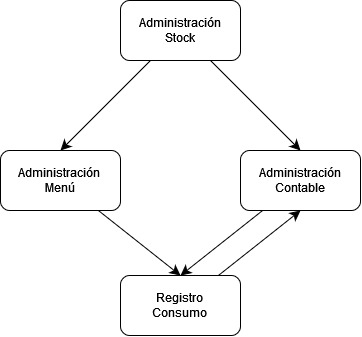
**Aplicación RestoApp**

Requerimientos Generales

Un restaurante que está avanzando con su transformación digital le pide a la compañía el desarrollo de una aplicación que les permita trabajar en los siguientes aspectos centrales de su negocio.

* **Control de inventario.** Consultar stock de un producto y bloquearlo para la venta si se agotó.
* **Contabilidad**. Llevar un estado actual del consumo de los clientes en el salón y permitir hacer balances entre lo comprado (inventario) y vendido (comidas).
* **Métricas**. Permiten medir el desempeño del negocio y ofrecer información para realizar promociones

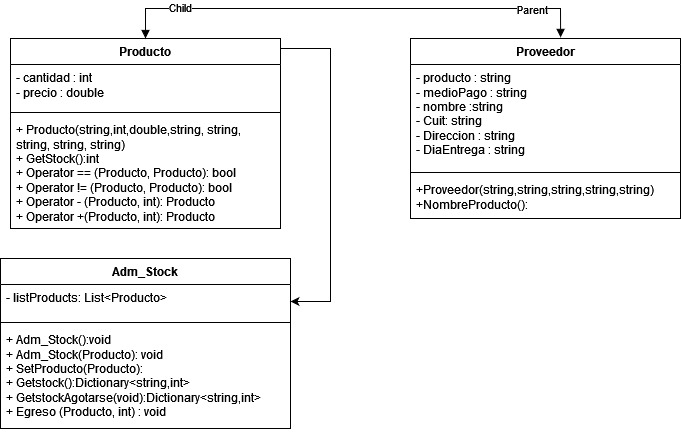


Diseño Técnico

**Administración de Stock**

Este proceso tendrá una administración de ingresos/egresos de productos. Para lograr esta funcionalidad se realizara tres clases donde Producto y Proveedor tendrán una herencia entre si y la clase global de la administración de STOCK dispondrá del PRODUCTO como una lista para gestionar.

A continuación se detalla la definición de las clases:



1. Producto y Proveedor

En primer lugar, definimos una clase llamada PRODUCTO donde tendrá como estado privado la Cantidad y el Precio del mismo. Tendrá un único constructor donde recibirá los estados del PRODUCTO y del proveedor. El nombre del producto y sus datos serán enviados a la clase padre por medio de una herencia.

La clase Producto tendrá una propiedad de acceso público donde realizaran la siguiente acción:

* Consulta del STOCK disponible del producto usando el estado CANTIDAD.

Las acciones anteriores solo podrán ser administradas por el **ENCARGADO**.

Tendrá operadores para poder administrar el Stock y búsqueda del producto:

* Operador ==, serán iguales si ambos productos son del mismo nombre.
* Operador ¡=, igual al punto == pero lo contrario.
* Operador – restara Stock (Cantidad).
* Operador + sumara Stock (Cantidad).

Por último, la clase PROVEEDOR la definimos como una clase abstracta pública que será utilizada para la herencia. El proveedor es el que genera el Producto y tendrá el nombre del mismo como los datos del proveedor para poder realizar la entrega del stock. Tendrá un solo constructor de instancia donde recibirá los valores de cada estado para inicializarlos y una propiedad para identificar el nombre del producto.

1. Administración Stock

La clase de Administración de STOCK tendrá como objetivo listar todo los productos y administrar el STOCK de los mismos. Como definimos una clase PRODUCTO, este mismo tipo de dato lo vamos a definir como un estado de la clase de tipo LISTA porque el producto puede ser de varios tipos.

La clase tendrá un constructor por defecto donde instanciara la lista y luego otro donde recibirá el producto a cargar como inicial. En caso de listar más productos, se definió una propiedad donde se encargara de realizar el SET del producto en la lista.

Por último, tendrá funcionalidades para la consulta y egreso de STOCK, esto lo definimos de la siguiente manera:

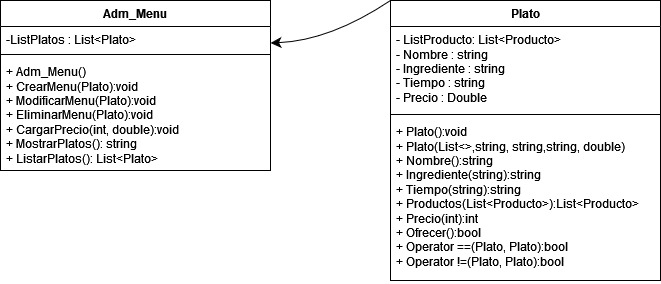
* Consulta de Stock, retornara una lista de Productos y sus cantidades.
* Consulta de Stock por agotarse, al igual que el punto anterior, retorna una lista pero para aquellos productos que están por agotarse. Se definió un mínimo de 10 unidades para que entre en esta lista.
* Egreso, esta funcionalidad descontara el stock del producto a la hora de utilizarse en una mesa.

