor != s; }
```



```
    bool operator<(const StringWrapper& other) const { return valor <
other.valor; }
    bool operator<(const std::string& str) const { return valor < str; }
    bool operator<=(const StringWrapper& other) const { return valor <=
other.valor; }
    bool operator>(const StringWrapper& other) const { return valor >
other.valor; }
    bool operator>=(const StringWrapper& other) const { return valor >=
other.valor; }

// ===== ITERADORES (inline) =====
using iterator = std::string::iterator;
using const_iterator = std::string::const_iterator;
using reverse_iterator = std::string::reverse_iterator;
using const_reverse_iterator = std::string::const_reverse_iterator;

iterator begin() { return valor.begin(); }
const_iterator begin() const { return valor.begin(); }
const_iterator cbegin() const { return valor.cbegin(); }
iterator end() { return valor.end(); }
const_iterator end() const { return valor.end(); }
const_iterator cend() const { return valor.cend(); }
reverse_iterator rbegin() { return valor.rbegin(); }
const_reverse_iterator rbegin() const { return valor.rbegin(); }
const_reverse_iterator crbegin() const { return valor.crbegin(); }
reverse_iterator rend() { return valor.rend(); }
const_reverse_iterator rend() const { return valor.rend(); }
const_reverse_iterator crend() const { return valor.crend(); }

// ===== SERIALIZACIÓN JSON =====
json toJson() const;
void fromJson(const json& j);

// ===== OPERADORES DE STREAM (amigos)
=====
friend std::ostream& operator<<(std::ostream& os, const StringWrapper&
sw);
friend std::istream& operator>>(std::istream& is, StringWrapper& sw);
friend std::istream& getline(std::istream& is, StringWrapper& sw);
friend std::istream& getline(std::istream& is, StringWrapper& sw, char
delim);
};

// ===== OPERADORES GLOBALES (deben ser inline en el header)
=====
```



```
inline StringWrapper operator+(const std::string& str, const StringWrapper&
sw) { return StringWrapper(str + sw.getValor()); }
inline StringWrapper operator+(const char* s, const StringWrapper& sw) {
return StringWrapper(std::string(s) + sw.getValor()); }
inline StringWrapper operator+(char c, const StringWrapper& sw) { return
StringWrapper(c + sw.getValor()); }

inline bool operator==(const std::string& str, const StringWrapper& sw) {
return str == sw.getValor(); }
inline bool operator==(const char* s, const StringWrapper& sw) { return
std::string(s) == sw.getValor(); }
inline bool operator!=(const std::string& str, const StringWrapper& sw) {
return str != sw.getValor(); }
inline bool operator!=(const char* s, const StringWrapper& sw) { return
std::string(s) != sw.getValor(); }

#endif // __STRINGWRAPPER_H__
```



Carpeta lib

Json.hpp

Lleva una biblioteca adicional para el manejo de archivo .json

Dicho recurso se encuentra en el siguiente link de GitHub:

<https://github.com/nlohmann/json>

Se adaptaron las clases a métodos toJson y fromJson de objetos que se manejan internamente en esta librería de only header o encabezado único.



Carpeta src

basemenu.cpp

```
#include "basemenu.hpp"

using namespace std;

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

            if (aux == "/r")
                throw InputExceptions::OperationCanceledException(
                    "Cancelado por usuario");

            result = stoi(aux);

            if (result > upperLimit || result < lowerLimit)
                throw InputExceptions::InvalidOption("Número Fuera de Rango");
            break;
        } catch (const InputExceptions::InvalidOption& ex) {
            this->errorMessage("Número Fuera de Rango\nVuelva a Intentarlo");
        } catch (const invalid_argument& msg) {
            this->errorMessage(
                "Entrada No Numérica\nAsegúrese de Ingresar Un Número Válido");
        }
    }
    return result;
}

float BaseMenu::readFloat(const std::string& oss,
                           const float& lowerLimit,
                           const float& upperLimit) {
    std::string aux("");
    float result;
    while (true) {
        try {
```



```
        system("CLS");
        std::cout << oss;
        std::getline(std::cin, aux);

        if (aux == "/r")
            throw InputExceptions::InvalidOption("Cancelado por usuario");

        result = std::stof(aux);

        if (result > upperLimit || result < lowerLimit)
            throw InputExceptions::InvalidOption("Número Fuera de Rango");
        break;
    } catch (const std::invalid_argument& ex) {
        this->errorMessage("Entrada Inválida\nInténtelo Nuevamente");
    } catch (const std::out_of_range& ex) {
        this->errorMessage("Entrada Fuera de Rango\nInténtelo Nuevamente");
    } catch (const InputExceptions::InvalidOption& msg) {
        system("CLS");
        std::cout << msg.what() << std::endl;
    }
}

return result;
}

Name BaseMenu::readName(string prompt,
                        const string& petition,
                        const string& petition2) {
    Name result;
    prompt += petition;
    result.setFirst(readLinePrompt(prompt));
    prompt += result.getFirst() + "\n";
    prompt += petition2;
    result.setLast(readLinePrompt(prompt));

    return result;
}

string BaseMenu::readLinePrompt(const string& prompt, bool allowEmpty) {
    string result;
    while (true) {
        system("CLS");
        cout << prompt;
        getline(cin, result);
        if (result == "/r")
```



```
        throw InputExceptions::OperationCanceledException();
    if (!allowEmpty && result.empty()) {
        system("CLS");
        this->errorMessage("No Puede Estar Vacío\nIntente Nuevamente.");
        continue;
    }
    return result;
}

char BaseMenu::readChar(const std::string& prompt, const std::string&
options) {
    while (true) {
        this->cleanScreen();
        string input;
        cout << prompt;
        getline(cin, input);

        if (!input.empty()) {
            char result = toupper(input[0]);
            if (options.find(result) != string::npos)
                return result;
        }

        this->cleanScreen();
        this->errorMessage(
            "La opcion ingresada es invalida.\nIntentelo nuevamente.");
        cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }
}

Category BaseMenu::readCategory(const std::string& prompt) {
    Category result;
    int op;
    op = this->readInteger(prompt, 1, 4);
    switch (op) {
        case 1:
            result = DESAYUNO;
            break;
        case 2:
            result = COMIDA;
            break;
        case 3:
            result = CENA;
            break;
    }
}
```



```
        case 4:
            result = NAVIDEÑO;
            break;
    }

    return result;
}

std::string BaseMenu::standarString(std::string text) {
    // Eliminar espacios al final
    size_t end = text.size();
    while (end > 0 && (text[end - 1] == ' ' || text[end - 1] == '\t')) {
        --end;
    }
    text.resize(end);

    // 2. Convertir a mayúsculas (solo letras ASCII)
    for (size_t i = 0; i < text.size(); ++i) {
        if (text[i] >= 'a' && text[i] <= 'z') {
            text[i] = text[i] - ('a' - 'A');
        }
    }

    return text;
}

std::string BaseMenu::clearColor(const std::string& s) {
    std::string out;
    out.reserve(s.size());
    for (size_t i = 0; i < s.size(); ++i) {
        unsigned char c = static_cast<unsigned char>(s[i]);
        if (c == 0x1B) { // ESC
            // si sigue '[', consumimos hasta una letra (final de secuencia CSI)
            if (i + 1 < s.size() && s[i + 1] == '[') {
                i += 2; // saltamos ESC and '['
                // consumir hasta encontrar una letra A-Z o a-z
                while (i < s.size()) {
                    unsigned char cc = static_cast<unsigned char>(s[i]);
                    if ((cc >= 'A' && cc <= 'Z') || (cc >= 'a' && cc <= 'z'))
                        break;
                    ++i;
                }
                // al salir, i apunta a la letra final; el bucle for incrementará i,
                // por eso dejamos que continúe normalmente (se descarta la letra
final
```



```
// también)
continue;
} else {
// ESC sin '[', saltar el ESC solo
continue;
}
out.push_back(c);
}

return out;
}

void BaseMenu::cleanScreen() {
system("CLS");
}

void BaseMenu::errorMessage(const std::string& prompt, const int&
windowWidth) {
this->cleanScreen();
auto printBorder = [&]() {
std::cout << "+";
for (int i = 0; i < windowHeight; ++i)
std::cout << "-";
std::cout << "+\n";
};

auto printCenteredLine = [&](const std::string& prompt = "") {
int broad = static_cast<int>(prompt.size());
int totalSpaces = windowHeight - broad;
int leftSpaces = totalSpaces / 2;
int rightSpaces = totalSpaces - leftSpaces;

std::cout << "|";
for (int i = 0; i < leftSpaces; ++i)
std::cout << " ";
std::cout << prompt;
for (int i = 0; i < rightSpaces; ++i)
std::cout << " ";
std::cout << "|\n";
};

// --- Imprimir recuadro ---
printBorder();
printCenteredLine("[ERROR]");
```



```
stringstream ss(prompt);
string line;

while (getline(ss, line, '\n'))
    printCenteredLine(line);

printBorder();
system("PAUSE");
}

void BaseMenu::enterToContinue() {
    cout << "+-----+" << endl;
    cout << " | Presiones [ENTER] | " << endl;
    cout << " | Para Continuar | " << endl;
    cout << "+-----+" << endl;
    cin.ignore();
    cin.get();
}

std::string BaseMenu::categoryToString(const Category& category) const {
    switch (category) {
        case DESAYUNO:
            return "desayuno";
        case COMIDA:
            return "comida";
        case CENA:
            return "cena";
        case NAVIDEÑO:
            return "navideño";
    }
    return "";
}

string BaseMenu::windowHeader(const string& prompt,
                             const int& widthBorder,
                             const string& c) const {
    ostringstream oss;
    int spaces((widthBorder - prompt.size()) / 2);

    oss << this->divider(c, widthBorder);
    oss << setw(spaces) << "" << prompt << endl;

    oss << this->divider(c, widthBorder);

    return oss.str();
```



```
}

std::string BaseMenu::divider(const std::string& character,
                             const int& size) const {
    std::string line;
    for (int i = 0; i < size; ++i) {
        line += character;
    }
    line += '\n';
    return line;
}

string BaseMenu::insertColorText(string prompt, std::string color) {
    // Mapa o Diccionario para los colores
    std::unordered_map<string, string> colorMap = {
        {"red", "\033[31m"}, {"green", "\033[32m"}, {"yellow", "\033[33m"},
        {"blue", "\033[34m"}, {"cyan", "\033[36m"}, {"magenta", "\033[35m"},
    };

    auto it = colorMap.find(color);
    if (it != colorMap.end()) {
        prompt.insert(0, it->second);
        prompt += "\033[37m";
    }

    return prompt;
}

string BaseMenu::centerText(string prompt, const int& windowHeight) {
    if (prompt.empty())
        return "";

    string result;
    int broad = prompt.size();
    int totalSpaces = windowHeight - broad;
    int leftSpaces = totalSpaces / 2;
    int rightSpaces = totalSpaces - leftSpaces;

    for (int i = 0; i < leftSpaces; ++i)
        result += " ";
    result += prompt;
    for (int i = 0; i < rightSpaces; ++i)
        result += " ";
    result += '\n';
    return result;
}
```



BaseMenu::~BaseMenu() {}



date.cpp

```
#include "date.hpp"

using namespace std;

Date::Date() {
    auto now = chrono::system_clock::now();
    time_t t = std::chrono::system_clock::to_time_t(now);
    std::tm local_tm = *std::localtime(&t);

    this->day = local_tm.tm_mday;
    this->month = local_tm.tm_mon + 1;
    this->year = local_tm.tm_year + 1900;
} // Inicializar al día de hoy

Date::Date(const Date& other)
: year(other.year), month(other.month), day(other.day) {}

Date::Date(const int& y, const int& m, const int& d)
: year(y), month(m), day(d) {
    if (!this->isValid(y, m, d))
        throw DateExceptions::InvalidDate();
}

int Date::toInt() const {
    return this->year * 10000 + this->month * 100 + this->day;
}

bool Date::isValid(const int& year, const int& month, const int& day) const
{
    int monthDay[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 31};

    if (year == 0 || month < 1 || month > 12)
        return false;

    if (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0))
        monthDay[2]++;
}

if (day < 1 || day > monthDay[month])
    return false;

return true;
}
```



```
int Date::getYear() const {
    return year;
}

int Date::getMonth() const {
    return month;
}

int Date::getDay() const {
    return day;
}

string Date::toString() const {
    return to_string(this->day) + "/" + to_string(this->month) + "/" +
        to_string(this->year);
}

void Date::setYear(const int& y) {
    if (!this->isValid(y, this->month, this->day))
        throw DateExceptions::InvalidDate();
    this->year = y;
}

void Date::setMonth(const int& m) {
    if (!this->isValid(this->year, m, this->day))
        throw DateExceptions::InvalidDate();
    this->month = m;
}

void Date::setDay(const int& d) {
    if (!this->isValid(this->year, this->month, d))
        throw DateExceptions::InvalidDate();
    this->day = d;
}

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

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

bool Date::operator<(const Date& other) const {
    return this->toInt() < other.toInt();
```



```
        }

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

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

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

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

    int Date::compare(const Date& a, const Date& b) {
        return a.toInt() - b.toInt();
    }

    Date& Date::operator=(const Date& other) {
        this->year = other.year;
        this->month = other.month;
        this->day = other.day;

        return *this;
    }

    nlohmann::json Date::toJson() const {
        nlohmann::json js{
            {"year", this->year},
            {"month", this->month},
            {"day", this->day},
        };

        return js;
    }

    void Date::fromJson(const nlohmann::json& js) {
        this->year = js.at("year").get<int>();
        this->month = js.at("month").get<int>();
        this->day = js.at("day").get<int>();
    }
```



ingredient.cpp

```
#include "ingredient.hpp"

Ingredient::Ingredient()
    : nameIngredient("default"), amount(0.0), unit("default") {}

Ingredient::Ingredient(const Ingredient& other)
    : nameIngredient(other.nameIngredient),
      amount(other.amount),
      unit(other.unit) {}

Ingredient::Ingredient(const std::string& n,
                      const float& a,
                      const std::string& u)
    : nameIngredient(n), amount(a), unit(u) {}

std::string Ingredient::getNameIngredient() const {
    return this->nameIngredient;
}

float Ingredient::getAmount() const {
    return this->amount;
}

std::string Ingredient::getUnit() const {
    return this->unit;
}

std::string Ingredient::toString() const {
    std::ostringstream oss;
    oss << "\033[31m-";
    oss << "\033[0m";
    oss << this->nameIngredient;
    oss << std::setfill('.');
    oss << std::setw(25 - this->nameIngredient.size()) << "";
    oss << std::setfill(' ');
    oss << this->amount << " " << this->unit;
    return oss.str();
}

void Ingredient::setNameIngredient(const std::string& nameIngredient) {
    if (nameIngredient.empty())
        throw InputExceptions::EmptyString()
```



