# Implementación Menú Inteligente

Alejandro Velasco[1] and Martín León[1]

[1] Universidad San Francisco de Quito. Quito, Ecuador.
favelascoc@estud.usfq.edu.ec
[2] Universidad San Francisco de Quito. Quito, Ecuador.
mleon1@estud.usfq.edu.ec

**Abstract.** Se implementó un Menú interactivo (GUI), en Java 8, en el cual existe un grupo de ingredientes que pueden o no formar parte de un plato. El Menú tiene diferentes funcionalidades que permitne facilitar al cliente al momento de pedir su plato. Entre las más importantes se encuentran: conocer que ingredientes lleva un plato y conocer en qué platos se encuentra un ingrediente específico. Además se implementó una interfaz gráfica amigable para que el cliente pueda interactuar con el Menú de forma sencilla.

**Keywords:** Menu · Plato · Ingrediente.

## 1 Introducción

Se quiere implementar un menú inteligente para que un cliente pueda elegir con facilidad un plato a su conveniencia. Existe un grupo de platos e ingredientes que contiene el Menu. Cada plato contiene diferenes ingredientes. El cliente puede conocer los ingredientes dado el nombre del plato, o puede conocer los platos en cuales un ingrediente forma parte. Así mismo, el cliente tiene la opción de saber el (los) ingrediente que menos se usan en todo el menu, el (los) nombre del plato que más ingredientes lleva. Finalmente, el cliente puede observar un reporte ordenado de todos los ingredientes que lleva cada plato y su nombre.
Para poder implementar este menú inteligente, se trabajó con Java 8, debido a que este lenguaje es puramente enfocado a objetos. De esta forma se pudo ordenar cada objeto Ingrediente y Plato dentro de la clase controladora, Menu. Ademaś, se utilizó JavaFx para que el cliente pueda interactuar con el programa a traves de una interfaz gráfica (GUI). Para escribir el código del proyecto utilizó NetBeans IDE 8.2.

## 2 Materiales y Métodos

Para resolver el objetivo planteado fue necesario crear dos clases entidades y una clase controladora: Ingrediente, Plato y Menú respectivamente.
En la Figura 1 se muestra el diagrama UML de este sistema.

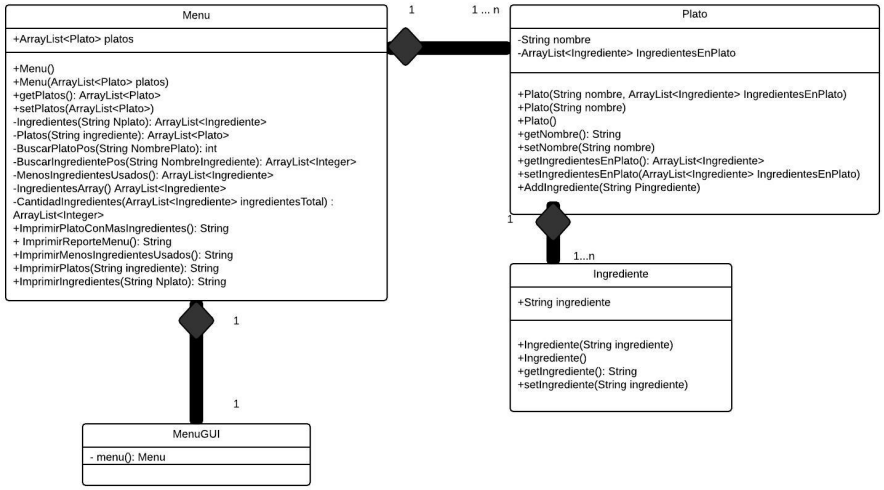


**Fig. 1.** UML

### 2.1 Clases Ingrediente y Plato

La clase Ingrediente tiene un único miembro dato: String Ingrediente, el cuál se refiere al nombre que identifica a cada ingrediente. Debido a que un ingrediente suelto (sin usar) no tiene mucho sentido, se creó una clase Plato que contiene varios ingredientes. Para que cumpla con la propiedad dinámica de un plato, la clase Plato tiene como miembro dato, además del nombre que lo identifica, un arreglo dinámico de platos (ArrayList<Plato> ingredientes).

### 2.2 Clase Menu

La clase controladora en este caso es la clase Menu. Como se mencionó anteriormente, un ingrediente suelto sin un plato no tiene sentido, por lo que la clase Menu, para contenter todos los grupos de platos e ingredientes, tuvo un solo miembro dato: ArrayList<Plato> platos.
La clase Menu es la controladora de todo el sistema entonces es aquí donde se implementan todas las funcionalidades mencionadas anteriormente.
A continuación se explica el algoritmo utilizado y el código escrito para cada caso.
Es importante mencionar que los códigos que serán analizados a continuacion no son los que el cliente podrá llamar. Los códigos que utiliza el cliente son los que empiezan con **Imprimir...()** mostrados en la Figura 1. Estos devuelven una secuencia de caracteres (String) que son mostrados en la interfaz gráfica. Por supuesto, cada una de estas funciones **Imprimir...()** utilizan las funciones presentadas a continuación.

