



Taller POO CookMaster

Por:

Diego Andrés Sanabria Ascanio

Isamar Katihuska Ramirez Peñaranda

Docente:

Christian David Jaimes Acevedo

Facultad de Ingenierías

Ingeniería de Software

Tecnológico de Antioquia - Institución Universitaria

Medellín, Colombia

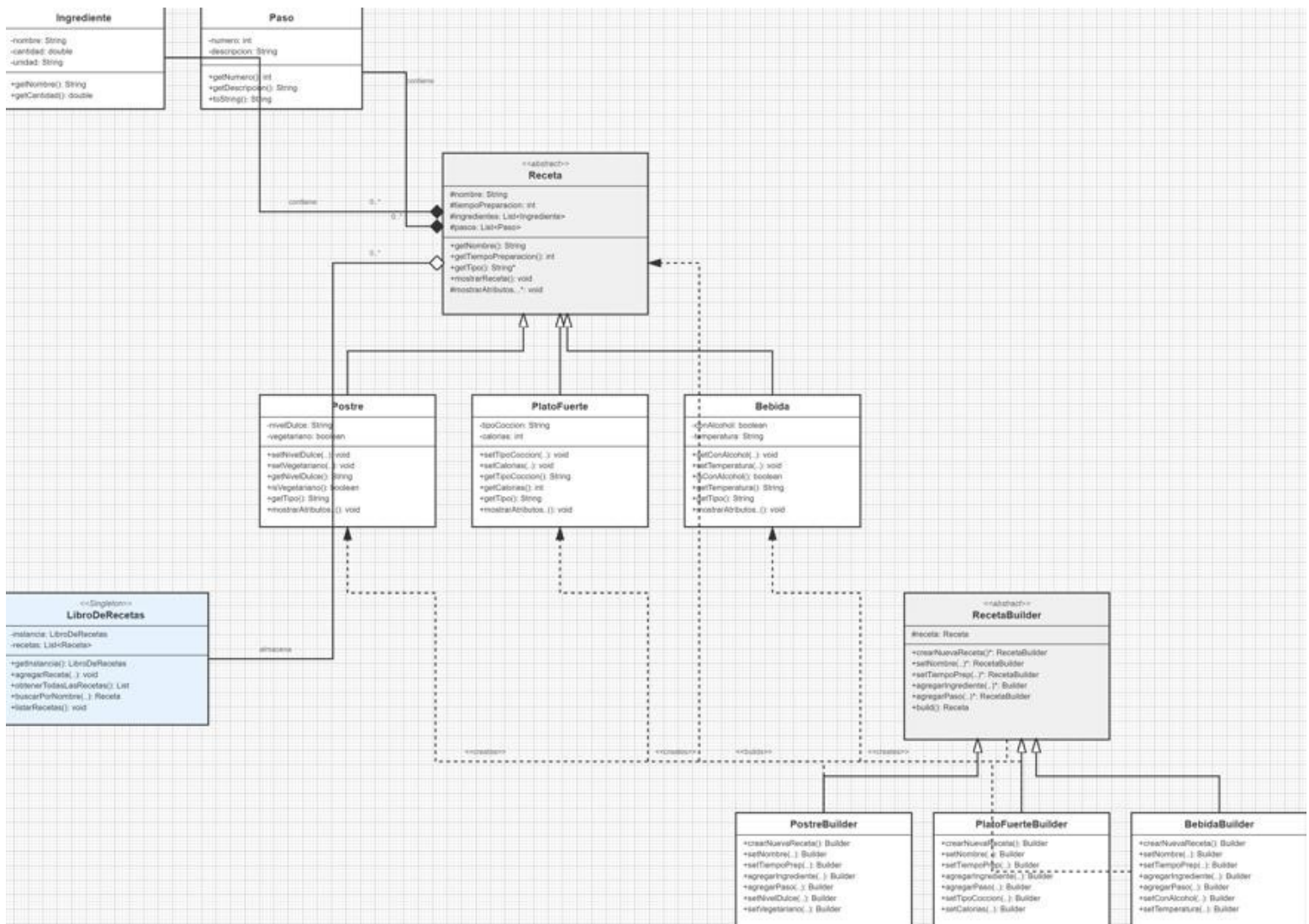
2025

Introducción

Para la realización de este trabajo, el primer paso fue elaborar un diagrama de clases que permitiera organizar de manera clara las ideas y comprender cómo debía estructurarse el sistema antes de comenzar a programar. Este diagrama sirvió como guía para identificar las clases necesarias, sus características y la forma en que se relacionan entre si gracias a este proceso inicial fue más sencillo planear el funcionamiento del sistema y evitar confusiones durante el desarrollo.