```
        "El nombre del ingrediente no puede estar vacio.");
    this->nameIngredient = nameIngredient;
}

void Ingredient::setAmount(const float& amount) {
    if (amount < 0)
        throw InputExceptions::NumberNotPositive(
            "La cantidad de ingrediente debe ser positiva");
    this->amount = amount;
}

void Ingredient::setUnit(const std::string& unit) {
    if (unit.empty())
        throw InputExceptions::EmptyString(
            "La unidad del ingrediente no puede estar vacia");
    this->unit = unit;
}

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

bool Ingredient::operator!=(const Ingredient& other) const {
    return !(this->nameIngredient == other.nameIngredient);
}

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

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

bool Ingredient::operator<=(const Ingredient& other) const {
    return this->nameIngredient <= other.nameIngredient;
}

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

int Ingredient::compare(const Ingredient& other) const {
    return (this->nameIngredient.compare(other.nameIngredient));
}
```



```
int Ingredient::compareTo(const Ingredient& ingredientA,
                           const Ingredient& ingredientB) {
    return ingredientA.nameIngredient.compare(ingredientB.nameIngredient);
}

int Ingredient::compareByAmount(const Ingredient& ingredientA,
                                 const Ingredient& ingredientB) {
    return ingredientA.amount - ingredientB.amount;
}

int Ingredient::compareByUnit(const Ingredient& ingredientA,
                               const Ingredient& ingredientB) {
    return ingredientA.unit.compare(ingredientB.unit);
}

nlohmann::json Ingredient::toJson() const {
    nlohmann::json js{
        {"name ingredient", this->nameIngredient},
        {"amount", this->amount},
        {"unit", this->unit},
    };
    return js;
}

void Ingredient::fromJson(const nlohmann::json& js) {
    this->nameIngredient = js.at("name ingredient").get<std::string>();
    this->amount = js.at("amount").get<float>();
    this->unit = js.at("unit").get<std::string>();
}

Ingredient& Ingredient::operator=(const Ingredient& other) {
    this->nameIngredient = other.nameIngredient;
    this->amount = other.amount;
    this->unit = other.unit;

    return *this;
}
```



ingredientmenu.cpp

```
#include "ingredientMenu.hpp"

using namespace std;

void IngredientMenu::addIngredients() {
    ostringstream oss;
    float dataFloat;
    char op;
    string dataString;
    Ingredient newIngredient;
    try {
        do {
            oss.str("");
            oss.clear();

            oss << this->windowHeader(
                "RECETARIO DIGITAL - AÑADIENDO INGREDIENTES");
            if (!this->ingredientList.isEmpty()) {
                oss << this->centerText(
                    " " +
                    this->insertColorText(
                        "Lista de Ingredientes Actual para " + recipeName,
                        "magenta"));
                oss << this->ingredientList.toString() << endl;
                oss << this->divider();
            }

            oss << this->insertColorText("      Ingrese el Nombre del Ingrediente:
",
                                         "cyan");

            while (true) {
                dataString = this->readLinePrompt(oss.str());
                newIngredient.setNameInredient(dataString);

                if (this->ingredientList.findDataL(newIngredient) != -1)
                    this->errorMessage(
                        "El Ingrediente Ya Esta en la Lista\nIntente Agregar Uno
Nuevo");
                else
                    break;
            }
        }
    }
```



```
oss << dataString << endl;

oss << this->insertColorText("    Ingrese la Cantidad que se Utiliza:
",
                                "cyan");
dataFloat = this->readFloat(oss.str(), 0.0, 9999);
oss << dataFloat << endl;
newIngredient.setAmount(dataFloat);

oss << this->insertColorText(
    "    Ingrese la Unidad de Medidad de la Cantidad: ", "cyan");
dataString = readLinePrompt(oss.str());
oss << dataString << endl;
newIngredient.setUnit(dataString);

this->ingredientList.insertSortedData(newIngredient);

oss << this->divider("-");
oss << "☒ ";
oss << this->insertColorText("Ingrediente Añadido con Exito!",
"green")
    << endl;
oss << "   ¿Desea Añadir Más Ingredientes? (S/N): ";

op = this->readChar(oss.str(), "S,N");

} while (op != 'N');

} catch (const InputExceptions::OperationCanceledException& ex) {
    this->errorMessage("Agregado de Ingrediente Cancelado.\nAvanzando...");
    return;
}
}

void IngredientMenu::deleteIngredients() {
if (this->ingredientList.isEmpty()) {
    this->errorMessage(
        "La Lista de Ingredientes está Vacía.\nNo Hay que Eliminar");
    return;
}

ostringstream oss;
char op;
int dataInt;
string dataString;
```



```
Ingredient searchedIngredient;

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

    oss << this->windowHeader("RECETARIO DIGITAL - ELIMINAR
INGREDIENTES");
    if (!this->ingredientList.isEmpty()) {
        oss << this->centerText(
            "👉" +
            this->insertColorText(
                "Lista de Ingredientes Actual para " + recipeName,
                "magenta"));
        oss << this->ingredientList.toString() << endl;
        oss << this->divider();
    }
    oss << this->centerText("Ingrese 'fin' para regresar");
    oss << this->insertColorText(
        "    Ingrese el Nombre del Ingrediente a Eliminar: ", "cyan");
    dataString = this->readLinePrompt(oss.str());
    oss << dataString << endl;

    searchedIngredient.setNameInredient(dataString);
    dataInt = this->ingredientList.findDataL(searchedIngredient);

    if (dataInt == -1) {
        oss << "Ingrediente " << this->insertColorText("No Encontrado", "red")
            << endl;
    }
    else {
        oss << this->insertColorText(
            "¿Esta Seguro que Desea Eliminar Este Ingrediente? (S/N): ",
            "yellow");
        op = this->readChar(oss.str(), "S,N");
        oss << op << endl;
        if (op == 'S') {
            this->ingredientList.deleteData(dataInt);
            oss << this->centerText(this->insertColorText(
                "!Ingrediente Eliminado Con Exito!", "green"));
        } else
            oss << this->centerText(
                this->insertColorText("Eliminacion Cancelada", "red")));
    }
}
```



```
oss << " X ¿Desea Eliminar Más Ingredientes? (S/N): ";

op = this->readChar(oss.str(), "S,N");

} while (op != 'N' && !this->ingredientList.isEmpty());
}

void IngredientMenu::modifyIngredient() {
    if (this->ingredientList.isEmpty()) {
        this->errorMessage("No Hay Ingredientes en Esta Receta\nRegresando...");
        return;
    }
    ostringstream oss;
    string dataString;
    char op, attributeModify;
    Ingredient searched;
    float dataFloat;
    int dataInt;

    try {
        do {
            oss.str("");
            oss.clear();

            oss << this->windowHeader(
                "RECETARIO DIGITAL - MODIFICAR INGREDIENTES");

            oss << this->centerText(
                " " +
                this->insertColorText(
                    "Lista de Ingredientes Actual para " + recipeName,
                    "magenta"));
            oss << this->ingredientList.toString() << endl;
            oss << this->divider();

            oss << this->insertColorText(
                "     Ingrese el nombre del Ingrediente a Modificar: ", "cyan");

            dataString = this->readLinePrompt(oss.str());
            searched.setNameInredient(dataString);
            dataInt = this->ingredientList.findDataL(searched);

            if (dataInt == -1) {
```



```
    oss << "Ingrediente " << this->insertColorText("No Encontrado",
"red")
        << endl;
}

else {
    oss << " ✎ Seleccione que Atributo Modificar" << endl;
    oss << this->insertColorText("      [A] ", "cyan")
        << "Nombre del Ingrediente" << endl;
    oss << this->insertColorText("      [B] ", "cyan")
        << "Cantidad del Ingrediente" << endl;
    oss << this->insertColorText("      [C] ", "cyan") << "Unidad de
Medida"
        << endl;

atributeModify = this->readChar(oss.str(), "A,B,C");

switch (atributeModify) {
    case 'A':
        oss << this->insertColorText(
            "      Ingrese el Nuevo Nombre del Ingrediente: ", "cyan");
        dataString = this->readLinePrompt(oss.str());
        this->ingredientList.retrieve(dataInt)->setNameInredient(
            dataString);
        break;
    case 'B':
        oss << this->insertColorText(
            "      Ingrese la Nueva Cantidad del Ingrediente: ", "cyan");
        dataFloat = this->readFloat(oss.str(), 0, 99999);
        this->ingredientList.retrieve(dataInt)->setAmount(dataFloat);
        break;
    case 'C':
        oss << this->insertColorText(
            "      Ingrese la Nueva Unidad de Medida Ingrediente: ",
"cyan");
        dataString = this->readLinePrompt(oss.str());
        this->ingredientList.retrieve(dataInt)->setUnit(dataString);
        break;
}

oss << this->centerText(
    this->insertColorText("¡Modificación Hecha Con Exito!",
"green"));
}
```



```
oss << " X ¿Desea Modificar Más Ingredientes? (S/N): ";

op = this->readChar(oss.str(), "S,N");

} while (op != 'N');
} catch (const InputExceptions::OperationCanceledException& ex) {
    this->errorMessage("Operación Cancelada\nRegresando...");
}
}

IngredientMenu::IngredientMenu()
: ingredientList(*new List<Ingredient,
Configure::maximumIngredientSize>),
recipeName("default") {}

IngredientMenu::IngredientMenu(
    List<Ingredient, Configure::maximumIngredientSize>& l,
    const std::string& r)
: ingredientList(l), recipeName(r) {}

IngredientMenu::IngredientMenu(const IngredientMenu& other)
: ingredientList(other.ingredientList), recipeName(other.recipeName) {}

void IngredientMenu::mainMenu() {
    if (this->ingredientList.isEmpty()) {
        this->errorMessage("No hay Ingredientes Registrados\nRegresando...");
        return;
    }
    ostringstream oss;
    char op;

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

        oss << this->windowHeader(
            "RECETARIO DIGITAL - MODIFICAR LISTA DE INGREDIENTES");
        oss << this->ingredientList.toString();
        oss << this->divider("-");
        oss << this->insertColorText(" [A] ", "cyan") << "Agregar
Ingredientes"
            << endl;
        oss << this->insertColorText(" [B] ", "cyan") << "Eliminar
Ingredientes"
            << endl;
    }
```



```
oss << this->insertColorText("      [C] ", "cyan") << "Modificar
Ingredientes"
    << endl;
oss << this->insertColorText("      [R] ", "cyan") << "Regresar" << endl;
oss << "Ingrese una Opción: ";
op = this->readChar(oss.str(), "A,B,C,R");
switch (op) {
    case 'A':
        this->addIngredients();
        break;
    case 'B':
        this->deleteIngredients();
        break;
    case 'C':
        this->modifyIngredient();
        break;
}
} while (op != 'R');
}
```



main.cpp

```
#include <windows.h>
#include "menu.hpp"

int main(){
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);
    Menu menu;

    return 0;
}
```



menu.cpp

```
#include "menu.hpp"

using namespace std;

bool Menu::handleOption(const string& prompt) {
    string validOptions("A,B,C,D,E,F,G,H,I,S");
    char op;

    op = this->readChar(prompt, validOptions);

    switch(op){
        case 'A':
            this->addRecipe();
            return true;
            break;
        case 'B':
            this->deleteRecipe();
            return true;
            break;
        case 'C':
            this->editRecipe();
            return true;
            break;
        case 'D':
            this->searchMenu();
            return true;
            break;
        case 'E':
            this->showRecipes();
            return true;
            break;
        case 'F':
            this->sortRecipes();
            return true;
            break;
        case 'G':
            this->saveToDisk();
            return true;
            break;
        case 'H':
            this->readFromDisk();
            return true;
    }
}
```



```
        break;
    case 'I':
        this->exportRecipe();
        return true;
        break;
    case 'S':
        this->exitMenu();
        return false;
        break;
    }
}

void Menu::showStartupAnimation(int duration_ms,
                                bool showProgressBar,
                                const std::string& alumno,
                                const std::string& profesor,
                                const std::string& materia,
                                const std::string& fecha) const
{
    // ASCII logo (personalizable)
    const char* logo[] = {
        R"(
          _____ _ _ _ _ _ - _ _ _ -
          | _ \| / \ | _ | _ | / \ | _ \| / \ | | | | |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
        )",
        R"(
          _____ _ _ _ _ _ - _ _ _ -
          | _ \| / \ | _ | _ | / \ | _ \| / \ | | | | |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
        )",
        R"(
          _____ _ _ _ _ _ - _ _ _ -
          | _ \| / \ | _ | _ | / \ | _ \| / \ | | | | |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
        )",
        R"(
          _____ _ _ _ _ _ - _ _ _ -
          | _ \| / \ | _ | _ | / \ | _ \| / \ | | | | |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
        )",
        R"(
          _____ _ _ _ _ _ - _ _ _ -
          | _ \| / \ | _ | _ | / \ | _ \| / \ | | | | |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
          | _ | _ | / \ | _ | _ | / \ | _ | _ | / \ |
        )"
    };
    const int logoLines = sizeof(logo) / sizeof(logo[0]);

    // Box configuration
    const int boxWidth = 90;           // ancho total del recuadro
    const int innerWidth = boxWidth - 4; // espacio interior para texto
    const int topMarginLines = 1;

    auto printBoxLineCentered = [&](const std::string& text){
        int txtLen = static_cast<int>(text.size());
        int padding = std::max(0, innerWidth - txtLen);
        int leftPad = padding / 2;
        int rightPad = padding - leftPad;
```



```
        std::cout << "| " << std::string(leftPad, ' ') << text <<
std::string(rightPad, ' ') << " |\n";
};

// Helper to print a labeled field with an underline area (form)
auto printField = [&](const std::string& label, const std::string&
value){
    std::string lbl = label + ": ";
    int spaceForValue = innerWidth - static_cast<int>(lbl.size());
    std::string display;
    if (!value.empty()) {
        // if value fits, print it; otherwise truncate with ellipsis
        if ((int)value.size() <= spaceForValue) display = value +
std::string(spaceForValue - (int)value.size(), ' ');
        else display = value.substr(0, spaceForValue-3) + "...";
    } else {
        // underline placeholder
        display = std::string(spaceForValue, '_');
    }
    std::cout << "|\n" << lbl << display << " |\n";
};