**Administración de Menú**

* **FALTA RELACION DE ADMINISTRACIÓN DE STOCK (Quitar el producto)**

Este proceso tendrá la funcionalidad de administrar el menú que contendrá el RESTO. Hay cierta funcionalidad que solo podrá ser ejecutada por el COCINERO y el ENCARGADO. Para lograr su funcionalidad generamos 2 clases donde una determine el detalle completo de un PLATO y el otro toma este objeto en forma de lista y realiza la variedad del MENU con funcionalidad de administrar e informar.

A continuación se detalla la definición de las clases:



1. Plato

Definimos una clase para el detalle principal del Plato donde contendrá información como:

* Productos (Definido como lista, puede ser uno o más, los productos ingresados se irán descontando al Adm. Stock.).
* Nombre del plato
* Ingrediente
* Tiempo de producción
* Precio

La clase tendrá un constructor por defecto donde instanciara la lista de Productos para su uso. Luego tendrá un constructor para cargar los detalles del Plato incluyendo la lista de los productos a utilizar.

Tendrá 6 propiedades de acceso a sus estados donde realizara lo siguiente:

* Consulta al Nombre del Plato.
* Consulta y modificación del valor Ingrediente.
* Consulta y modificación del valor Tiempo de producción.
* Consulta y modificación a la lista de Productos.
* Consulta y modificación al precio del Plato.

Por último, se define 3 métodos que realizaran las siguientes acciones:

* Mostrar Detalle, arma un detalle de la lista de los platos con sus productos e ingredientes.
* Operador ==, dos platos son iguales si sus nombres son el mismo.
* Operador ¡=, dos platos son distintos si sus nombres son distintos, reutilización del punto anterior.

1. Administración menú

Esta clase será la encargada de administrar el Menú del RESTO. Como los platos pueden variar entre productos, variedad y precio entonces se define como un atributo de tipo lista de la clase PLATO.

La clase tendrá un solo constructor de instancia donde internamente instanciara la lista de Plato para su uso. En caso de generar, modificar o eliminar los platos cargados se definió unos métodos para dicha funcionalidad, esto solo lo puede hacer el **COCINERO**:

* Crear Menú, genera un menú nuevo y lo carga a la lista de PLATOS
* Modificar Menú, recibe un plato y lo modifica en la lista.
* Eliminar Menú, busca el menú a eliminar y lo quita de la lista.

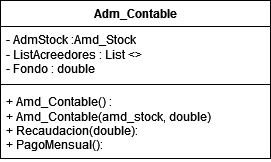
Tendrá un método para mostrar el detalle de todos los platos, esto incluirá el nombre del plato, los productos utilizados y la cantidad. A este método lo vamos a definir como Mostrar Platos. Luego, podrán listarse aquellos platos que no dispongan de stock suficiente en sus productos, esto lo definiremos como Listar Platos.

Por último y muy importante, se define un método para cargar el precio de cada plato, esto solo lo puede realizar el **ENCARGADO**.

**Administración Contable**

En esta instancia, la funcionalidad es administrar la contabilidad del negocio. Se opera el ingreso de STOCK, los pagos y los medios utilizados para realizarlos, la recaudación diaria y los pagos mensuales de sus empleados. Para realizar esta funcionalidad definimos una clase donde tenga internamente la administración de STOCK.

A continuación se detalla la definición de las clases:



Definimos la clase con 3 estados donde una de ellas será la administración de STOCK, la lista de acreedores y los fondos.

Tendrá un constructor por defecto donde instanciara la lista inicial de los acreedores. El segundo constructor se encargara de recibir la administración de stock y los fondos.

* FALTA LISTA DE ACREEDORES

Tendrá un método para la recaudación diaria donde se tendrá en cuenta la suma total de las mesas del local y el delivery del día. Este método recibirá por parámetro el monto de lo recaudado y se sumara al fondo.