A partir de la información representada en el diagrama se construyó CookMaster, un sistema diseñado para gestionar recetas y todos los elementos que las componen, como sus ingredientes y pasos de preparación. El proyecto no solo permitió entender mejor cómo se conecta cada parte del programa, sino también la importancia de organizar y planificar antes de escribir código. En este informe se presenta tanto el diseño realizado en el diagrama como la explicación del funcionamiento del sistema que se desarrolló a partir de él.

Diagrama de clases



https://drive.google.com/file/d/1xIL2KkjU0gLiBTxuj6_hHJw5kZt3zmc/view?usp=sharing

Explicación del diagrama de clases

El diagrama representa un sistema orientado a objetos para la creación, gestión y almacenamiento de recetas, utilizando principios de herencia, composición y patrones de diseño como Builder y Singleton. A continuación, se explica cada parte del modelo:

1. Clases básicas: Ingrediente y Paso

- Ingrediente representa cada elemento necesario en una receta.

Contiene atributos como el nombre, la cantidad y la unidad de medida.

- Paso describe cada instrucción del procedimiento de preparación.

Contiene un número de orden y una descripción del paso.

Estas clases no heredan de otras, pero forman parte de la composición de las recetas.

2. Clase abstracta Receta

Receta es una clase abstracta que define lo común entre todas las recetas del sistema:

- nombre
- tiempo de preparación
- lista de ingredientes
- lista de pasos

Además, define comportamientos generales que comparten todas las recetas, como mostrar la receta, y métodos abstractos que las subclases deben implementar, como:

- `getTipo ()`
- `mostrarAtributosEspecificos ()`

Es la base para los diferentes tipos de recetas.

3. Subclases: Postre, Bebida y PlatoFuerte

Estas clases heredan de Receta y añaden atributos específicos:

- Postre
- nivel de dulzor
- si es o no vegetariano

Bebida

- sí contiene alcohol
- la temperatura a la que se sirve

PlatoFuerte

- tipo de cocción
- número de calorías

Cada una implementa sus propios métodos específicos y redefine el tipo de receta que representan.

4. Patrón Builder: construcción paso a paso

Para permitir crear recetas de manera flexible, el sistema utiliza el patrón de diseño Builder, representado por:

- RecetaBuilder (abstracta)
- Define los pasos necesarios para construir una receta.
- Permite agregar ingredientes, pasos y datos generales.

Y tres builders concretos:

- PostreBuilder
- BebidaBuilder
- PlatoFuerteBuilder

Cada builder está especializado en su tipo de receta. Así se asegura que:

- cada receta se construya correctamente,
- no falten atributos obligatorios,
- y se pueda configurar paso a paso (fluido).

5. LibroDeRecetas (Singleton)

Esta clase implementa el patrón Singleton, lo que significa que:

- solo puede existir una única instancia del libro de recetas,

- sirve como repositorio central de todas las recetas creadas.

Sus responsabilidades son:

- almacenar recetas
- listar todas las recetas
- buscar una receta por nombre
- permitir agregar nuevas

Esto garantiza un control único sobre el registro de recetas dentro del sistema.

6. Relaciones del sistema

El diagrama utiliza tres tipos principales de relaciones:

Herencia (flecha con triángulo)

Postre, Bebida y PlatoFuerte heredan de Receta.

Los builders concretos heredan de RecetaBuilder.

Composición (rombo sólido)

Receta está compuesta por ingredientes y pasos.

Si se elimina la receta, se eliminan sus ingredientes y pasos.