// Clear and draw top margin
system("CLS");
for (int i = 0; i < topMarginLines; ++i) std::cout << "\n";

// Top border
std::cout << "+" << std::string(boxWidth - 2, '-') << "+\n";

// Empty spacer row
std::cout << "|\n" << std::string(innerWidth, ' ') << " |\n";

// Logo centered (reserve a few lines)
for (int i = 0; i < logoLines; ++i) {
    printBoxLineCentered(logo[i]);
}

// Empty spacer
std::cout << "|\n" << std::string(innerWidth, ' ') << " |\n";

// Title centered
printBoxLineCentered("RECETARIO DIGITAL");

// Subtitle
printBoxLineCentered("Inicializando...");
```



```
// Empty spacer
std::cout << " " << std::string(innerWidth, ' ') << " |\n";

// Metadata fields (Alumno, Profesor, Materia, Fecha)
printField("Alumno", alumno);
printField("Profesor", profesor);
printField("Materia", materia);
printField("Fecha", fecha);

// Empty spacer at bottom of box
std::cout << " " << std::string(innerWidth, ' ') << " |\n";

// Bottom border
std::cout << "+" << std::string(boxWidth - 2, '-') << "+\n\n";

// Spinner + optional progress bar below the box
const std::string spinner = "|/-\\";
const int spinnerCount = static_cast<int>(spinner.size());
const int spinnerIntervalMs = 100;
int total = std::max(1, duration_ms);
int steps = std::max(1, total / spinnerIntervalMs);

for (int step = 0; step < steps; ++step) {
    char s = spinner[step % spinnerCount];
    std::cout << "\r [" << s << "] ";
    if (showProgressBar) {
        const int barWidth = 40;
        double frac = static_cast<double>(step) /
static_cast<double>(steps);
        int pos = static_cast<int>(barWidth * frac);
        std::cout << "[";
        for (int i = 0; i < barWidth; ++i) {
            if (i < pos) std::cout << "=";
            else if (i == pos) std::cout << ">";
            else std::cout << " ";
        }
        std::cout << "] ";
        int percent = static_cast<int>(frac * 100.0);
        if (percent > 100) percent = 100;
        std::cout << std::setw(3) << percent << "%";
    } else {
        std::cout << "Cargando...";
    }
    std::cout << std::flush;
```



```
        std::this_thread::sleep_for(std::chrono::milliseconds(spinnerIntervalMs));
    }

    // Finish: force 100% display
    if (showProgressBar) {
        const int barWidth = 40;
        std::cout << "\r [*] [";
        for (int i = 0; i < barWidth; ++i) std::cout << "=";
        std::cout << "] 100%\n";
    } else {
        std::cout << "\r [*] Listo.\n";
    }

    std::cout << std::flush;

    // Short pause so the user sees final state
    std::this_thread::sleep_for(std::chrono::milliseconds(500));

    // Clear screen before entering menu (optional)
    system("CLS");
}

void Menu::insertSortedDataByCategory(const Recipe& newData) {
    if(this->sortedBy == "id")
        this->recipeBook.insertSortedData(newData);
    else if(this->sortedBy == "recipeName")
        this->recipeBook.insertSortedData(newData, Recipe::compareByName);
    else if(this->sortedBy == "preparationTime")
        this->recipeBook.insertSortedData(newData,
Recipe::compareByPreparationTime);
    else if(this->sortedBy == "category")
        this->recipeBook.insertSortedData(newData,
Recipe::compareByCategory);
    else if(this->sortedBy == "author")
        this->recipeBook.insertSortedData(newData, Recipe::compareByAuthor);
    else if(this->sortedBy == "creationDate")
        this->recipeBook.insertSortedData(newData,
Recipe::compareByCreationDate);
    else
        throw MenuExceptions::InvalidInsertCategory();
}
```



```
Recipe* Menu::searcher(const string& prompt, int& index) {
    ostringstream oss;
    char op;
    int dataInt;
    string dataString;
    Name dataName;
    Date dataDate;
    Category dataCategory;

    string colorOpts("cyan");
    Recipe searchedRecipe;

    oss << prompt << endl;
    oss << this->insertColorText("Mediante Que Parámetro Quiere Buscar Su
Receta: ", "green") << endl;
    oss << this->insertColorText("      [A] ", colorOpts) << "  ID" << endl;
    oss << this->insertColorText("      [B] ", colorOpts) << "  Nombre de la
Receta" << endl;
    oss << this->insertColorText("      [C] ", colorOpts) << "  Autor" <<
endl;
    oss << this->insertColorText("      [D] ", colorOpts) << "  Categoría
(Primera Coincidencia)" << endl;
    oss << this->insertColorText("      [R] ", colorOpts) << "  Regresar" <<
endl;
    oss << this->insertColorText("Ingrese una Opcion: ", "cyan");

    op = this->readChar(oss.str(), "A,B,C,D,R");
    oss << op << endl;

    switch (op){
        case 'A':
            oss << "\n" << this->insertColorText("      Ingrese el ID a Buscar: ",
"cyan");
            dataInt = this->readInteger(oss.str(),0,99999);
            searchedRecipe.setId(dataInt);
            index = this->recipeBook.findDataL(searchedRecipe);
            break;
        case 'B':
            oss << "\n" << this->insertColorText("      Ingrese el Nombre de la
Receta a Buscar: ", "cyan");
            dataString = this->readLinePrompt(oss.str());
            searchedRecipe.setRecipeName(dataString);
            index = this->recipeBook.findDataL(searchedRecipe,
Recipe::compareByName);
```



```
        break;
    case 'C':
        dataName = this->readName(oss.str(), this-
>insertColorText("    Ingrese el Nombre del Chef: ", "cyan"), this-
>insertColorText("    Ingrese el Apellido del Chef: ", "cyan"));
        searchedRecipe.setAuthor(dataName);
        index = this->recipeBook.findDataL(searchedRecipe,
Recipe::compareByAuthor);
        break;
    case 'D':
        oss << this->insertColorText("    Ingrese la Categoría : ", "cyan");
        oss << "    Categoría: " << this->insertColorText("[1] ", "magenta")
<< this->insertColorText("Desayuno", "green") << this->insertColorText("[2]
", "magenta") << this->insertColorText("Comida", "green") << this-
>insertColorText("[3] ", "magenta") << this->insertColorText("Cena",
"green") << this->insertColorText("[4] ", "magenta") << this-
>insertColorText("Navideño", "green") << ":" ;
        dataCategory = this->readCategory(oss.str());
        searchedRecipe.setCategory(dataCategory);
        index = this->recipeBook.findDataL(searchedRecipe,
Recipe::compareByCategory);
        break;
    case 'R':
        return nullptr;
    }

    return index == -1 ? nullptr : this->recipeBook.retrieve(index);

}

void Menu::addRecipe() {
    ostringstream oss;
    string dataString, dataString2;
    Name dataName;
    int dataInt, dataInt2;
    Category dataCategory;
    char op;

    do{
        Recipe newRecipe;

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

```



```
    this->cleanScreen();

    oss << this->windowHeader("RECETARIO DIGITAL - AÑADIR RECETA");
    oss << this->centerText("Vamos a Agregar Una Nueva Receta");
    oss << this->centerText("Ingrese " + this->insertColorText("'/'r'",
"red") + " en Cualquier Momento para Cancelar");
    oss << this->divider("-");

    try{
        oss << this->insertColorText("    Ingrese el ID Numérico del
Platillo: ", "cyan");
        while(true){
            dataInt = this->readInteger(oss.str(), 0, 9999999);
            newRecipe.setId(dataInt);
            dataInt2 = this->recipeBook.findDataL(newRecipe);
            if(dataInt2 != -1){
                dataString = this->recipeBook.retrieve(dataInt2)-
>toString();
                this->errorMessage("El ID ya esta registrado: \n" +
dataString + "\nIntentelo Nuevamente");
            }
            else
                break;
        }
        oss << dataInt << endl;

        oss << this->insertColorText("    Ingrese el Nombre del
Platillo: ", "cyan");
        dataString = this->readLinePrompt(oss.str());
        oss << dataString << endl;
        newRecipe.setRecipeName(dataString);

        dataString = this->insertColorText("    Ingrese el Nombre del
Chef: ", "cyan");
        dataString2 = this->insertColorText("    Ingrese el Apellido del
Chef: ", "cyan");

        dataName = this->readName(oss.str(), dataString, dataString2);
        oss << dataString << dataName.getFirst() << endl;
        oss << dataString2 << dataName.getLast() << endl;
        newRecipe.setAuthor(dataName);

        oss << "    Categoría: " << this->insertColorText("[1] ",
"magenta") << this->insertColorText("Desayuno", "green") << this-
>insertColorText("[2] ", "magenta") << this->insertColorText("Comida",
```



```
"green") << this->insertColorText("[3] ", "magenta") << this-
>insertColorText("Cena", "green") << this->insertColorText("[4] ",
"magenta") << this->insertColorText("Navideño", "green") << ":" ;
    dataCategory = this->readCategory(oss.str());
    oss << this->insertColorText("    Categoria Seleccionada: ",
"cyan") << this->categoryToString(dataCategory) << endl;
    newRecipe.setCategory(dataCategory);

    oss << this->insertColorText("    Ingrese el Tiempo de
Preparacion en Minutos: ", "cyan");
    dataInt = this->readInteger(oss.str(), 1, 99999);
    oss << dataInt << endl;
    newRecipe.setPreparationTime(dataInt);

    //Añadir los Ingredientes que el usuario Desee
    IngredientMenu iM(newRecipe.getIngredientList(),
newRecipe.getRecipeName());
    iM.addIngredients();

    oss << this->insertColorText("    Inserte el Procedimiento Paso
a Paso", "cyan") << endl;
    oss << "        Ingrese '' << this->insertColorText("fin",
"red") << "' para terminar." << endl;
    dataInt = 1;
    while(true){
        dataString = this->readLinePrompt(oss.str());
        if(dataString == "fin")
            break;
        oss << "    " << dataInt << ". " << dataString << endl;
        newRecipe.addStepToProcedure(dataString);
        dataInt++;
    }

    newRecipe.setCreationDate(Date());

    this->insertSortedDataByCategory(newRecipe);
    this->modify = true;

} catch(const InputExceptions::OperationCanceledException& ex){
    this->errorMessage("Operación Cancelada\nRegresando...");
    return;
}
oss << this->centerText(this->insertColorText("  !Receta Agrega
Con Éxito!", "green"));
```



```
    oss << this->insertColorText(" ? ¿Desea Ingresar Una Nueva Receta?  
(S/N): ", "yellow");  
    op = this->readChar(oss.str(), "S,N");  
  
    }while(op != 'N');  
}  
  
void Menu::deleteRecipe() {  
    if(this->recipeBook.isEmpty()){  
        this->errorMessage("No Hay Recetas Registradas\nIngrese Recetas  
Antes");  
        return;  
    }  
  
    ostringstream oss;  
    char op, ver;  
    int index = 23;  
    Recipe* objective = nullptr;  
    this->cleanScreen();  
  
    do{  
        oss.str("");  
        oss.clear();  
        oss << this->>windowHeader("RECETARIO DIGITAL - ELIMINAR UNA  
RECETA");  
        objective = this->searcher(oss.str(), index);  
  
        if(objective == nullptr){  
            oss << "Receta " << this->insertColorText("No Encontrada",  
"red") << endl;  
            oss << this->insertColorText("¿Desea Intentar Otra Búsqueda?  
(S/N): ", "yellows");  
            op = this->readChar(oss.str(), "S,N");  
        }  
        else{  
            oss << this->insertColorText("¡Receta Encontrada!", "green") <<  
endl;  
            oss << objective->toString() << endl;  
            oss << "¿Esta Seguro Que Desea Eliminarla? (S/N): ";  
            ver = this->readChar(oss.str(), "S,N");  
            oss << ver << endl;  
  
            if(ver == 'S'){

                this->recipeBook.deleteData(index);
```



```
oss << this->centerText(this->insertColorText("¡Receta
Eliminada Con Exito!", "green")) << endl;
    this->modify = true;
}
else{
    oss << this->insertColorText("      Eliminación Cancelada.",
"cyan") << endl;
}

oss << "¿Realizar Otra Eliminación? (S/N): ";
op = this->readChar(oss.str(), "S,N");
}

}while(op != 'N' && !this->recipeBook.isEmpty());

if(this->recipeBook.isEmpty())
    this->errorMessage("Ya No Hay Recetas Registradas\nVuelva a Ingresar
Más Recetas");
}

void Menu::editRecipe() {
if(this->recipeBook.isEmpty()){
    this->errorMessage("No Hay Recetas Registradas.\nRegresando..."); 
    return;
}

char op, repeater;
int dataInt, dataInt2;
string dataString;
Name dataName;
Category dataCategory;
Date dataDate;
ostringstream oss;
int index;
Recipe* objetivo;
Recipe searched;

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

    this->cleanScreen();
    oss << this->windowHeader("RECETARIO DIGITAL - EDITAR RECETA");
    objetivo = this->searcher(oss.str(), index);
```



```
if(objetive != nullptr){
    oss << objetivo->toString() << endl;
    oss << this->centerText("Seleccione el Apartado a Modificar: ")
<< endl;
    oss << this->divider("-");

    oss << this->insertColorText("[A] ", "cyan") << "Id" << endl;
    oss << this->insertColorText("[B] ", "cyan") << "Nombre de la
Receta" << endl;
        oss << this->insertColorText("[C] ", "cyan") << "Autor" << endl;
        oss << this->insertColorText("[D] ", "cyan") << "Categoría" <<
endl;
        oss << this->insertColorText("[E] ", "cyan") << "Tiempo de
Preparación" << endl;
        oss << this->insertColorText("[F] ", "cyan") << "Procedimiento"
<< endl;
        oss << this->insertColorText("[G] ", "cyan") << "Lista de
Ingredientes" << endl;
        oss << this->insertColorText("[R] ", "cyan") << "Regresar" <<
endl;
    oss << this->divider("-");
    oss << "Seleccione una opción: ";
    op = this->readChar(oss.str(), "A,B,C,D,E,F,G,H,R");
    oss << op << endl;

    try{
        switch(op){
            case 'A':
                oss << this->insertColorText("     Ingrese el nuevo
ID: ", "yellow");
                dataInt = this->readInteger(oss.str(), 0, 99999);
                searched.setId(dataInt);
                dataInt2 = this->recipeBook.findDataL(searched);
                if(dataInt2 != -1)
                    this->errorMessage("ID ya registrado: \n" +
this->recipeBook.retrieve(dataInt2)->toString());
                else{
                    objetivo->setId(dataInt);
                    oss << dataInt << endl;
                }
                break;
            case 'B':
                oss << this->insertColorText("     Ingrese el Nuevo
Nombre de la Receta: ", "yellow");
    