### 2.3 Clase MenuGUI

El GUI creado con JavaFX toma el nombre de MenuGUI, en este, se encuentra como campo un objeto del tipo Menu en el cul se van a aadir las elecciones que tome el usuario mediante la interfaz grfica y a partir de ellas se van a mostrar las utilidades del programa, es decir, la resolución de los incisos planteados.

**Dado el nombre de un plato conocer sus ingredientes.** Esta funcionalidad se implementa de manera sencilla debido a que la clase se basa en un arreglo dinámico de Platos.
Para ello, lo primero que se realiza es buscar la posición en la cual se encuentra el plato buscado (Figura 2). Una vez que se tiene la posición dentro del arreglo, *i.e* el plato encontrado, se obtiene el arrego dinámico de Ingredientes del Plato correspondiente al plato deseado (Figura 3).

```
private int BuscarPlatoPos(String NombrePlato){
    int pos=-1;
    for(int i =0; i< platos.size();i++){
        if(NombrePlato.equals(platos.get(i).getNombre()))
            pos=i;
    }
    return pos;
}
```

**Fig. 2.** Búsqueda de Plato

```
private ArrayList<Ingrediente> Ingredientes(String Nplato){
    int index = BuscarPlatoPos(Nplato);
    if(index != -1){
        return platos.get(index).getIngredientesEnPlato();
    }
    return null;
}
```

**Fig. 3.** Ingredientes en Plato

**Dado un ingrediente conocer en cules platos l forma parte.** Debido a que la clase Menu contiene un arreglo de Plato, y a su vez Plato contiene un arreglo de Ingrediente; se puede extraer esta información en un arreglo bidimensional de Plato e Ingrediente (Tabla 1).

El algoritmo para conocer donde se encuentra un plato es sencillo si se abstrae esta idea: se realiza la búsqueda de las posiciones en las que un Ingrediente dado

**Table 1.** Arreglo Bidimensional Abstracto

| | | | | | |
|---|---|---|---|---|---|
| Plato a | Ingrediente 1a | Ingrediente 2a | Ingrediente 3a | ··· | Ingrediente na |
| Plato b | Ingrediente 1b | Ingrediente 2b | ··· | Ingrediente nb | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| Plato z | Ingrediente 1z | Ingrediente 2z | ··· | ··· | ··· |

se encuentra dentro de esta matriz abstracta (4). Por ejemplo, un Ingrediente 3 puede encontrarse en Plato a, Plato c; entonces las posiciones serían: 0 y 2. Aquellas posiciones representan las columnas de Tabla 1. Entonces se obtienen los objetos Plato que contienen dicho Ingrediente (Figura 5).

```
private ArrayList<Integer> BuscarIngredientePos(String NombreIngrediente){
    ArrayList<Integer> posiciones= new ArrayList<>();
    ArrayList<Ingrediente> pingredientes = new ArrayList<>();
    for(int i =0; i< platos.size();i++){
        pingredientes = platos.get(i).getIngredientesEnPlato();
        for(int j=0; j <pingredientes.size();j++)
            if(NombreIngrediente.equals(pingredientes.get(j).getIngrediente())){
                posiciones.add(i);
            }
    }
    return posiciones;
}
```

**Fig. 4.** Posiciones de un Ingrediente

```
private ArrayList<Plato> Platos(String ingrediente){
    ArrayList <Integer> posiciones = BuscarIngredientePos(ingrediente);//posiciones en las que se encuentra el plato
    ArrayList <Plato> PlatosConIngrediente = new ArrayList<>();

    for(int i =0; i < posiciones.size();i++)//añade el objeto ingrediente encontrado
        PlatosConIngrediente.add(platos.get(posiciones.get(i)));

    return PlatosConIngrediente;
}
```

**Fig. 5.** Platos que Contienen Ingrediente

**El (los) ingrediente que menos se usa.** Para conocer los ingredientes que menos se usa en todo el Menú, se utilizaron varias funciones de soporte.
La primera función creada sirve para conocer todos los ingredientes ingresados en el Menú hasta el momento, sin repetición, en forma de un arreglo dinámico (Figura 6).

```
private ArrayList<Ingrediente> IngredientesArray(){
    ArrayList<Ingrediente> ingredientesTotal = new ArrayList<>();
    for(int i =0; i< platos.size();i++){//por cada plato
        for(int j=0; j < platos.get(i).getIngredientesEnPlato().size();j++){//por cada ingrediente en el plato
            ingredientesTotal.add(platos.get(i).getIngredientesEnPlato().get(j));
            if(ingredientesTotal.size()==0)//para primer elemento
                ingredientesTotal.add(platos.get(i).getIngredientesEnPlato().get(j));
        }
    }
```

