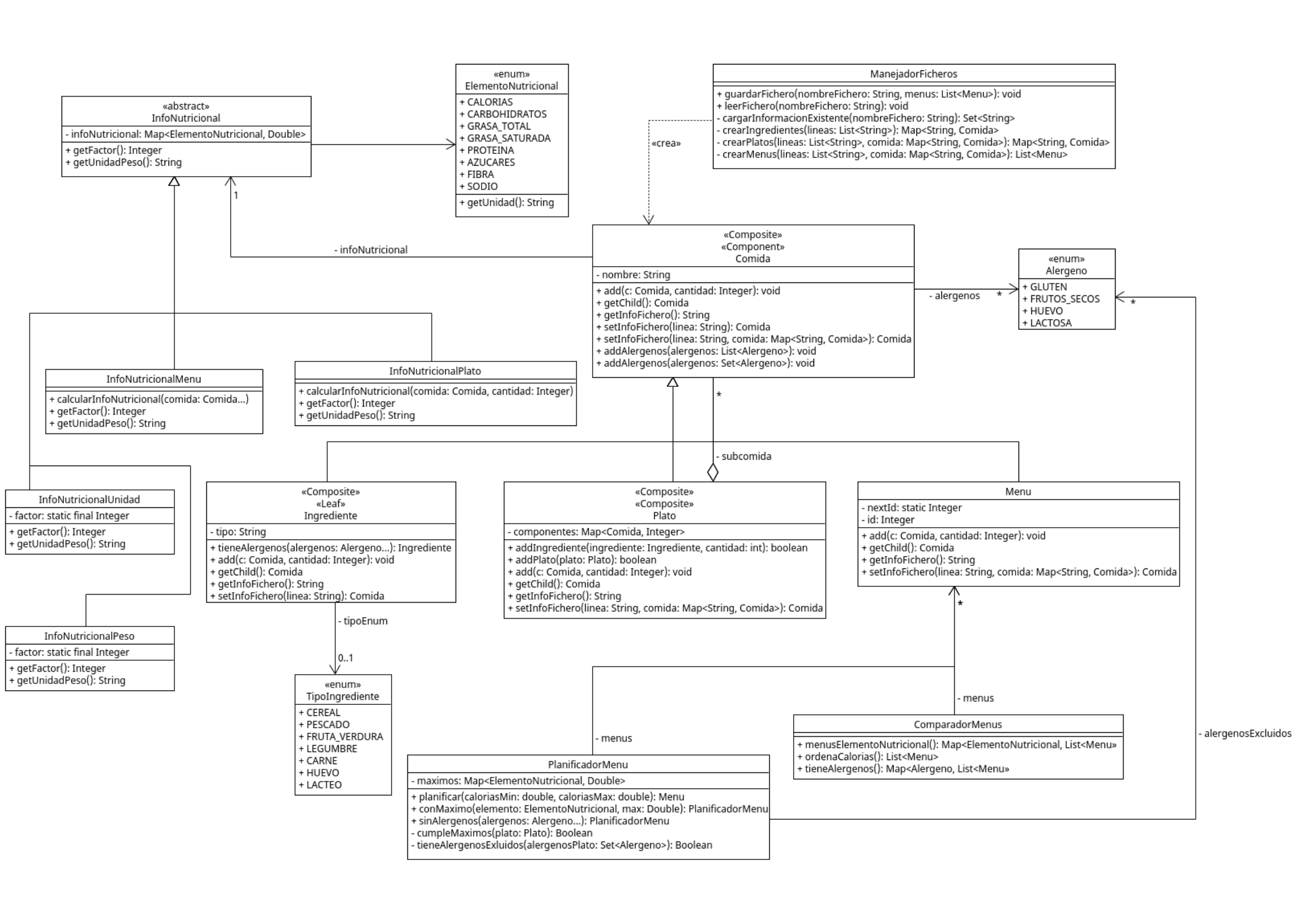
«abstract»
InfoNutricional
- infoNutricional: Map<ElementoNutricional, Double>
+ getFactor(): Integer
+ getUnidadPeso(): String
«enum»
ElementoNutricional
+ CALORIAS
+ CARBOHIDRATOS
+ GRASA_TOTAL
+ GRASA_SATURADA
+ PROTEINA
+ AZUCARES
+ FIBRA
+ SODIO
+ getUnidad(): String
ManejadorFicheros
+ guardarFichero(nombreFichero: String, menus: List<Menu>): void
+ leerFichero(nombreFichero: String): void
- cargarInformacionExistente(nombreFichero: String): Set<String>
- crearIngredientes(lineas: List<String>): Map<String, Comida>
- crearPlatos(lineas: List<String>, comida: Map<String, Comida>): Map<String, Comida>
- crearMenus(lineas: List<String>, comida: Map<String, Comida>): List<Menu>
«crea»
1
- infoNutricional
«Composite»
«Component»
Comida
- nombre: String
+ add(c: Comida, cantidad: Integer): void
+ getChild(): Comida
+ getInfoFichero(): String
+ setInfoFichero(linea: String): Comida
+ setInfoFichero(linea: String, comida: Map<String, Comida>): Comida
+ addAlergenos(alergenos: List<Alergeno>): void
+ addAlergenos(alergenos: Set<Alergeno>): void
- alergenos
*
«enum»
Alergeno
+ GLUTEN
+ FRUTOS_SECOS
+ HUEVO
+ LACTOSA
*
InfoNutricionalMenu
+ calcularInfoNutricional(comida: Comida...)