```



```
        dataString = this->readLinePrompt(oss.str());
        objetive->setRecipeName(dataString);
        oss << dataString << endl;
        break;
    case 'C':
        dataName = this->readName(oss.str(), this-
>insertColorText("    Inserte el Nuevo Nombre del Chef: ", "yellow"), this-
>insertColorText("    Ingrese el Nuevo Apellido del Chef: ", "yellow"));
        oss << this->insertColorText("    Ingrese el Nuevo
Nombre del Chef: ", "yellow") << dataName.getFirst() << endl;
        oss << this->insertColorText("    Ingrese el Nuevo
Apellido del Chef: ", "yellow") << dataName.getLast() << endl;
        objetive->setAuthor(dataName);
        break;
    case 'D':
        oss << this->insertColorText("    Ingrese la Nueva
Categoría de la Receta: ", "yellow");
        oss << "    Categoría: " << this-
>insertColorText("[1] ", "magenta") << this->insertColorText("Desayuno",
"green") << this->insertColorText("[2] ", "magenta") << this-
>insertColorText("Comida", "green") << this->insertColorText("[3] ",
"magenta") << this->insertColorText("Cena", "green") << this-
>insertColorText("[4] ", "magenta") << this->insertColorText("Navideño",
"green") << ":" ;
        dataCategory = this->readCategory(oss.str());
        oss << this->categoryToString(dataCategory) << endl;
        objetive->setCategory(dataCategory);
        break;
    case 'E':
        oss << this->insertColorText("    Ingrese el Nuevo
Tiempo de Preparación en Minutos: ", "yellow");
        dataInt = this->readInteger(oss.str(), 0, 999999);
        oss << dataInt << endl;
        objetive->setPreparationTime(dataInt);
        break;
    case 'F':
    {
        ProcedureMenu mP(objetive->getProcedureList(),
objetive->getRecipeName());
        mP.mainMenu();
    }
        break;
    case 'G':
    {
```



```
IngredientMenu iM(objective-
>getIngredientList(), objective->getRecipeName());
    iM.mainMenu();
}
break;
case 'H':
break;
case 'R':
return;
break;

}
} catch(const InputExceptions::OperationCanceledException& ex){
    this->errorMessage("Operación Cancelada\nRegresando...");
    return;
}
if(index != -1){
    oss << this->centerText(this->insertColorText("Modificación
Realizada con Exito", "green"));
    this->modify = true;
}
}

else
    this->errorMessage("Receta No Encontrada\nRegresando..");

oss << "¿Desea Realizar Otra Modificación? (S/N): ";
repeater = this->readChar(oss.str(), "S,N");
} while(repeater != 'N');
}

void Menu::showRecipes() {
if(this->recipeBook.isEmpty()){
    this->errorMessage("No Hay Recetas Que Mostar\nRegresando..");
    return;
}

ostringstream oss;
char op, repeater;
Category dataCategory;
Recipe comparative;

do{
    oss.str("");

```



```
oss.clear();
oss << this->>windowHeader(" ⑩ RECETARIO DIGITAL - MOSTRAR RECETAS");
oss << this->centerText(this->insertColorText("Mostrar Todas las
Recetas o Filtradas", "blue")) << endl;
oss << this->divider("-");

oss << this->insertColorText("      [A] ", "cyan") << "Mostrar Todas
las Recetas" << endl;
oss << this->insertColorText("      [B] ", "cyan") << "Filtrar Por
Categoría" << endl;
oss << this->insertColorText("      [R] ", "cyan") << "Regresar" <<
endl;
oss << this->insertColorText("Seleccione Una Opción: ", "yellow");
op = this->readChar(oss.str(), "A,B,R");

oss << op << endl;
try{
    switch(op){
        case 'A':
            oss.str("");
            oss.clear();
            oss << this->>windowHeader("TODAS LAS RECETAS");
            oss << this->recipeBook.toString() << endl;
            oss << this->divider();
            break;

        case 'B':
            oss << this->insertColorText("Ingrese /r para Cancelar",
"red") << endl;
            oss << this->insertColorText("      Seleccione una
Categoría ", "yellow");
            oss << "      Categoría: " << this->insertColorText("[1]
", "magenta") << this->insertColorText("Desayuno", "green") << this-
>insertColorText("[2] ", "magenta") << this->insertColorText("Comida",
"green") << this->insertColorText("[3] ", "magenta") << this-
>insertColorText("Cena", "green") << this->insertColorText("[4] ",
"magenta") << this->insertColorText("Navideño", "green") << ":" ;
            dataCategory = this->readCategory(oss.str());
            comparative.setCategory(dataCategory);

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

            oss << this->>windowHeader("TODAS RECETAS TIPO: " + this-
>categoryToString(dataCategory));
```



```
oss << this->recipeBook.toString(comparative,
Recipe::compareByCategory);
    oss << this->divider();
    break;
    case 'R':
        return;
        break;
    }
} catch(const InputExceptions::OperationCanceledException& ex){
    this->errorMessage("Operación Cancelada.\nRegresando...");  

    return;
}
oss << "¿Desea Repetir el Menú? (S/N) ✓: ";
repeater = this->readChar(oss.str(), "S,N");
} while(repeater != 'N');

}

void Menu::searchMenu() {
if(this->recipeBook.isEmpty()){
    this->errorMessage("No hay Recetas Registradas.\nRegresando...");  

    return;
}
ostringstream oss;
Recipe* searched;
char op;
int aux;

do{
    oss.str("");
    oss.clear();
    oss << this->windowHeader(" RECTARIO DIGITAL - BUSCAR RECETA");
    searched = this->searcher(oss.str(), aux);

    if(searched == nullptr)
        oss << this->centerText( "Receta " + this->insertColorText("No
Encontrada", "red"));
    else
        oss << searched->toString();
    oss << this->insertColorText("¿Desea Buscar Otra Receta (S/N)?: ",
"yellow");
    op = this->readChar(oss.str(), "S,N");
} while(op != 'N');

}
```



```
void Menu::sortRecipes(){
    if(this->recipeBook.getLastPosition() + 1 < 2){
        this->errorMessage("No Hay Suficientes Recetas Registradas.\nNo Se
Requiere Ordenamiento.");
        return;
    }
    ostringstream oss;
    Recipe categorySort;
    char op;
    string dataString;
    Name dataName;
    Date dateTime;

    oss << this->windowHeader(" RECETARIO DIGITA - ORDENAR RECETAS");
    oss << this->insertColorText("Seleccione Sobre Qué Categoría Quiere
Ordenar Sus Recetas: ", "yellow") << endl;
    oss << this->divider();

    oss << this->insertColorText(" [A] ", "cyan") << "Ordenar por ID" <<
endl;
    oss << this->insertColorText(" [B] ", "cyan") << "Ordenar por Nombre
de la Receta" << endl;
    oss << this->insertColorText(" [C] ", "cyan") << "Ordenar por Nombre
del Autor" << endl;
    oss << this->insertColorText(" [D] ", "cyan") << "Ordenar por
Categoría" << endl;
    oss << this->insertColorText(" [E] ", "cyan") << "Ordenar por Tiempo
de Preparación" << endl;
    oss << this->insertColorText(" [F] ", "cyan") << "Ordenar por Por
Fecha de Creación" << endl;
    oss << this->insertColorText(" [S] ", "cyan") << "Salir" << endl;
    oss << this->insertColorText("Seleccione una Opción: ", "yellow");

    op = this->readChar(oss.str(), "A,B,C,D,E,F,S");
    oss << op << endl;

    switch(op){
        case 'A':
            this->recipeBook.sortDataQuick();
            this->sortedBy = "id";
            break;
        case 'B':
            this->recipeBook.sortDataQuick(Recipe::compareByName);
            this->sortedBy = "recipeName";
            break;
    }
}
```



```
        case 'C':
            this->recipeBook.sortDataQuick(Recipe::compareByAuthor);
            this->sortedBy = "author";
            break;
        case 'D':
            this->recipeBook.sortDataQuick(Recipe::compareByCategory);
            this->sortedBy = "category";
            break;
        case 'E':
            this-
>recipeBook.sortDataQuick(Recipe::compareByPreparationTime);
            this->sortedBy = "preparationTime";
            break;
        case 'F':
            this->recipeBook.sortDataQuick(Recipe::compareByCreationDate);
            this->sortedBy = "creationDate";
            break;
        case 'S':
            return;
            break;
    }
    oss << this->centerText(this->insertColorText("¡Datos Ordenados Con
Éxito!", "green"));
    this->cleanScreen();
    cout << oss.str();
    system("PAUSE");
}

void Menu::saveToDisk() {
    if(this->recipeBook.isEmpty()){
        this->errorMessage("No Hay Recetas Que Guardar\nRegresando...");
        return;
    }
    string dataString;
    ostringstream oss;

    oss << this->windowHeader(" RECETARIO DIGITAL - GUARDAR RECETAS EN EL
DISCO ");
    oss << this->centerText(this->insertColorText("Ingrese '/r' para
Cancelar la Operación", "yellow")) << endl;
    oss << this->divider("-");
    oss << this->insertColorText("Ingrese el Nombre con el Cual Quiere
Guardar al Disco (Se Guardará en Formato .Json): ", "yellow");

    try{
```



```
dataString = this->readLinePrompt(oss.str());  
  
ofstream dataFile(dataString + ".json");  
  
if(dataFile.is_open()){  
    dataFile << this->recipeBook.toJson().dump(4);  
    oss << dataString + ".json" << endl;  
    oss << this->centerText(this->insertColorText("  ¡Datos  
Guardados Con Éxito!", "green"));  
    dataFile.close();  
    this->modify = false;  
}  
else  
    oss << this->centerText(this->insertColorText("  Ha Ocurrido  
Un Error Al Crear el Archivo.", "red"));  
  
} catch(const InputExceptions::OperationCanceledException& ex){  
    this->errorMessage("Operación Cancelada\nRegresando");  
    return;  
}  
  
this->cleanScreen();  
oss << this->insertColorText("Regresando...", "magenta");  
cout << oss.str() << endl;  
system("PAUSE");  
  
}  
  
void Menu::readFromDisk() {  
    ostringstream oss;  
    string dataString;  
  
    oss << this->windowHeader("  RECETARIO DIGITAL - LEER RECETAS DEL  
ALMACENAMIENTO");  
    oss << this->insertColorText("ADVERTENCIA: Si se leen datos con  
registros en la aplicación\n estos serán sobrescritos", "red") << endl;  
    oss << this->centerText(this->insertColorText("Ingrese '/r' para  
cancelar", "cyan")) << endl;  
    oss << this->divider("-");  
    oss << this->insertColorText("Ingrese el Nombre de la Base de Datos  
.json (sin la extensión):", "yellow");  
  
    try{  
        dataString = this->readLinePrompt(oss.str());  
        oss << dataString << endl;
```



```
ifstream dataFile(dataString + ".json");

if(dataFile.is_open()){
    nlohmann::json js;
    dataFile >> js;
    this->recipeBook.fromJson(js.at("data"));
    dataFile.close();
    oss << dataString << ".json" << endl;
    oss << this->centerText(this->insertColorText("  ¡Datos
Cargados Con Éxito!", "green")) << endl;

    //Recuperar el Tipo de Orden que Tenía
    if(this->recipeBook.isSorted())
        this->sortedBy = "id";
    else if(this->recipeBook.isSorted(Recipe::compareByName))
        this->sortedBy = "recipeName";
    else if(this->recipeBook.isSorted(Recipe::compareByAuthor))
        this->sortedBy = "author";
    else if(this->recipeBook.isSorted(Recipe::compareByCategory))
        this->sortedBy = "category";
    else if(this-
>recipeBook.isSorted(Recipe::compareByCreationDate))
        this->sortedBy = "creationDate";
    }
    else{
        oss << this->centerText(this->insertColorText("No se encontró o
no se pudo abrir el archivo.", "red")) << endl;
    }
}

} catch(const InputExceptions::OperationCanceledException& ex){
    this->errorMessage("Operación Cancelada\nRegresando...");
    return;
}

this->cleanScreen();
oss << this->insertColorText("Regresando...", "magenta");
cout << oss.str() << endl;
system("PAUSE");
}

void Menu::exportRecipe(){
if(this->recipeBook.isEmpty()){
    this->errorMessage("No Hay Recetas Que Exportar\n");
    return;
}
```



```
}

ostringstream oss;
Recipe* r;
int index;
string dataString;
char op;

do{
    oss.str("");
    oss.clear();
    oss << this->windowHeader("RECETARIO DIGITAL - EXPORTAR UNA
RECETA.");
    r = this->searcher(oss.str(), index);

    if(r == nullptr){
        oss << this->centerText(this->insertColorText("Receta No
Encontrada", "red")) << endl;
    }
    else{
        oss << r->toString() << endl;
        oss << this->divider("-");
        oss << this->insertColorText("Ingrese el Nombre del Archivo
Destino (con extensión): ", "cyan");
        dataString = this->readLinePrompt(oss.str());
        oss << dataString << endl;
        ofstream os(dataString);
        if(os.is_open()){
            os << this->clearColor(r->toString());
            os.close();
            oss << this->centerText(this->insertColorText("!Exportación
Realizada con Éxito!", "green")) << endl;
        }
        else{
            this->insertColorText("El Archivo no se Puede Crear",
"red");
        }
    }

    oss << this->insertColorText("¿Desea Realizar Otra Exportación?
(S/N): ", "yellow");
    op = this->readChar(oss.str(), "S,N");

} while(op != 'N');
}
```



```
void Menu::exitMenu(){
    ostringstream oss;
    char op;
    if(this->modify){
        oss << this->windowHeader("👉 RECETARIO DIGITAL - SALIENDO DEL
RECETARIO");
        oss << "Hubo Cambios en las Recetas Aún no Guardados" << endl;
        oss << this->insertColorText("      ¿Desea Guardar los Cambios Antes
de Salir? (S/N): ", "cyan");
        op = this->readChar(oss.str(), "S,N");
        oss << op << endl;

        if(op == 'S')
            this->saveToDisk();
    }

    oss.str("");
    oss.clear();
    oss << this->divider();
    oss << this->centerText(this->insertColorText("Saliendo del Programa.", "cyan")) << endl;
    oss << this->centerText(this->insertColorText("Que Tenga un Lindo Día
:D", "cyan")) << endl;
    oss << this->divider();

    this->cleanScreen();
    cout << oss.str();
    system("PAUSE");
}

void Menu::mainMenu() {
    bool repeater;
    ostringstream oss;
    string colorOpts = "cyan";

    do{
        oss.str("");
        oss.clear();
        this->cleanScreen();
        oss << this->windowHeader("⌚ MENU PRINCIPAL - GESTIÓN DE RECETAS");

        oss << "      " << this->insertColorText("[A]", colorOpts) << " 🍳
Agregar Nueva Receta" << endl;
    }
}
```