**Fig. 6.** Arreglo de Ingredientes

Otra función de apoyo creada sirve para conocer la cantidad de veces que un Ingrediente fue utilizado (Figura 7) Por ejemplo, un Ingrediente 5b se encuentra en Plato B y Plato z. Entonces el resultado de esta función sería como: $\{0, 1, 0, 0, \cdots, 0, 1\}$
Con estas dos funcionalidades de apoyo se puede conocer de cada objeto In-

```
private ArrayList<Integer> CantidadIngredientes(ArrayList<Ingrediente> ingredientesTotal){
    ArrayList<Integer> cantidadIngredientes = new ArrayList<>();
    String ingredienteTotalSTR;
    ArrayList<Ingrediente> IngredientesEnPlatoArray = new ArrayList<>();
    int count;
    for(int i =0; i < ingredientesTotal.size();i++){//para cada ingrediente de mi ingredientestotal
        ingredienteTotalSTR = ingredientesTotal.get(i).getIngrediente();//guarda nombre del ingrediente
        count=0;//inicializo contador
        for(int j =0; j < platos.size();j++){//busco en cada plato
            IngredientesEnPlatoArray = platos.get(j).getIngredientesEnPlato();
            for(int k=0; k<platos.get(j).getIngredientesEnPlato().size();k++){//en cada ingrediente del plato en el menu
                if(ingredienteTotalSTR.equals(IngredientesEnPlatoArray.get(k).getIngrediente())){//si es el mismo
                    count++;
                }
            }
        }
        cantidadIngredientes.add(count);//para cada ingrediente a buscar
    }
    return cantidadIngredientes;
}
```

**Fig. 7.** Cantidad de Ingredientes

grediente también la cantidad de veces que se encuentra en el Menú. Entonces es simple implementar una comparación para obtener el el/los ingredientes que menos se usan (Figura 8). Siguiendo al ejemplo, el Ingrediente 5b se usa en $0 + 1 + 0 + 0 + \cdots + 0 + 1 = 2$.

**El (los) nombre del plato que ms ingredientes lleva.** Como cada objeto Plato tiene un arreglo de objetos Ingrediente, entonces se puede conocer la cantidad de ingredientes que llevan utilizando la funcón $size()$ de $ArrayList < Ingrediente >$ (Figura 9).

**Un reporte ordenado de los ingredientes que lleva cada plato y su nombre.** La implementación de un reporte ordenado se muestra en la Figura 10. El algoritmo para la misma consiste en realizar iteraciones dentro del Menu para cada objeto Plato y a su vez iteraciones dentro de Plato para cada objeto Ingrediente. Es decir, se recorre la matriz abstracta con el fin de ordenar

```
private ArrayList<Ingrediente> MenosIngredientesUsados(){
    ArrayList<Ingrediente> totalIngredientes = IngredientesArray();//ver definicion de funcion
    ArrayList<Integer> cantidadIngredientes = CantidadIngredientes(totalIngredientes);
    ArrayList<Ingrediente> menorIngrediente = new ArrayList<>();
    int menor=Integer.MAX_VALUE;
    for(int i =0; i< cantidadIngredientes.size();i++){
        if(menor> cantidadIngredientes.get(i)){
            menor = cantidadIngredientes.get(i);
        }
    }
    for(int i =0; i< cantidadIngredientes.size();i++){
        if(menor == cantidadIngredientes.get(i)){//por arreglos paralelos
            menorIngrediente.add(totalIngredientes.get(i));
        }
    }
    return menorIngrediente;
}
```

**Fig. 8.** Cantidad de Ingredientes

```
public String ImprimirPlatoConMasIngredientes(){
    String str = "";
    int mayorTemp;
    int mayor= Integer.MIN_VALUE;
    for(int i =0; i< platos.size();i++){
        mayorTemp=platos.get(i).getIngredientesEnPlato().size();
        if(mayorTemp > mayor)
            mayor = mayorTemp;
    }
    str=str+"El/Los plato con mas ingredientes : ";
    for(int i =0; i< platos.size();i++){
        if(platos.get(i).getIngredientesEnPlato().size()== mayor)
            str = str+ platos.get(i).getNombre()+" ";
    }
    return str;
}
```

**Fig. 9.** Plato con más Ingredientes

de manera alfabética los platos y adems de ordenar de manera alfabética los ingredientes que integran cada plato. Tabla 1.

```
public String ImprimirReporteMenu(){
    String str = "";
    ArrayList<Ingrediente> ingredientesEnPlatoArray;
    for(int i =0; i< platos.size();i++){
        str=str+"Plato : "+platos.get(i).getNombre()+" tiene los ingredientes: ";
        ingredientesEnPlatoArray=platos.get(i).getIngredientesEnPlato();
        for(int j =0; j<ingredientesEnPlatoArray.size();j++)
            str = str + ingredientesEnPlatoArray.get(j).getIngrediente()+ ", ";
        str= str+"\n";
    }
    return str;
}
```

