Programación Orientada a Objetos 2
UNQ

Trabajo Final

2do. Sem. 2023

Terminal Portuaria

Github

Integrantes:
- **Cardozo Lara Noelí**
- **Marzaroli Candela**
- **Roldan Marcos**

Decisiones de diseño:

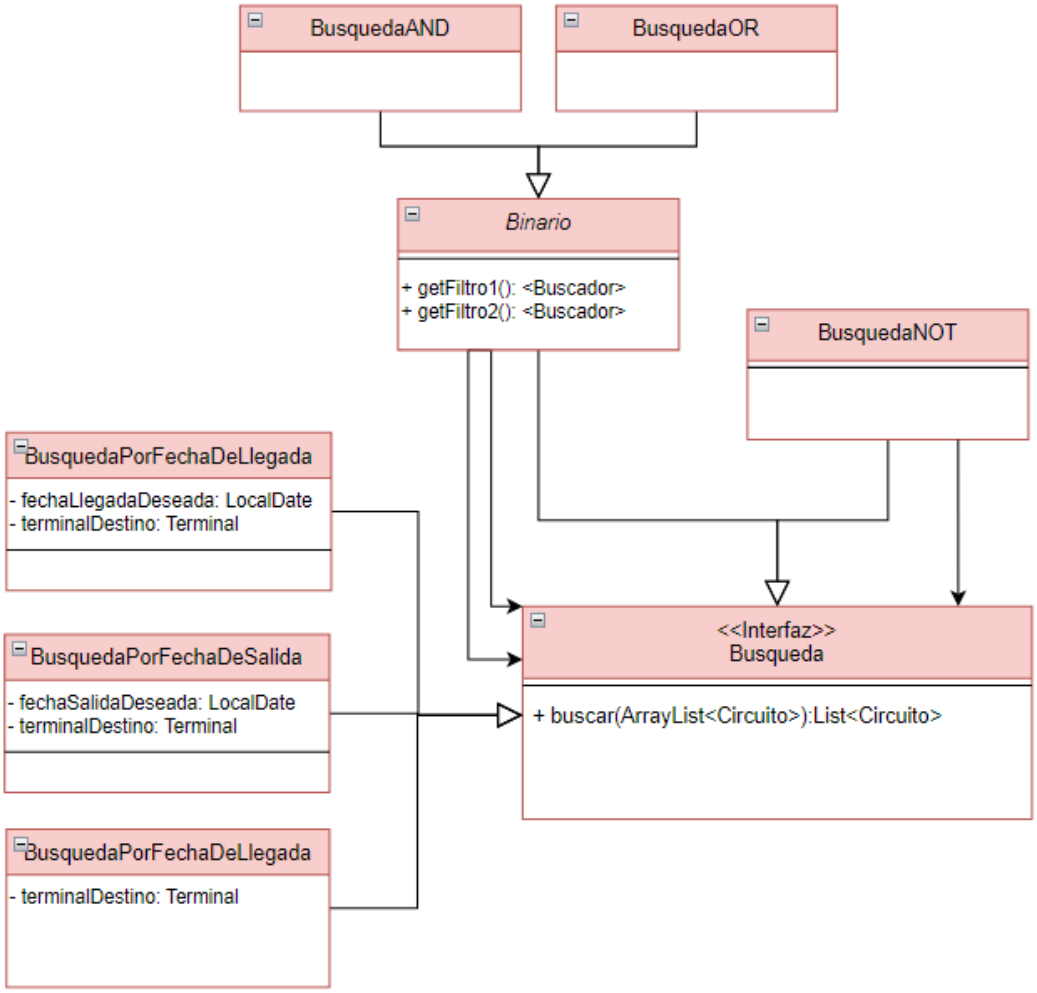
- La factura se enviará al consignee al momento de la entrega. La responsabilidad del pago se dejará a la negociación entre ambas partes.
- El tiempo que tarda un tramo en ir desde la terminal inicial a la terminal destino, lo retornamos con un **int**, el cual hace referencia a la cantidad de días que tardará en recorrer dicho tramo
- El servicio de electricidad se debe contratar previamente considerando la cantidad de días que va a estar en la terminal, ya que sino debería ser cobrado como un servicio de almacenamiento específico para los container de tipo Reefer.
- El servicio de almacenado nunca se factura, ya que la factura se envía cuando el buque deja la carga. En ese caso se debería realizar una sobrefacturación específica para ese servicio, pero no es algo que está contemplado en nuestro sistema.