```
    oss << "      " << this->insertColorText("[B]", colorOpts) << " X
    Eliminar Una Receta" << endl;
    oss << "      " << this->insertColorText("[C]", colorOpts) << " ✎
    Modificar Una Receta" << endl;
    oss << "      " << this->insertColorText("[D]", colorOpts) << " 🔎
    Buscar Una Receta" << endl;
    oss << "      " << this->insertColorText("[E]", colorOpts) << " 📄
    Mostrar Recetas" << endl;
    oss << "      " << this->insertColorText("[F]", colorOpts) << " 📄
    Ordenar El Catalogo de Recetas" << endl;
    oss << "      " << this->insertColorText("[G]", colorOpts) << " 🗂️
    Guardar al Disco" << endl;
    oss << "      " << this->insertColorText("[H]", colorOpts) << " 📩
    Cargar Recetas del Disco" << endl;
    oss << "      " << this->insertColorText("[I]", colorOpts) << " 📄
    Exportar Una Receta" << endl;
    oss << "      " << this->insertColorText("[S]", colorOpts) << " ⌚
    Salir del Programa" << endl;

    oss << this->divider() << endl;

    oss << "Seleccione una Opcion [A...S]: ";
    repeater = this->handleOption(oss.str());

} while(repeater);
}

Menu::Menu() : recipeBook(* new List<Recipe>), sortedBy("id"), modify(false)
{
    this->showStartupAnimation();
    this->mainMenu();
}

Menu::Menu(const Menu& other) : recipeBook(other.recipeBook) {
    this->showStartupAnimation();
    this->mainMenu();
}

Menu::Menu(List<Recipe>& l, const std::string& s, const bool& m) :
recipeBook(l), sortedBy(s), modify(m) {
    this->showStartupAnimation();
    this->mainMenu
}
```



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());
}

nlohmann::json Name::toJson() const {
    nlohmann::json js{
        {"Last", this->last},
        {"First", this->first},
    };

    return js;
}

void Name::fromJson(const nlohmann::json& js) {
    this->last = js.at("Last").get<std::string>();
    this->first = js.at("First").get<std::string>();
}
```



proceduremenu.cpp

```
#include "proceduremenu.hpp"

using namespace std;

void ProcedureMenu::mainMenu() {
    if (this->process.isEmpty()) {
        this->errorMessage("No hay Pasos Registrados\nRegresando... ");
        return;
    }
    ostringstream oss;
    char op;

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

        oss << this->windowHeader("RECETARIO DIGITAL - MODIFICAR LISTA DE
PASOS");
        oss << this->process.toString(true);
        oss << this->divider("-");
        oss << this->insertColorText(" [A] ", "cyan") << "Agregar Pasos" <<
endl;
        oss << this->insertColorText(" [B] ", "cyan") << "Eliminar Pasos"
            << endl;
        oss << this->insertColorText(" [C] ", "cyan") << "Modificar Pasos"
            << endl;
        oss << this->insertColorText(" [R] ", "cyan") << "Regresar" << endl;
        oss << "Ingrese una Opción: ";
        op = this->readChar(oss.str(), "A,B,C,R");
        switch (op) {
            case 'A':
                this->addProcedure();
                break;
            case 'B':
                this->deleteProcedure();
                break;
            case 'C':
                this->editProcedure();
                break;
        }
    } while (op != 'R');
}
```



```
}
```

```
void ProcedureMenu::addProcedure() {
    ostringstream oss;
    char op;
    string dataString;
    StringWrapper newStep;
    try {
        do {
            oss.str("");
            oss.clear();

            oss << this->windowHeader("RECETARIO DIGITAL - AÑADIENDO PASOS");
            if (!this->process.isEmpty()) {
                oss << this->centerText(
                    "📋 " + this->insertColorText(
                        "Lista de Pasos Actual para " + recipeName,
                        "magenta"));
                oss << this->process.toString(true) << endl;
                oss << this->divider();
            }
            oss << this->insertColorText("    Ingrese '/r' para Cancelar", "red")
                << endl;
            oss << this->insertColorText("    Ingrese el Nuevo Paso Para La
Receta: ",
                "cyan");
            newStep = this->readLinePrompt(oss.str());

            oss << newStep << endl;

            this->process.insertElement(newStep, this->process.getLastPosition() +
1);

            oss << this->divider("-");
            oss << "☑ ";
            oss << this->insertColorText("Paso Añadido con Exito!", "green") <<
endl;
            oss << " 🎉 ¿Desea Añadir Más Pasos? (S/N): ";

            op = this->readChar(oss.str(), "S,N");

        } while (op != 'N');
    } catch (const InputExceptions::OperationCanceledException&) {
        this->errorMessage("Operación Cancelada.\nRegresando...");
        return;
    }
}
```



```
        }

    }

void ProcedureMenu::deleteProcedure() {
    if (this->process.isEmpty()) {
        this->errorMessage("La Lista de Pasos está Vacía.\nNo Hay que
Eliminar");
        return;
    }

    ostringstream oss;
    char op;
    int dataInt;
    string dataString;
    StringWrapper searchedStep;

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

        oss << this->windowHeader("RECETARIO DIGITAL - ELIMINAR PASOS");
        if (!this->process.isEmpty()) {
            oss << this->centerText(
                "👉 " + this->insertColorText(
                    "Lista de Pasos Actual para " + recipeName,
                    "magenta"));
            oss << this->process.toString(true) << endl;
            oss << this->divider();
        }
        oss << this->centerText("Ingrese 'fin' para regresar");
        oss << this->insertColorText("    Ingrese el Número del Paso a Eliminar:
",
                                         "cyan");

        while (true) {
            try {
                dataString = this->readLinePrompt(oss.str());
                if (this->standarString(dataString) == "FIN") {
                    oss << this->centerText(
                        this->insertColorText("Regresando...", "blue"));
                    return;
                }
                dataInt = stoi(dataString) - 1;
            }
        }
    }
}
```



```
    this->process.retrieve(dataInt);
    oss << dataInt + 1 << endl;
    oss << this->insertColorText(
        "¿Esta Seguro que Desea Eliminar Este Ingrediente? (S/N): ",
        "yellow");
    op = this->readChar(oss.str(), "S,N");
    oss << op << endl;
    if (op == 'S') {
        this->process.deleteData(dataInt);
        oss << this->centerText(
            this->insertColorText("!Paso Eliminado Con Exito!", "green"));
    } else
        oss << this->centerText(
            this->insertColorText("Eliminacion Cancelada", "red"));
    break;
} catch (const DataContainersExceptions::InvalidPosition& ex) {
    oss << this->centerText(
        this->insertColorText("Posición Inválida", "red"));
} catch (const InputExceptions::OperationCanceledException& ex) {
    this->errorMessage("Operación Cancelada.\nRegresando...");
    return;
} catch (const invalid_argument& ex) {
    this->errorMessage("Entrada No Numérica\nInténtelo Nuevamente");
}
}
oss << " ✗ ¿Desea Eliminar Más Ingredientes? (S/N): ";

op = this->readChar(oss.str(), "S,N");

} while (op != 'N' && !this->process.isEmpty());

this->errorMessage("Ya No Hay Pasos Registrados.\nRegresando...");
}

void ProcedureMenu::editProcedure() {
if (this->process.isEmpty()) {
    this->errorMessage("No Hay Ingredientes en Esta Receta\nRegresando...");
    return;
}
ostringstream oss;
string dataString;
char op;
StringWrapper* searched;
int dataInt;
```



```
try {
    do {
        oss.str("");
        oss.clear();

        oss << this->windowHeader("RECETARIO DIGITAL - MODIFICAR PASOS");
        oss << this->centerText(
            "📋 " + this->insertColorText(
                "Lista de Pasos Actual para " + recipeName,
                "magenta"));
        oss << this->process.toString(true) << endl;
        oss << this->divider();
        oss << this->insertColorText("    Ingrese '/r' Para Cancelar", "red")
            << endl;
        oss << this->insertColorText(
            "    Ingrese el Índice del Paso a Modificar: ", "cyan");

        dataInt = this->readInteger(oss.str(), 0, 999999);
        oss << dataInt << endl;

        try {
            searched = this->process.retrieve(dataInt - 1);
            oss << this->insertColorText(
                "    Ingrese la Modificación para el Paso: ", "yellow");
            *searched = this->readLinePrompt(oss.str());
            oss << *searched << endl;
            oss << this->centerText(
                this->insertColorText("¡Modificación Hecha Con Exito!", "green"));
        } catch (const DataContainersExceptions::InvalidPosition& ex) {
            oss << this->centerText("Paso" +
                this->insertColorText("No Encontrado", "red"));
        }
    }

    oss << " ✎ ¿Desea Modificar Más Ingredientes? (S/N): ";
    op = this->readChar(oss.str(), "S,N");

} while (op != 'N');
} catch (const InputExceptions::OperationCanceledException& ex) {
    this->errorMessage("Operación Cancelada\nRegresando...");
}
}
```

ProcedureMenu::ProcedureMenu()



```
: process(*new List<StringWrapper, Configure::maximunProcedureSize>),
    recipeName("default") {}
ProcedureMenu::ProcedureMenu(
    List<StringWrapper, Configure::maximunIngredientSize>& l,
    const std::string& r)
: process(l), recipeName(r) {}
ProcedureMenu::ProcedureMenu(const ProcedureMenu& other)
: process(other.process), recipeName(other.recipeName) {}
```



recipe.cpp

```
#include "recipe.hpp"

Recipe::Recipe()
    : id(-1),
      recipeName("default"),
      category(DESAYUNO),
      preparationTime(-1),
      procedureList(*new List<StringWrapper,
Configure::maximunIngredientSize>),
      ingredientList(*new List<Ingredient,
Configure::maximunIngredientSize>),
      creationDate(Date()) {}

Recipe::Recipe(const Recipe& other)
    : id(other.id),
      recipeName(other.recipeName),
      author(other.author),
      category(other.category),
      preparationTime(other.preparationTime),
      procedureList(other.procedureList),
      ingredientList(other.ingredientList),
      creationDate(other.creationDate) {}

Recipe::Recipe(const int& i,
              const std::string& rN,
              const Name& n,
              const Category& c,
              const int& pT,
              const List<StringWrapper, Configure::maximunIngredientSize>&
pL,
              const List<Ingredient, Configure::maximunIngredientSize>& iL,
              const Date& cD)
    : id(i),
      recipeName(rN),
      author(n),
      category(c),
      preparationTime(pT),
      procedureList(pL),
      ingredientList(iL),
      creationDate(cD) {}

int Recipe::getId() const {
    return this->id;
```



```
}

std::string Recipe::getRecipeName() const {
    return this->recipeName;
}

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

Category Recipe::getCategory() const {
    return this->category;
}

int Recipe::getPreparationTime() const {
    return this->preparationTime;
}

List<StringWrapper, Configure::maximunIngredientSize>&
Recipe::getProcedureList() {
    return this->procedureList;
}

List<Ingredient, Configure::maximunIngredientSize>&
Recipe::getIngredientList() {
    return this->ingredientList;
}

Date Recipe::getCreationDate() const {
    return this->creationDate;
}

std::string Recipe::toString(const std::string& format) const {
    std::ostringstream oss;

    if (format == "full") {
        oss << "\033[36mID de la Receta: \033[37m" << this->id << std::endl;
        oss << "\033[36mFecha de Creación \033[37m: "
            << this->creationDate.toString() << std::endl;
        oss << "\033[36mNombre de la Receta \033[37m: " << this->recipeName
            << std::endl;
        oss << "\033[36mAutor \033[37m: " << this->author.toString() <<
        std::endl;
        oss << "\033[36mCategoría \033[37m: ";
        switch (this->category) {
```



```
        case DESAYUNO:
            oss << "Desayuno";
            break;
        case COMIDA:
            oss << "Comida";
            break;
        case CENA:
            oss << "Cena";
            break;
        case NAVIDEÑO:
            oss << "Navideño";
            break;
    }
    oss << std::endl;

    oss << "\033[36mTiempo \033[37m:      " << this->preparationTime
        << " minutos." << std::endl;
    oss << std::setfill('-');
    oss << std::setw(53) << "" << std::endl;
    oss << std::setfill(' ');
    oss << "\033[33mIngredientes \033[37m: " << std::endl;
    oss << this->ingredientList.toString();
    oss << std::setfill('-');
    oss << std::setw(53) << "" << std::endl;
    oss << std::setfill(' ');
    oss << "\033[35mProcedimiento \033[37m: " << std::endl;
    oss << this->procedureList.toString(true);
} else if (format == "table") {
    // Truncar strings largos para que quepan en columnas
    std::string truncName = recipeName.length() > 22
        ? recipeName.substr(0, 19) + "..."
        : recipeName;

    std::string truncAuthor = author.toString().length() > 18
        ? author.toString().substr(0, 15) + "..."
        : author.toString();

    // Formato compacto con columnas alineadas
    oss << std::left << std::setw(5) << id << " | "
        << std::setw(22)
        << truncName << " | "
        << std::setw(12) << category << " | "
        << std::setw(18) << truncAuthor << " | "
        << std::right <<
    std::setw(4)
        << preparationTime << " min | "
        << std::left << std::setw(6)
        << ingredientList.getLastPosition() + 1 << " ing.";
}
```



```
        return oss.str();
    }

void Recipe::setId(const int& id) {
    if (id < 0)
        throw InputExceptions::NumberNotPositive("Id debe ser positivo");
    this->id = id;
}

void Recipe::setRecipeName(const std::string& recipeName) {
    if (recipeName.empty())
        throw InputExceptions::EmptyString(
            "El nombre de la receta no puede estar vacio.");
    this->recipeName = recipeName;
}

void Recipe::setAuthor(const Name& author) {
    this->author = author;
}

void Recipe::setCategory(const Category& category) {
    this->category = category;
}

void Recipe::setPreparationTime(const int& preparationTime) {
    if (preparationTime < 0)
        throw InputExceptions::NumberNotPositive(
            "El tiempo de preparacion debe ser positivo.");
    this->preparationTime = preparationTime;
}

void Recipe::setProcedureList(
    const List<StringWrapper, Configure::maximunIngredientSize>&
    procedureList) {
    this->procedureList = procedureList;
}

void Recipe::setIngredientList(
    const List<Ingredient, Configure::maximunIngredientSize>&
ingredientList) {
    this->ingredientList = ingredientList;
}

void Recipe::setCreationDate(const Date& date) {
```



```
this->creationDate = date;
}

void Recipe::addIngredient(const Ingredient& ingredient) {
    if (this->ingredientList.findDataL(ingredient) != -1)
        throw RecipeExceptions::RepeatedIngredient();
    this->ingredientList.insertSortedData(ingredient);
}

void Recipe::deleteIngredient(const Ingredient& ingredient) {
    if (this->ingredientList.findDataL(ingredient) == -1)
        throw RecipeExceptions::NonExistentIngredient();
    this->ingredientList.deleteData(this-
>ingredientList.findDataL(ingredient));
}

void Recipe::clearIngredients() {
    this->ingredientList.deleteAll();
}

Ingredient* Recipe::searchIngredient(const Ingredient& ingredient) {
    if (this->ingredientList.findDataL(ingredient) == -1)
        return nullptr;
    return this->ingredientList.retrieve(
        this->ingredientList.findDataL(ingredient));
}

void Recipe::addStepToProcedure(const std::string& step) {
    if (step.empty())
        throw InputExceptions::EmptyString("Procedimiento no puede estar
vacio.");
    this->procedureList.insertSortedData(step);
}

void Recipe::deleteStepFromProcedure(const int& index) {
    if (!this->procedureList.isValidPosition(index - 1))
        throw DataContainersExceptions::InvalidPosition("No existe el paso " +
                                                       std::to_string(index + 1));
    this->procedureList.deleteData(index - 1);
}

void Recipe::clearProcedure() {
    this->procedureList.deleteAll();
}
```



```
bool Recipe::operator==(const Recipe& other) const {
    return this->id == other.id;
}

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

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

bool Recipe::operator<=(const Recipe& other) const {
    return this->id <= other.id;
}

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

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

int Recipe::compare(const Recipe& recipeA, const Recipe& recipeB) {
    return recipeA.id - recipeB.id;
}

int Recipe::compareByName(const Recipe& recipeA, const Recipe& recipeB) {
    return recipeA.recipeName.compare(recipeB.recipeName);
}

int Recipe::compareByAuthor(const Recipe& recipeA, const Recipe& recipeB) {
    return recipeA.author.compareTo(recipeB.author);
}

int Recipe::compareByCategory(const Recipe& recipeA, const Recipe& recipeB)
{
    return recipeA.category - recipeB.category;
}

int Recipe::compareByPreparationTime(const Recipe& recipeA,
                                     const Recipe& recipeB) {
    return recipeA.preparationTime - recipeB.preparationTime;
```



```
        }

    int Recipe::compareByCreationDate(const Recipe& recipeA,
                                      const Recipe& recipeB) {
        return recipeA.creationDate.compareTo(recipeB.creationDate);
    }

    nlohmann::json Recipe::toJson() const {
        nlohmann::json js{
            {"id", this->id},
            {"recipe name", this->recipeName},
            {"author", this->author.toJson()},
            {"category", this->category},
            {"preparation time", this->preparationTime},
            {"procedure list", this->procedureList.toJson()},
            {"ingredient list", this->ingredientList.toJson()},
            {"creation date", this->creationDate.toJson()},
        };
        return js;
    }

    void Recipe::fromJson(const nlohmann::json& js) {
        this->id = js.at("id").get<int>();
        this->recipeName = js.at("recipe name").get<std::string>();
        this->author.fromJson(js.at("author"));
        this->category = js.at("category").get<Category>();
        this->preparationTime = js.at("preparation time").get<int>();
        this->procedureList.fromJson(js.at("procedure list").at("data"));
        this->ingredientList.fromJson(js.at("ingredient list").at("data"));
        this->creationDate.fromJson(js.at("creation date"));
    }

    Recipe& Recipe::operator=(const Recipe& other) {
        this->id = other.id;
        this->recipeName = other.recipeName;
        this->author = other.author;
        this->category = other.category;
        this->preparationTime = other.preparationTime;
        this->procedureList = other.procedureList;
        this->ingredientList = other.ingredientList;
        this->creationDate = other.creationDate;

        return *this;
    }
}
```