Agregación (rombo vacío)

LibroDeRecetas tiene una colección de recetas, pero las recetas pueden existir fuera del libro.

Este diagrama modela un sistema completo para gestionar recetas culinarias. Utiliza:

- Clases abstractas para definir estructura común
- Herencia para especializar tipos de recetas
- Composición para estructurar sus ingredientes y pasos
- El patrón Builder para construir recetas paso a paso

- El patrón Singleton para administrar un libro global de recetas

Es un diseño modular, escalable y alineado con buenas prácticas de programación orientada a objetos.

Explicación del Código – CookMaster

<https://github.com/Mort0/CookMaster.git>

El proyecto CookMaster es un sistema en Java diseñado para gestionar recetas de cocina de forma organizada y modular. El sistema permite crear, almacenar y consultar recetas de distintos tipos (Postres, Platos Fuertes y Bebidas), aplicando principios de programación orientada a objetos y patrones de diseño.

1. Estructura General del Sistema

El sistema está compuesto por las siguientes clases principales:

- **Ingrediente:** Representa un ingrediente con nombre, cantidad y unidad.
- **Paso:** Define un paso numerado del procedimiento de preparación.
- **Receta (abstracta):** Clase base que almacena nombre, tiempo de preparación, lista de ingredientes y pasos.
- **Postre, PlatoFuerte y Bebida:** Extienden a *Receta* y añaden atributos propios (ej. nivel de dulce, tipo de cocción, con/sin alcohol).

2. Patrones de Diseño Utilizados

Builder Pattern

Se utiliza para construir recetas de manera clara y flexible.

Cada tipo de receta tiene su propio *Builder* (PostreBuilder, PlatoFuerteBuilder, BebidaBuilder) que permite agregar ingredientes, pasos y atributos específicos mediante una sintaxis fluida.

Singleton Pattern

La clase LibroDeRecetas implementa este patrón para asegurar que solo exista una instancia centralizada que almacene todas las recetas.

Esto permite agregar, listar y buscar recetas desde un único punto.

3. Relaciones Entre Clases

- La clase Receta usa composición, ya que contiene listas de Ingredientes y Pasos.
- Las clases Postre, PlatoFuerte y Bebida heredan de Receta.
- Los Builders construyen los objetos de cada tipo de receta.
- LibroDeRecetas funciona como repositorio único (Singleton) para almacenar las recetas creadas.

4. Funcionamiento General

1. Se obtiene la instancia única del libro de recetas.
2. Se crea una receta utilizando el Builder correspondiente.
3. La receta se agrega al libro.
4. Se puede consultar o mostrar la receta cuando se necesite.

Este diseño permite que el sistema sea fácil de extender, mantener y reutilizar, aplicando principios sólidos de programación orientada a objetos.

Conclusiones

1. El desarrollo de CookMaster permitió aplicar de forma adecuada los patrones de diseño Builder y Singleton dentro de un sistema organizado y modular gracias al patrón Builder, la creación de recetas complejas se realiza de manera ordenada y flexible, mientras que el patrón Singleton asegura que exista un único libro de recetas, manteniendo la coherencia en el manejo de los datos.
2. La arquitectura del sistema demuestra una correcta separación de responsabilidades y un uso apropiado de la herencia para diferenciar los tipos de recetas, esto favorece la extensibilidad del proyecto y facilita su mantenimiento. En general, CookMaster evidencia buenas prácticas de programación y un diseño pensado para ser escalable y reutilizable.
3. El diagrama de clases representa de forma clara la estructura fundamental del sistema. La relación de herencia entre la clase abstracta Receta y sus subclases especializadas muestra cómo se organizan los diferentes tipos de recetas. Asimismo, las composiciones con Ingrediente y Paso dejan ver que cada receta está formada por elementos esenciales que dependen completamente de ella.
4. El diagrama también refleja la presencia de los patrones Builder y Singleton, mostrando cómo se gestionan la creación de objetos y la administración centralizada del libro de recetas. En conjunto, el diagrama proporciona una visión ordenada y comprensible del diseño del sistema, facilitando su análisis y futuras mejoras.