Patrones de diseño utilizados y roles según la definición de Gamma et. al.:

Aplicamos un patrón composite para nuestro buscador de circuitos complejos. Lo usamos ya que nos permite combinar los buscadores no atómicos de maneras complejas que permiten recursión.

Los roles que cumplen nuestras clases son:

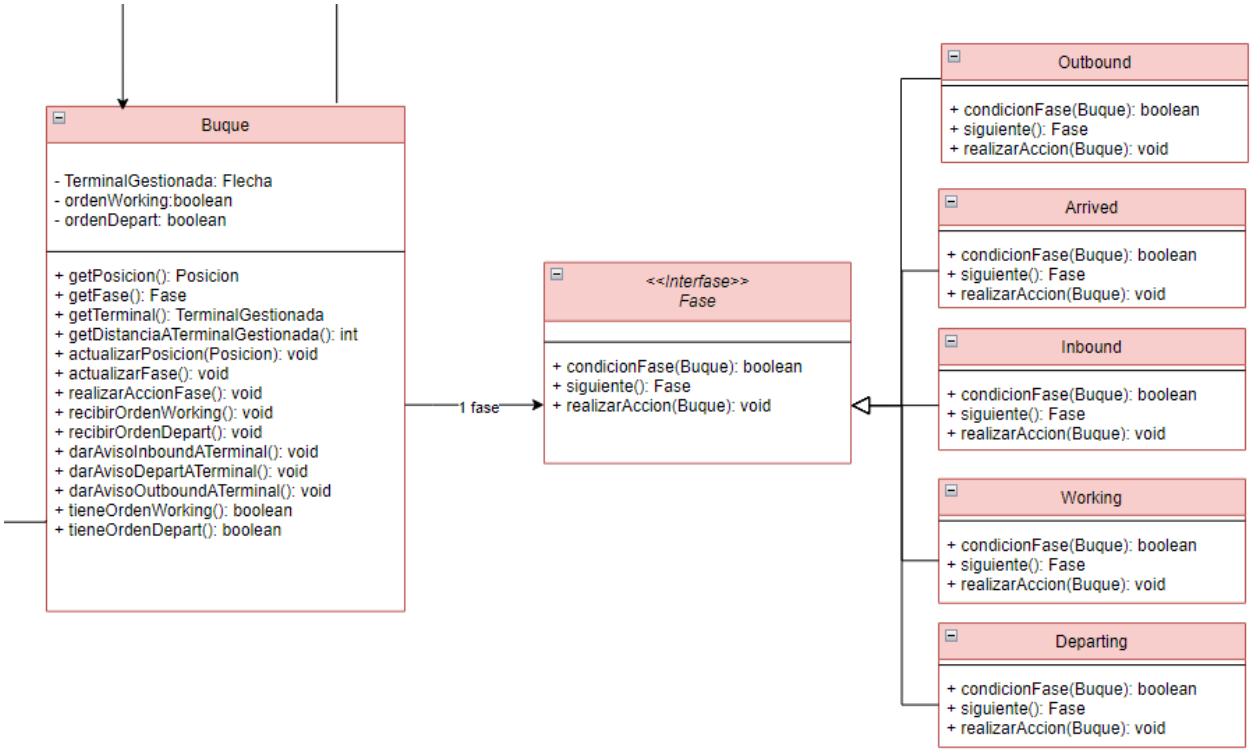
- Component: Interfaz de Busqueda
- Composites: Binario, BusquedaNOT
- Hojas: FechaLlegada, FechaSalida, PuertoDestino
- Cliente: TerminalGestionada.



Para representar las diferentes fases de nuestros buques decidimos usar el patrón de State, ya que estas fases dependen de algo interno, ya sea de la distancia a la terminal gestionada o de si tienen ordenes especificas.

Los roles que cumplen nuestras clases son:

- Context: Buque
- State: Interfaz de Fase
- Concrete States: Outbound, Inbound, Arrived, Working , Departing, Buque



Por último, para nuestro buscador de el mejor circuito usamos un patrón strategy, ya que este no cuenta con busquedas complejas como el anterior, sino de 3 condiciones de “mejor” preexistentes, y deben de poder ser cambiadas dinámicamente.

Los roles que cumplen nuestras clases son:

- Context: Naviera
- Strategy: Interfaz de BuscadorMejorCircuito
- Concrete Strategies: MenorPrecio, MenorTiempo y MenorCantidadTerminales