stringwrapped.cpp

```
#include "stringwrapped.hpp"

// ====== CONSTRUCTORES ======
StringWrapper::StringWrapper() : valor("") {}
StringWrapper::StringWrapper(const std::string& v) : valor(v) {}
StringWrapper::StringWrapper(const char* v) : valor(v) {}
StringWrapper::StringWrapper(const StringWrapper& other) :
    valor(other.valor) {}
StringWrapper::StringWrapper(StringWrapper&& other) noexcept
    : valor(std::move(other.valor)) {}
StringWrapper::StringWrapper(size_t n, char c) : valor(n, c) {}

// ====== OPERADORES DE ASIGNACIÓN ======
StringWrapper& StringWrapper::operator=(const StringWrapper& other) {
    if (this != &other) {
        valor = other.valor;
    }
    return *this;
}

StringWrapper& StringWrapper::operator=(StringWrapper&& other) noexcept {
    if (this != &other) {
        valor = std::move(other.valor);
    }
    return *this;
}

StringWrapper& StringWrapper::operator=(const std::string& v) {
    valor = v;
    return *this;
}

StringWrapper& StringWrapper::operator=(const char* v) {
    valor = v;
    return *this;
}

StringWrapper& StringWrapper::operator=(char c) {
    valor = c;
    return *this;
}

// ====== MODIFICADORES ======
void StringWrapper::clear() {
```



```
valor.clear();
}

StringWrapper& StringWrapper::insert(size_t pos, const std::string& str) {
    valor.insert(pos, str);
    return *this;
}

StringWrapper& StringWrapper::insert(size_t pos, const char* s) {
    valor.insert(pos, s);
    return *this;
}

StringWrapper& StringWrapper::erase(size_t pos, size_t len) {
    valor.erase(pos, len);
    return *this;
}

StringWrapper& StringWrapper::append(const std::string& str) {
    valor.append(str);
    return *this;
}

StringWrapper& StringWrapper::append(const char* s) {
    valor.append(s);
    return *this;
}

StringWrapper& StringWrapper::append(size_t n, char c) {
    valor.append(n, c);
    return *this;
}

void StringWrapper::push_back(char c) {
    valor.push_back(c);
}

void StringWrapper::pop_back() {
    valor.pop_back();
}

StringWrapper& StringWrapper::replace(size_t pos,
                                         size_t len,
                                         const std::string& str) {
    valor.replace(pos, len, str);
```



```
        return *this;
    }

void StringWrapper::swap(StringWrapper& other) {
    valor.swap(other.valor);
}

// ===== OPERADORES DE CONCATENACIÓN =====
StringWrapper& StringWrapper::operator+=(const StringWrapper& other) {
    valor += other.valor;
    return *this;
}

StringWrapper& StringWrapper::operator+=(const std::string& str) {
    valor += str;
    return *this;
}

StringWrapper& StringWrapper::operator+=(const char* s) {
    valor += s;
    return *this;
}

StringWrapper& StringWrapper::operator+=(char c) {
    valor += c;
    return *this;
}

// ===== SERIALIZACIÓN JSON =====
json StringWrapper::toJson() const {
    return json(valor);
}

void StringWrapper::fromJson(const json& j) {
    valor = j.get<std::string>();
}

// ===== OPERADORES DE STREAM =====
// Nota: Las funciones amigas no llevan el prefijo StringWrapper:::
std::ostream& operator<<(std::ostream& os, const StringWrapper& sw) {
    os << sw.valor;
    return os;
}

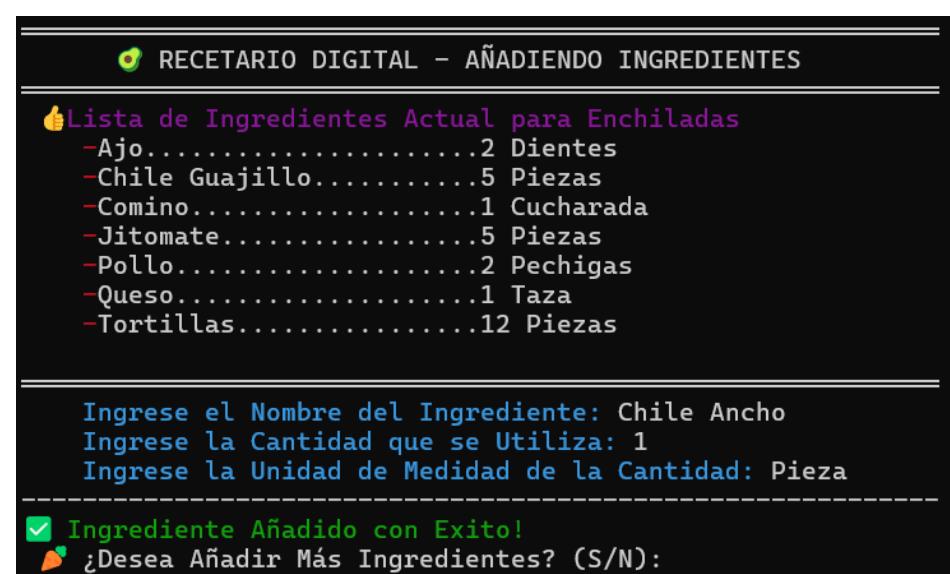
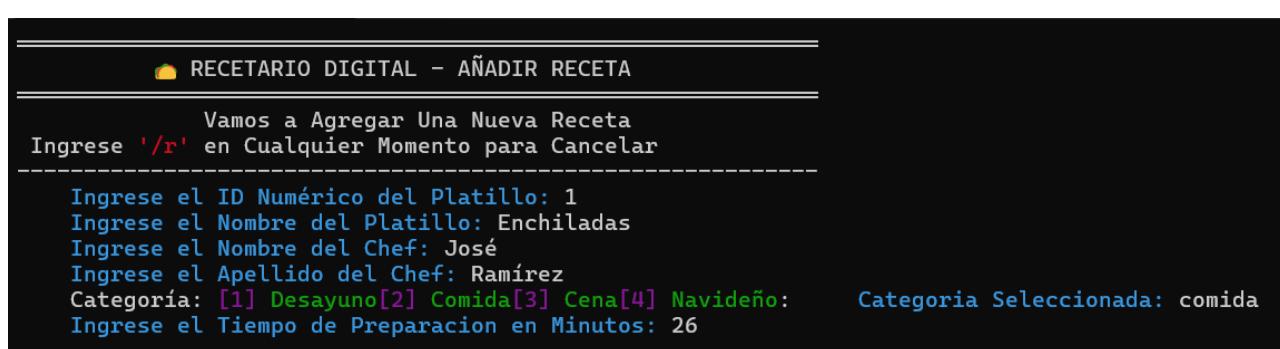
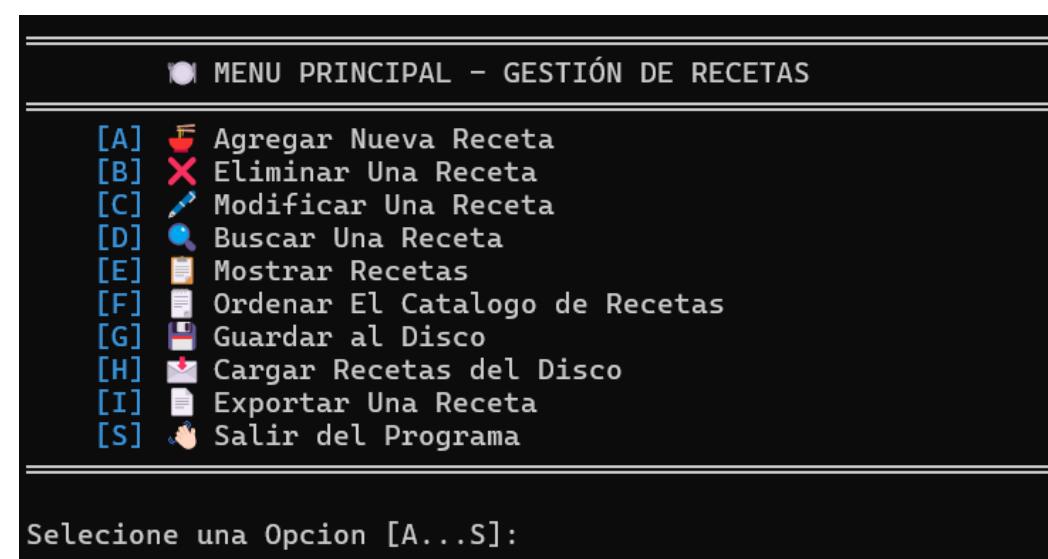
std::istream& operator>>(std::istream& is, StringWrapper& sw) {
```



```
    is >> sw.valor;
    return is;
}

std::istream& getline(std::istream& is, StringWrapper& sw) {
    return std::getline(is, sw.valor);
}

std::istream& getline(std::istream& is, StringWrapper& sw, char delim) {
    return std::getline(is, sw.valor, delim);
}
```



Pantalla 0

Primera pantalla del programa, muestra una breve animación de carga con datos esenciales y la fecha de ejecución

Pantalla 1

Menú principal con las opciones centrales del programa.

-Si se ingresa una opción que no esté en el programa:

pantalla 1.1.1

- Opción [A]: *pantalla 2*
- Opción [B]: *pantalla 3*
- Opción [C]: *pantalla 4*
- Opción [D]: *pantalla 5*
- Opción [E]: *pantalla 6*
- Opción [F]: *pantalla 7*
- Opción [G]: *pantalla 8*
- Opción [H]: *pantalla 9*
- Opción [I]: *pantalla 10*
- Opción [S]: *pantalla 11*

Pantalla 2.1

Pantalla para agregar una nueva receta pidiendo los datos

- Si se ingresa un ID no numérico: *pantalla 1.1.2*
- Si se ingresa un ID repetido: *pantalla ID repetido*
- Si se ingresa una cadena Vacía: *pantalla 1.1.3*
- Si se ingresa '/r': *pantalla 1.1.4*

Pantalla 2.2

Submenú de para añadir cuantos ingredientes sean necesarios

- Si se ingresa un espacio vacío: *pantalla 1.1.3*
- Si se ingresa un ingrediente repetido: *entrada inválida*
 - Si se confirma: *pantalla 2.2*
 - Si se niega: *pantalla 2.3*



RECETARIO DIGITAL - AÑADIR RECETA

Vamos a Agregar Una Nueva Receta
Ingrese 'r' en Cualquier Momento para Cancelar

Ingrese el ID Numérico del Platillo: 1
Ingrese el Nombre del Platillo: Enchiladas
Ingrese el Nombre del Chef: José
Ingrese el Apellido del Chef: Ramírez
Categoría: [1] Desayuno [2] Comida [3] Cena [4] Navideño Categoria Seleccionada: comida
Ingresar el Tiempo de Preparación en Minutos: 26
Inserte el Procedimiento Paso a Paso
Ingresar 'fin' para terminar.