**Fig. 10.** Reporte Ordenado

### 2.4 Casos de Prueba

Dentro del Menu se colocaron los siguientes objetos Plato con sus respectivos objetos Ingredientes:

1. Plato Locro: Papa, Queso, y Leche.

2. Plato Sanduche: Queso, Aguacate, Pan y Lechuga.
3. Plato Cafe: Agua, Cafe y Azucar.
4. Plato Lasagna: Queso, Carne, Fideo y Pasta.
5. Plato Fanesca: Leche, Choclo y Mani.

**Dado el nombre de un plato conocer sus ingredientes.** En este caso se quiso conocer los ingredientes que contenía **Cafe**. El resultado esperado era que este plato tuviera: *Agua, Cafe* y *Azucar*.
El resultado fue el siguiente:

```
Plato : Cafe - tiene: Agua Cafe Azucar
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Fig. 11.** Resultado 1

Con ello se demuestra que sí se obtuvo el resultado esperado.

**Dado un ingrediente conocer en cules platos l forma parte.** Para el ingrediente **Queso**, esta función debería mostrar que se encuentran en los platos: *Locro, Sanduche* y *Lasagna*.
El resultado fue el siguiente:
Con ello se demuestra que sí se obtuvo el resultado esperado.

```
El ingrediente: Queso - se encuentra en : Locro Sanduche Lasagna
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Fig. 12.** Resultado 2

**El (los) ingrediente que menos se usa.** Llamando a esta función se esperaba que sean varios ingredientes que menos se usan en todo el Menú: *Papa, Aguacate, Pan, Lechuga, Agua, Cafe, Azucar, Carne, Fiedo, Pasta, Choclo* y *Mani*.
El resultado fue el siguiente:
Con ello se demuestra que sí se obtuvo el resultado esperado.

```
El (los) ingrediente que menos se usa : -Papa -Aguacate -Pan -Lechuga -Agua -Cafe -Azucar -Carne -Fideo -Pasta -Choclo -Mani -
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Fig. 13.** Resultado 3

**El (los) nombre del plato que ms ingredientes lleva.** Llamando a esta función se esperaba que los platos con mas ingredientes fuesen: *Sanduche* y *Lasagna*.
El resultado fue el siguiente:
Con ello se demuestra que sí se obtuvo el resultado esperado.

```
El/Los plato con mas ingredientes : -Sanduche -Lasagna -
```

**Fig. 14.** Resultado 4

**Un reporte ordenado de los ingredientes que lleva cada plato y su nombre.** El reporte ordenado para el menu fue:

```
Plato : Cafe tiene los ingredientes: Agua, Cafe, Azucar,
Plato : Fanesca tiene los ingredientes: Leche, Choclo, Mani,
Plato : Lasagna tiene los ingredientes: Queso, Carne, Fideo, Pasta,
Plato : Locro tiene los ingredientes: Papa, Queso, Leche,
Plato : Sanduche tiene los ingredientes: Queso, Aguacate, Pan, Lechuga,

BUILD SUCCESSFUL (total time: 0 seconds)
```

**Fig. 15.** Resultado 5

## 3 Conclusiones

En este proyecto se implementó un Menú que contiene varios ingredientes en cada plato. La clase controladora, Menu, maneja un arreglo dinámico de objetos Plato, que a su vez contiene un arreglo dinámico de objetos Ingrediente. Con ello, la clase Menu tiene 5 diferentes funcionalidades para que el cliente pueda aprovechar a su disposición mediante el uso de una interaz gráfica que es el vínculo entre el programa y el usuario.

## 4 Recomendaciones

A pesar de que se puede conocer "El (los) ingrediente que menos se usa", esta función puede ser implementada de mejor manera si se cambia todo el sistema. Se recomienda, para futuras implementaciones de este Menú interactivo, analizar y trabajar en diferentes maneras de hacer que el programa sea más amigable, en cuanto a código, en esta funcionalidad. El programa puede ser más útil aun si se incluye una lista de precios a cada plato, donde el usuario puede escoger el rango de precios que desea. También, se puede incluir una funcionalidad que, así como

se buscan qué platos contienen qué ingrediente, hace una lista de ingredientes que NO se quiere tener, esta sería muy útil para las personas con cierto tipo de alergia o dieta específica. Para facilitar su elección de plato.

## References

1. Deitel, P., Deitel, H. (n.d.). *Java How to Program.* Deitel.
2. Liang, D. (n.d.). *Introduction to Java Programming.* Pearson.