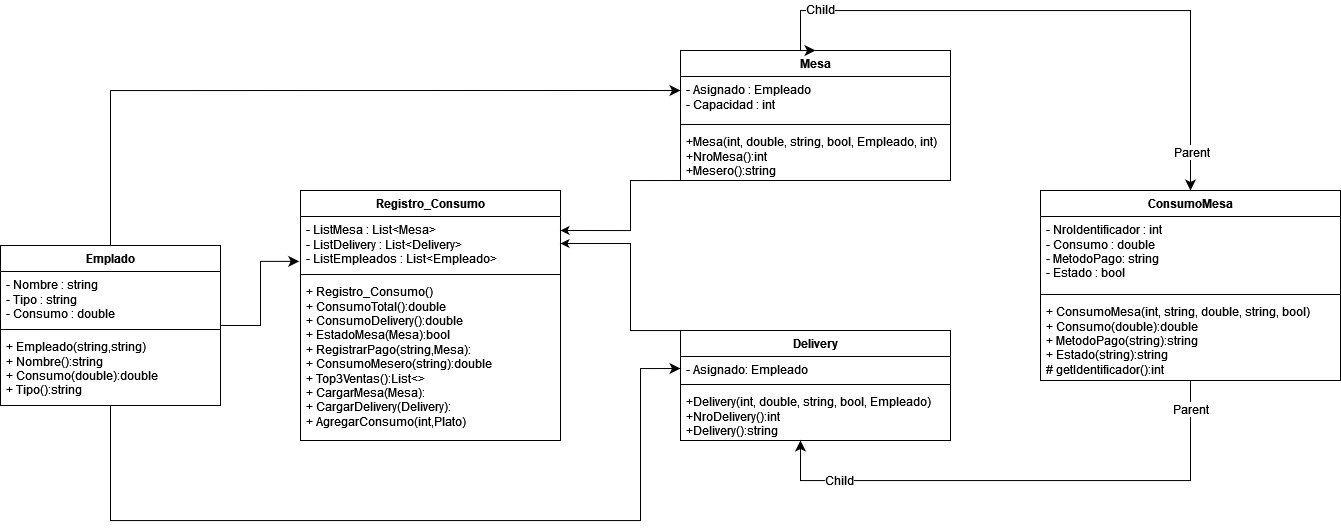
Por último, tendrá un método para el pago mensual que será encargado de validar los fondos disponibles para realizar el pago. En caso de tener una limitación con los fondos se realizara la prioridad de pagos en el siguiente orden:

* + 1. Se abona a los encargados
    2. Se abona a los cocineros
    3. Último, meseros y delivery.

**Registro de Consumos**

Esta sección administrara el registro de los consumos de cada mesa y delivery. Para realizar la funcionalidad de esta operación definimos 3 clases donde 2 serán los consumos entre Delivery y Mesa y el último será el registro y administración de los dos.

A continuación se detalla la definición de las clases:



1. Empleado

Definimos una clase Empleado para la identificación de Mesero y Delivery. Tendrá datos básicos y propiedades para retornar los valores que se requieran.

1. Consumo Mesa

Relacionamos la Mesa y el Delivery en una clase padre que contenga la información que comparten entre sí:

* Nro. de Identificador
* Consumo
* Método de Pago
* Estado

Tendrá un constructor de instancia para recibir los estados y propiedades para poder acceder a esta información. Solo será modificable por estas propiedades los siguientes datos:

* Consumo
* Método de Pago
* Estado

1. Mesa

Para la mesa le definimos una clase donde tendrá como estado los siguientes detalles para identificar:

* Capacidad
* Mesero asignado

Tendrá un constructor de instancia donde recibirá la información general de la MESA, el mesero asignado y la capacidad. Los datos generales lo enviaran a la clase padre.

Tendrá propiedades de acceso solo lectura para la identificación del Nro. De Mesa y el Mesero asignado. Las demás propiedades de modificación y lectura los realizara la clase padre, esto está definido en la sección 1. Consumo Mesa.

1. Delivery

Para el Delivery le definimos una clase donde tendrá como estado los siguientes detalles para identificar:

* Delivery asignado

Tendrá un constructor de instancia donde se le asignara los detalles generales del Delivery y la persona asignada. Los datos generales se los enviara a la clase Padre.

Tendrá propiedades de acceso solo lectura para la identificación del Nro. Delivery y el Delivery asignado. Las demás propiedades de modificación y lectura los realizara la clase padre, esto está definido en la sección 1. Consumo Mesa.

1. Registro Consumo

Esta clase se encargara de registrar/administrar los consumo del local. Para ello, generamos dos atributos de tipo LIST de la clase MESA y DELIVERY.

Tendrá un solo constructor de instancia donde instanciara las listas que tiene como estado.

Esta operación tendrá varias funcionalidades:

* Para el Consumo Total se realizara una sumatoria total del consumo de mesas y delivery.
* Para el consumo Delivery solo se realizara la sumatoria total del consumo sobre el Delivery.
* Para la verificación de estado de mesas se implementa un método donde obtendrá el estado del mismo.
* Para registrar un pago se implementara un método que recibirá el medio de pago y si es Mesa o Delivery, esto lo impactara en la lista del estado de la clase.
* En caso de obtener el consumo total por mesero se implementara un método donde recibirá el mesero en la cual se desea saber la información y retornara el consumo total del mismo.
* Se definirá un método donde retornara el top 3 de las ventas totales con mayor consumo.
* Tendrá un método para cargarlas mesas y otro para poder cargar el Delivery. Estos estarán por separados y se irán cargando en las listas del estado de la clase.
* Por último, se define un método para agregar el consumo a la Mesa/Delivery. Esto lo va hacer por medio del identificador y el Plato seleccionado.