1. Hervir los chiles y jitomates para suavizarlos
2. Licuarlos con especias, ajo, cebolla y caldo
3. Colar la salsa
4. Freir ligeramente las tortillas con aceite y sumergirlas en la salsa
5. Rellenas las tortillas con pollo deshebrado y enrollarlas
6. Servir y decorar con crema, queso rallado, cebolla picada y aguacate

!Receta Agregada Con Exito!

? ¿Desea Ingresar Una Nueva Receta? (S/N):

RECETARIO DIGITAL - ELIMINAR UNA RECETA

¡Receta Encontrada!
ID de la Receta: 1
Fecha de Creación: 19/1/2023
Nombre de la Receta: Galletas de Avena
Autor: Diego García
Categoría: Comida
Tiempo: 13 minutos.

● Ingredientes:
-Salsa BBQ.....1.5 Taza
-Chile.....1.5 Paquete
-Vinagre.....1 Cucharada
-Caldo de Res.....1 Cucharada
-Pollo.....400 Gramos
-Albahaca.....100 Piezas

■ Procedimiento:
1. Precalentar los ingredientes hasta que espese
2. Mezclar la salsa y servir caliente
3. Mezclar el arroz hasta que estén dorados
4. Reposar la pasta y reservar
5. Sazonar las verduras hasta integrar completamente
6. Sazonar los ingredientes hasta que estén dorados

? Esta Seguro Que Desea Eliminarla? (S/N): N
Eliminación Cancelada.
¿Realizar Otra Eliminación? (S/N): |

RECETARIO DIGITAL - ELIMINAR UNA RECETA

¡Receta Encontrada!
ID de la Receta: 1
Fecha de Creación: 19/1/2023
Nombre de la Receta: Galletas de Avena
Autor: Diego García
Categoría: Comida
Tiempo: 13 minutos.

● Ingredientes:
-Salsa BBQ.....1.5 Taza
-Chile.....1.5 Paquete
-Vinagre.....1 Cucharada
-Caldo de Res.....1 Cucharada
-Pollo.....400 Gramos
-Albahaca.....100 Piezas

■ Procedimiento:
1. Precalentar los ingredientes hasta que espese
2. Mezclar la salsa y servir caliente
3. Mezclar el arroz hasta que estén dorados
4. Reposar la pasta y reservar
5. Sazonar las verduras hasta integrar completamente
6. Sazonar los ingredientes hasta que estén dorados

? Esta Seguro Que Desea Eliminarla? (S/N): S
¡Receta Eliminada Con Exito!

? Realizar Otra Eliminación? (S/N): |

RECETARIO DIGITAL - EDITAR RECETA

Mediante Que Parámetro Quiere Buscar Su Receta:
[A] ID
[B] Nombre de la Receta
[C] Autor
[D] Categoría (Primera Coincidencia)
[R] Regresar
Ingresar una Opción:

Pantalla 2.3

Agregar los pasos de la receta, cuantos sean necesarios

- Si se ingresa un espacio vacío: **pantalla 1.1.3**
- Si se quiere agregar otra receta: **pantalla 2.1**

Pantalla 3.1

Eliminación rechaza, preguntamos si quiere eliminar otra receta

- Si se confirma **pantalla 3**
- Si se rechaza: **pantalla 1**

Pantalla 3.2

Eliminación confirmada, preguntamos si quiere eliminar otra receta

- Si se confirma **pantalla 3**
- Si se rechaza: **pantalla 1**

Pantalla 4

Pantalla para modificar una receta:

- Si no hay recetas registradas: **pantalla faltan recetas**
- Para la búsqueda: **pantalla búsqueda**



RECETARIO DIGITAL - EDITAR RECETA

ID de la Receta: 2
Fecha de Creación : 19/11/2025
Nombre de la Receta : Omelette de Queso
Autor : José Martínez
Categoría : Navideño
Tiempo : 33 minutos.

Ingrediente :
- Crema.....1 Paquete
- Salsa de Soja.....4 Taza
- Zanahoria.....50 Cucharada
- Orégano.....50 Cucharadita
- Tomate.....1 Dientes

Procedimiento :
1. Cortar la pasta y servir caliente
2. Batir el arroz hasta que estén dorados
3. Hervir la salsa hasta integrar completamente
4. Picar los ingredientes a 180°C por 25 minutos

Seleccione el Apartado a Modificar:

[A] Id
[B] Nombre de la Receta
[C] Autor
[D] Categoría
[E] Tiempo de Preparación
[F] Procedimiento
[G] Lista de Ingredientes
[R] Regresar

Seleccione una opción:

[A] Id
[B] Nombre de la Receta
[C] Autor
[D] Categoría
[E] Tiempo de Preparación
[F] Procedimiento
[G] Lista de Ingredientes
[R] Regresar

Seleccione una opción: A
Ingrese el nuevo ID: 10

[A] Id
[B] Nombre de la Receta
[C] Autor
[D] Categoría
[E] Tiempo de Preparación
[F] Procedimiento
[G] Lista de Ingredientes
[R] Regresar

Seleccione una opción: A
Ingrese el nuevo ID: 1
Modificación Realizada con Éxito

¿Desea Realizar Otra Modificación? (S/N):

RECETARIO DIGITAL - MODIFICAR LISTA DE PASOS

1. Precalentar la masa y servir caliente
2. Batir la masa hasta integrar completamente
3. Mezclar la pasta hasta integrar completamente
4. Hornear la pasta y reservar
5. Picar la masa hasta que estén dorados
6. Hervir la masa a fuego medio

[A] Agregar Pasos
[B] Eliminar Pasos
[C] Modificar Pasos
[R] Regresar

Ingrese una Opción:

Pantalla 4.1

Pantalla para elegir que apartado modificar:

- Si se ingresa una opción inválida: **pantalla entrada inválida**

-Opciones {A, B, C, D, F}: **pantalla 4.1.1**

- Opción [F]: **pantalla 4.1.2**
- Opción [G]: **pantalla 4.1.3**

Pantalla 4.1.1

Modificar atributos:

- (En caso de modificar el ID) Si se ingresa un ID repetido: **pantalla ID repetido**
- Si se ingresa un dato válido: 4.1.1.1

Pantalla 4.1.1.1

Confirma la Modificación:

- Si se confirma: **pantalla 4**
- Si se niega: **pantalla 1**

Pantalla 4.1.2

Modificar procedimiento:

- Si ingresa una opción inválida: **pantalla entrada inválida**

- Opción [A]: **pantalla 4.1.2.1**
- Opción [B]: **pantalla 4.1.2.2**
- Opción [C]: **pantalla 4.1.2.3**
- Opción [R]: **pantalla 4.1**



RECETARIO DIGITAL - AÑADIENDO PASOS

Lista de Pasos Actual para Tostadas Integrales de Aguacate

1. Precalentar la masa y servir caliente
2. Batir la masa hasta integrar completamente
3. Mezclar la pasta hasta integrar completamente
4. Hornear la pasta y reservar
5. Picar la masa hasta que estén dorados
6. Hervir la masa a fuego medio

Ingrese '/r' para Cancelar
Ingrese el Nuevo Paso Para La Receta: Enplatar con aditivos de selección

Paso Añadido con Exito!
X ¿Desea Añadir Más Pasos? (S/N):

Pantalla 4.1.2.1

Agregar Paso:

-Si ingresa una cadena vacía: **pantalla entrada inválida**

-Si se confirma: **pantalla 4.1.2.1**

-Si se niega: **pantalla 4.1.2**

RECETARIO DIGITAL - ELIMINAR PASOS

Lista de Pasos Actual para Tostadas Integrales de Aguacate

1. Precalentar la masa y servir caliente
2. Batir la masa hasta integrar completamente
3. Mezclar la pasta hasta integrar completamente
4. Hornear la pasta y reservar
5. Picar la masa hasta que estén dorados
6. Hervir la masa a fuego medio
7. Enplatar con aditivos de selección

Ingrese 'fin' para regresar
Ingrese el Número del Paso a Eliminar: 6
¿Esta Seguro que Desea Eliminar Este Ingrediente? (S/N): S
X ¡Paso Eliminado Con Exito!
X ¿Desea Eliminar Más Ingredientes? (S/N):

Pantalla 4.1.2.2

Eliminar Paso:

-Si ingresa una cadena vacía: **pantalla entrada inválida**

-Si se confirma: **pantalla 4.1.2.2**

-Si se niega o ingresa 'fin': **pantalla 4.1.2**

RECETARIO DIGITAL - MODIFICAR LISTA DE INGREDIENTES

-Pimienta.....	4 Dientes
-Zanahoria.....	300 Mililitros
-Arroz.....	50 Hoja
-Carne de Res.....	0.5 Hoja
-Cilantro.....	100 Cucharadita
-Vinagre.....	0.25 Cucharada
-Caldo de Res.....	200 Gramos
-Nuez.....	250 Piezas
-Chocolate.....	2 Dientes

[A] Agregar Ingredientes
[B] Eliminar Ingredientes
[C] Modificar Ingredientes
[R] Regresar
Ingrese una Opción:

Pantalla 4.1.3

Pantalla para editar la lista de Ingredientes:

-Si ingresa una cadena vacía: **pantalla entrada inválida**

-Opción [A]: **pantalla 2.2**

-Opción [B]: **pantalla 4.1.3.1**

-Opción [C]: **pantalla 4.1.3.2**

- Opción [R]: **pantalla 4.1**

RECETARIO DIGITAL - ELIMINAR INGREDIENTES

Lista de Ingredientes Actual para Tostadas de Aguacate

-Pimienta.....	4 Dientes
-Zanahoria.....	300 Mililitros
-Arroz.....	50 Hoja
-Carne de Res.....	0.5 Hoja
-Cilantro.....	100 Cucharadita
-Vinagre.....	0.25 Cucharada
-Caldo de Res.....	200 Gramos
-Nuez.....	250 Piezas
-Chocolate.....	2 Dientes

Ingrese 'fin' para regresar
Ingrese el Nombre del Ingrediente a Eliminar: Pimienta
¿Esta Seguro que Desea Eliminar Este Ingrediente? (S/N): S
X ¡Ingrediente Eliminado Con Exito!
X ¿Desea Eliminar Más Ingredientes? (S/N):

Pantalla 4.1.3.1

Pantalla para eliminar ingredientes

-Si ingresa una cadena vacía: **pantalla opción inválida**

-Si se ingresa un nombre no listado: **pantalla entrada inválida**

- Si se confirma: **pantalla 4.1.3.1**

-Si se niega: **pantalla 4.1.3**



RECETARIO DIGITAL - BUSCAR RECETA

Mediante Que Parámetro Quiere Buscar Su Receta:

- [A] ID
- [B] Nombre de la Receta
- [C] Autor
- [D] Categoría (Primera Coincidencia)
- [R] Regresar

Ingrese una Opcion:

RECETARIO DIGITAL - BUSCAR RECETA

ID de la Receta: 1
Fecha de Creación : 19/1/2023
Nombre de la Receta : Galletas de Avena
Autor : Diego Garcia
Categoría : Comida
Tiempo : 13 minutos.

Ingrediente :

- Salsa BBQ.....1.5 Taza
- Chile.....1.5 Paquete
- Vinagre.....1 Cucharada
- Caldo de Res.....1 Cucharada
- Pollo.....400 Gramos
- Albahaca.....100 Piezas

Procedimiento :

1. Precalentar los ingredientes hasta que espese
2. Mezclar la salsa y servir caliente
3. Mezclar el arroz hasta que estén dorados
4. Reposar la pasta y reservar
5. Sazonar las verduras hasta integrar completamente
6. Sazonar los ingredientes hasta que estén dorados

¿Desea Buscar Otra Receta (S/N)?:

RECETARIO DIGITAL - MOSTRAR RECETAS

Mostrar Todas las Recetas o Filtradas

[A] Mostrar Todas las Recetas
[B] Filtrar Por Categoría
[R] Regresar

Seleccione Una Opción:

TODAS LAS RECETAS

ID de la Receta: 1
Fecha de Creación : 19/1/2023
Nombre de la Receta : Galletas de Avena
Autor : Diego Garcia
Categoría : Comida
Tiempo : 13 minutos.