+ getFactor(): Integer
+ getUnidadPeso(): String
InfoNutricionalPlato
+ calcularInfoNutricional(comida: Comida, cantidad: Integer)
+ getFactor(): Integer
+ getUnidadPeso(): String
*
- subcomida
InfoNutricionalUnidad
- factor: static final Integer
+ getFactor(): Integer
+ getUnidadPeso(): String
«Composite»
«Leaf»
Ingrediente
- tipo: String
+ tieneAlergenos(alergenos: Alergeno...): Ingrediente
+ add(c: Comida, cantidad: Integer): void
+ getChild(): Comida
+ getInfoFichero(): String
+ setInfoFichero(linea: String): Comida
«Composite»
«Composite»
Plato
- componentes: Map<Comida, Integer>
+ addIngrediente(ingrediente: Ingrediente, cantidad: int): boolean
+ addPlato(plato: Plato): boolean
+ add(c: Comida, cantidad: Integer): void
+ getChild(): Comida
+ getInfoFichero(): String
+ setInfoFichero(linea: String, comida: Map<String, Comida>): Comida
Menu
- nextId: static Integer
- id: Integer
+ add(c: Comida, cantidad: Integer): void
+ getChild(): Comida
+ getInfoFichero(): String
+ setInfoFichero(linea: String, comida: Map<String, Comida>): Comida
*
InfoNutricionalPeso
- factor: static final Integer
+ getFactor(): Integer
+ getUnidadPeso(): String
- tipoEnum
0..1
«enum»
TipoIngrediente
+ CEREAL
+ PESCADO
+ FRUTA_VERDURA
+ LEGUMBRE
+ CARNE
+ HUEVO
+ LACTEO
- menus
- menus
ComparadorMenus
+ menusElementoNutricional(): Map<ElementoNutricional, List<Menu»
+ ordenaCalorias(): List<Menu>
+ tieneAlergenos(): Map<Alergeno, List<Menu»
- alergenosExcluidos
PlanificadorMenu
- maximos: Map<ElementoNutricional, Double>
+ planificar(caloriasMin: double, caloriasMax: double): Menu
+ conMaximo(elemento: ElementoNutricional, max: Double): PlanificadorMenu
+ sinAlergenos(alergenos: Alergeno...): PlanificadorMenu
- cumpleMaximos(plato: Plato): Boolean
- tieneAlergenosExluidos(alergenosPlato: Set<Alergeno>): Boolean

Para este diseño, hacemos uso del patrón Composite por la jerarquía que tienen los platos, usamos el método propio del patrón, getChild, para evitar menús en la jerarquía y para diferenciar entre platos e ingredientes a la hora de operar. Cada elemento, menú, ingrediente o plato, tiene una información nutricional asociada, aunque de distinto tipo, por ello, tenemos una clase padre abstracta que implementa la funcionalidad común entre los diversos tipos, a la hora de instanciar un elemento de comida, se instancia también su objeto de información nutricional correspondiente. En el ingrediente almacenamos el tipo enumerado pero también en formato de String, uno de los dos tiene que ser nulo y así controlamos los diversos tipos de ingredientes que hay.

Externo al funcionamiento del diseño tenemos manejadores, comparadores y planificadores, el manejador imprimirá los elementos del sistema en un fichero y también los creará si lee el fichero, de ahí la relación mostrada. El planificador, tiene una lista de platos según la cual devuelve una serie de menús compatibles con los requisitos que tiene, también un número arbitrario de alérgenos que no pueden estar en la solución presentada. Por útlimo, el comparador, simplemente almacena una lista de menús y devuelve colecciones ordenadas según el criterio solicitado.