Ingrediente :

- Salsa BBQ.....1.5 Taza
- Chile.....1.5 Paquete
- Vinagre.....1 Cucharada
- Caldo de Res.....1 Cucharada
- Pollo.....400 Gramos
- Albahaca.....100 Piezas

Procedimiento :

1. Precalentar los ingredientes hasta que espese
2. Mezclar la salsa y servir caliente
3. Mezclar el arroz hasta que estén dorados
4. Reposar la pasta y reservar
5. Sazonar las verduras hasta integrar completamente
6. Sazonar los ingredientes hasta que estén dorados

ID de la Receta: 2
Fecha de Creación : 19/11/2025
Nombre de la Receta : Omelette de Queso
Autor : José Martinez

Pantalla 5

Pantalla de búsqueda de recetas

- Si ingresa una cadena vacía: **pantalla opción inválida**
- Si no hay recetas registradas. **Pantalla falta recetas**
 - Va a: **pantalla búqueda**

Pantalla 5.1

Pantalla que muestra las características y el registro completo de la receta encontrada

- Si se confirma: **pantalla 5**
- Si se niega: **pantalla 1**

Pantalla 6

Pantalla para mostrar listado de ingredientes

- Si no hay recetas registradas: **pantalla faltan recetas**
- Si ingresa una opción inválida: **pantalla opción inválida**
 - Opción [A]: **pantalla 5.1**
 - Opción [B]: **pantalla 5.2**
 - Opción [R]: **pantalla 1**

Pantalla 6.1

Listado informativo de todas las recetas

- Regresa a: **pantalla 5**



```
RECETARIO DIGITAL - MOSTRAR RECETAS
Mostrar Todas las Recetas o Filtradas
[A] Mostrar Todas las Recetas
[B] Filtrar Por Categoría
[R] Regresar
Seleccione Una Opción: B
Ingrese / para Cancelar
Seleccione una Categoría      Categoría: [1] Desayuno[2] Comida[3] Cena[4] Navideño: 3
```

```
TODAS RECETAS TIPO: cena
ID de la Receta: 5
Fecha de Creación : 10/8/2024
Nombre de la Receta : Enchiladas Verdes
Autor : Lucía Salazar
Categoría : Cena
Tiempo : 103 minutos.

Ingrediente :
-Cacao..... 200 Piezas
-Espinaca..... 50 Piezas
-Salsa BBQ..... 200 Hoja
-Crema..... 3 Cucharada
-Perejil..... 300 Gramos
-Nata..... 150 Gramos
-Caldo de Res..... 400 Cucharada
-Papa..... 1.5 Cucharadita
-Pan..... 300 Gramos
-Pasta..... 100 Piezas

Procedimiento :
1. Precalentar el pollo y reservar
2. Hornear la pasta y reservar
3. Hervir las verduras hasta integrar completamente
4. Sazonar la mezcla a fuego medio
5. Batir la pasta hasta integrar completamente
6. Cocer la carne hasta integrar completamente
7. Batir el aderezo hasta que esté al dente
```

```
RECETARIO DIGITAL - ORDENAR RECETAS
Seleccione Sobre Qué Categoría Quiere Ordenar Sus Recetas:
[A] Ordenar por ID
[B] Ordenar por Nombre de la Receta
[C] Ordenar por Nombre del Autor
[D] Ordenar por Categoría
[E] Ordenar por Tiempo de Preparación
[F] Ordenar por Fecha de Creación
[S] Salir
Seleccione una Opción:
```

```
RECETARIO DIGITAL - ORDENAR RECETAS
Seleccione Sobre Qué Categoría Quiere Ordenar Sus Recetas:
[A] Ordenar por ID
[B] Ordenar por Nombre de la Receta
[C] Ordenar por Nombre del Autor
[D] Ordenar por Categoría
[E] Ordenar por Tiempo de Preparación
[F] Ordenar por Fecha de Creación
[S] Salir
Seleccione una Opción: B
¡Datos Ordenados Con Éxito!
Presione una tecla para continuar . . .
```

Pantalla 6.2

Pantalla para preguntar qué categoría quiere mostrar

-Si se ingresa una opción inválida: **pantalla opción inválida**

-Al seleccionar una categoría: **pantalla 5.2.1**

Pantalla 6.2.1

Lista informativo de todas las recetas registradas con la categoría puesta

-Regresa a: **pantalla 1**

Pantalla 7

Opciones para el ordenamiento de la lista de recetas

-Si hay menos de dos recetas: **pantalla faltan recetas**

-selección de una categoría de ordenamiento: **pantalla 6.1**

- Opción [R]: **pantalla 1**

Pantalla 7.1

Mensaje de ordenamiento satisfactorio

-Regresa a: **pantalla 1**



```
RECETARIO DIGITAL - GUARDAR RECETAS EN EL DISCO
Ingrese '/r' para Cancelar la Operación
Ingrese el Nombre con el Cual Quiere Guardar al Disco (Se Guardará en Formato .Json): |
```

Pantalla 8

Pantalla para seleccionar el nombre del archivo .json que se guardará

- Si no hay recetas: *pantalla faltan recetas*
- Si se ingresa '/r': *pantalla 1*
- si se ingresa un nombre: *pantalla 7.1*

```
RECETARIO DIGITAL - GUARDAR RECETAS EN EL DISCO
Ingrese '/r' para Cancelar la Operación
Ingrese el Nombre con el Cual Quiere Guardar al Disco (Se Guardará en Formato .Json): ejemploDataset.json
✓ jDatos Guardados Con Exito!
Regresando...
Presione una tecla para continuar . . .
```

Pantalla 8.1

Se avisa si el archivo se creó y guardó exitosamente

- Regresa a: *pantalla 1*

```
RECETARIO DIGITAL - LEER RECETAS DEL ALMACENAMIENTO
ADVERTENCIA: Si se leen datos con registros en la aplicación
estos serán sobreescritos
Ingrese '/r' para cancelar
Ingrese el Nombre de la Base de Datos .json (sin la extensión):|
```

Pantalla 9

Pantalla para cargar un archivo al programa

- Ingresa '/r': *pantalla 1*
- Ingresa un nombre no existente: *pantalla 8.1*
- Ingresa una dataset válido: *pantalla 8.2*

```
RECETARIO DIGITAL - LEER RECETAS DEL ALMACENAMIENTO
ADVERTENCIA: Si se leen datos con registros en la aplicación
estos serán sobreescritos
Ingrese '/r' para cancelar
Ingrese el Nombre de la Base de Datos .json (sin la extensión):ejemploInvalido
No se encontró o no se pudo abrir el archivo.
Regresando...
Presione una tecla para continuar . . .
```

Pantalla 9.1

Avisa que el archivo no existe o no se pudo abrir.

- Regresa a: *pantalla 1*



RECETARIO DIGITAL - EXPORTAR UNA RECETA.

Mediante Que Parámetro Quiere Buscar Su Receta:

- [A] ID
- [B] Nombre de la Receta
- [C] Autor
- [D] Categoría (Primera Coincidencia)
- [R] Regresar

Ingrese una Opcion:

RECETARIO DIGITAL - EXPORTAR UNA RECETA.

ID de la Receta: 2
Fecha de Creación : 19/11/2025
Nombre de la Receta : Omelette de Queso
Autor : José Martínez
Categoría : Navideño
Tiempo : 33 minutos.

Ingrediente :

- Crema.....1 Paquete
- Salsa de Soja.....4 Taza
- Zanahoria.....50 Cucharada
- Orégano.....50 Cucharadita
- Tomate.....1 Dientes

Procedimiento :

1. Cortar la pasta y servir caliente
2. Batir el arroz hasta que estén dorados
3. Hervir la salsa hasta integrar completamente
4. Picar los ingredientes a 180°C por 25 minutos

Ingrese el Nombre del Archivo Destino (con extensión): ejemploExportacion.txt
!Exportación Realizada con Éxito!

¿Desea Realizar Otra Exportación? (S/N):

RECETARIO DIGITAL - EXPORTAR UNA RECETA.

ID de la Receta: 2
Fecha de Creación : 19/11/2025
Nombre de la Receta : Omelette de Queso
Autor : José Martínez
Categoría : Navideño
Tiempo : 33 minutos.

Ingrediente :

- Crema.....1 Paquete
- Salsa de Soja.....4 Taza
- Zanahoria.....50 Cucharada
- Orégano.....50 Cucharadita
- Tomate.....1 Dientes

Procedimiento :

1. Cortar la pasta y servir caliente
2. Batir el arroz hasta que estén dorados
3. Hervir la salsa hasta integrar completamente
4. Picar los ingredientes a 180°C por 25 minutos

Ingrese el Nombre del Archivo Destino (con extensión): ejemploExportacion.txt
!Exportación Realizada con Éxito!

¿Desea Realizar Otra Exportación? (S/N):

Antes de proceder a la pantalla

Pantalla 10

Pantalla preliminar para la exportación de una receta en otro formato

-Se va a: **pantalla búsqueda**

-seguimos en: **pantalla 9.1**

Pantalla 10.1

Pantalla que muestra la receta que se quiere exportar

-Si se ingresa una cadena vacía: **pantalla cadena vacía**

- Al ingresar un nombre válido: **pantalla 9.2**

Pantalla 10.2

Pantalla que avisa que hubo una exportación exitosa

-Si se confirma: **pantalla 9**

- Si se niega: **pantalla 1**

Pantalla 11

Si hubo cambios: **pantalla 10.1**

-Si no hubo cambios: **pantalla 10.2**



```
Saliendo del Programa.  
Que Tenga un Lindo Día :D  
  
Presione una tecla para continuar . . .
```

Pantalla 11.2

Mensaje de despedida al usuario y finaliza el programa.

-Se sale del programa

```
[ERROR]  
La opcion ingresada es invalida.  
Intentelo nuevamente.  
  
Presione una tecla para continuar . . . |
```

Pantalla entrada inválida

Avisa al usuario que el dato ingresado no está entre las opciones

-Regresa a la pantalla que lo llamó

```
[ERROR]  
El ID ya esta registrado:  
ID de la Receta: 1  
Fecha de Creación : 19/1/2023  
Nombre de la Receta : Galletas de Avena  
Autor : Diego García  
Categoría : Comida  
Tiempo : 13 minutos.  
  
● Ingrediente :  
-Salsa BBQ.....1.5 Taza  
-Chile.....1.5 Paquete  
-Vinegar.....1 Cucharada  
-Caldo de Res.....1 Cucharada  
-Pollo.....400 Gramos  
-Albahaca.....100 Piezas  
  
■ Procedimiento :  
1. Precalentar los ingredientes hasta que espese  
2. Mezclar la salsa y servir caliente  
3. Mezclar el arroz hasta que estén dorados  
4. Reposar la pasta y reservar  
5. Sazonar las verduras hasta integrar completamente  
6. Sazonar los ingredientes hasta que estén dorados  
  
Intentelo Nuevamente  
  
Presione una tecla para continuar . . .
```

Pantalla ID repetido

Muestra al usuario la receta que posee el ID que quiere añadir o modificar, indicando que no puede utilizar aquel.

-Regresa a la pantalla que lo llamó

```
[ERROR]  
No Hay Recetas Registradas  
Ingrrese Recetas Antes  
  
Presione una tecla para continuar . . . |
```

Pantalla faltan recetas

Indica al usuario que la funcionalidad que quiere como editar, eliminar, etc., necesita recetas para funciones

-Regresa a la pantalla que lo llamó



```
Mediante Que Parámetro Quiere Buscar Su Receta:  
[A] ID  
[B] Nombre de la Receta  
[C] Autor  
[D] Categoría (Primera Coincidencia)  
[R] Regresar  
Ingrese una Opcion: |
```

```
Mediante Que Parámetro Quiere Buscar Su Receta:  
[A] ID  
[B] Nombre de la Receta  
[C] Autor  
[D] Categoría (Primera Coincidencia)  
[R] Regresar  
Ingrese una Opcion: A  
Ingrese el ID a Buscar:
```

```
RECETARIO DIGITAL - ELIMINAR UNA RECETA  
Receta No Encontrada  
¿Desea Intentar Otra Búsqueda? (S/N): |
```

```
Mediante Que Parámetro Quiere Buscar Su Receta:  
[A] ID  
[B] Nombre de la Receta  
[C] Autor  
[D] Categoría (Primera Coincidencia)  
[R] Regresar  
Ingrese una Opcion: B  
Ingrese el Nombre de la Receta a Buscar: Tostadas de Aguacate
```

```
Mediante Que Parámetro Quiere Buscar Su Receta:  
[A] ID  
[B] Nombre de la Receta  
[C] Autor  
[D] Categoría (Primera Coincidencia)  
[R] Regresar  
Ingrese una Opcion: D  
Ingrese la Categoría : Categoría: [1] Desayuno[2] Comida[3] Cena[4] Navideño: 3
```

Pantalla búsqueda

Pantalla intermedia entre muchas funciones que necesitan de ubicar una receta

-Opción [A]: *pantalla b1*

-Opción [B]: *pantalla b2*

-Opción [C]: *pantalla b3*

-Opción [R]: *regresa a pantalla que lo llamó*

Pantalla b1

Pide el ID para buscar

-Si se encuentra: *retorna a la función que lo llamó con la receta*

-Si no se encuentra; *pantalla b1.1*

Pantalla b1.1

Avisa que la receta no fue encontrada

-Si se confirma: *pantalla b1*

-Si se niega: *pantalla que lo llamó*

Pantalla b2

Pide el nombre de la receta a buscar

-Si se encuentra: *pantalla que lo llamó*

-Si no se encuentra: *pantalla b1.1*

Pantalla b3

Pide el identificador de categoría para buscar

-Si se encuentra: *pantalla que lo llamó*

-Si no se encuentra: *pantalla b1.1*



Conclusiones

La realización del proyecto de este proyecto representó una experiencia sumamente enriquecedora y formativa, en la que pude aplicar de manera integral los conocimientos teóricos y prácticos adquiridos sobre programación orientada a objetos, estructuras de datos, algoritmos recursivos y manejo de archivos en C++. Este trabajo no solo me permitió tener un poco más del dominio del lenguaje, sino también comprender de manera más profunda la importancia del diseño estructurado, la modularidad y la correcta administración de los recursos de memoria, aplicando lo visto en clase y más propuestas que me surgieron desde la lectura de las indicaciones.

Desde el inicio, el desarrollo del proyecto implicó analizar cuidadosamente los requerimientos y traducirlos en un modelo de clases coherente y funcional. La creación de las clases Recipe, Name, Date, Ingredient, etc., permitió aplicar de forma práctica los principios de encapsulación, abstracción y reutilización del código. Definir constructores, destructores, métodos de acceso, comparadores explícitos y sobrecargas de operadores me ayudó a entender cómo garantizar la integridad de los datos y mantener un flujo de información controlado entre las distintas partes del programa.

A lo largo del desarrollo también fue enriquecedor conocer como hacer el manejo de archivos en formato json, para lo cual utilicé una biblioteca externa, una fuente de herramientas más que me servirá en proyectos futuros.

Durante la fase de pruebas, al cargar y manipular al menos varias recetas con varios ingredientes cada una, logré constatar el correcto funcionamiento de los métodos implementados y la estabilidad general del programa. Este proceso me enseñó a planificar casos de prueba, detectar errores lógicos y realizar depuración sistemática, habilidades fundamentales en el desarrollo de software. También fue gratificante observar cómo la arquitectura orientada a objetos facilitó las modificaciones y ampliaciones, permitiendo agregar nuevas funciones sin alterar la lógica central.



En conjunto, este proyecto me permitió integrar conocimientos de distintas áreas: desde la lógica algorítmica y la recursión hasta el diseño orientado a objetos y el manejo de archivos. Más allá del resultado funcional, considero que lo más valioso fue el proceso de razonamiento y toma de decisiones técnicas que implicó cada etapa del desarrollo. Cada clase, método y algoritmo requirió justificar su propósito y su implementación, lo cual fortaleció mi criterio como programador y mi capacidad para abordar problemas de forma estructurada y eficiente.

El desarrollo de este programa me emocionó bastante, y su actualización a la versión final es aún más intrigante y emocionante.