



U N I V E R S I D A D
Panamericana

Samantha Licea Domínguez

Patrones de Diseño y Arquitecturas de Software

“Proyecto Final”

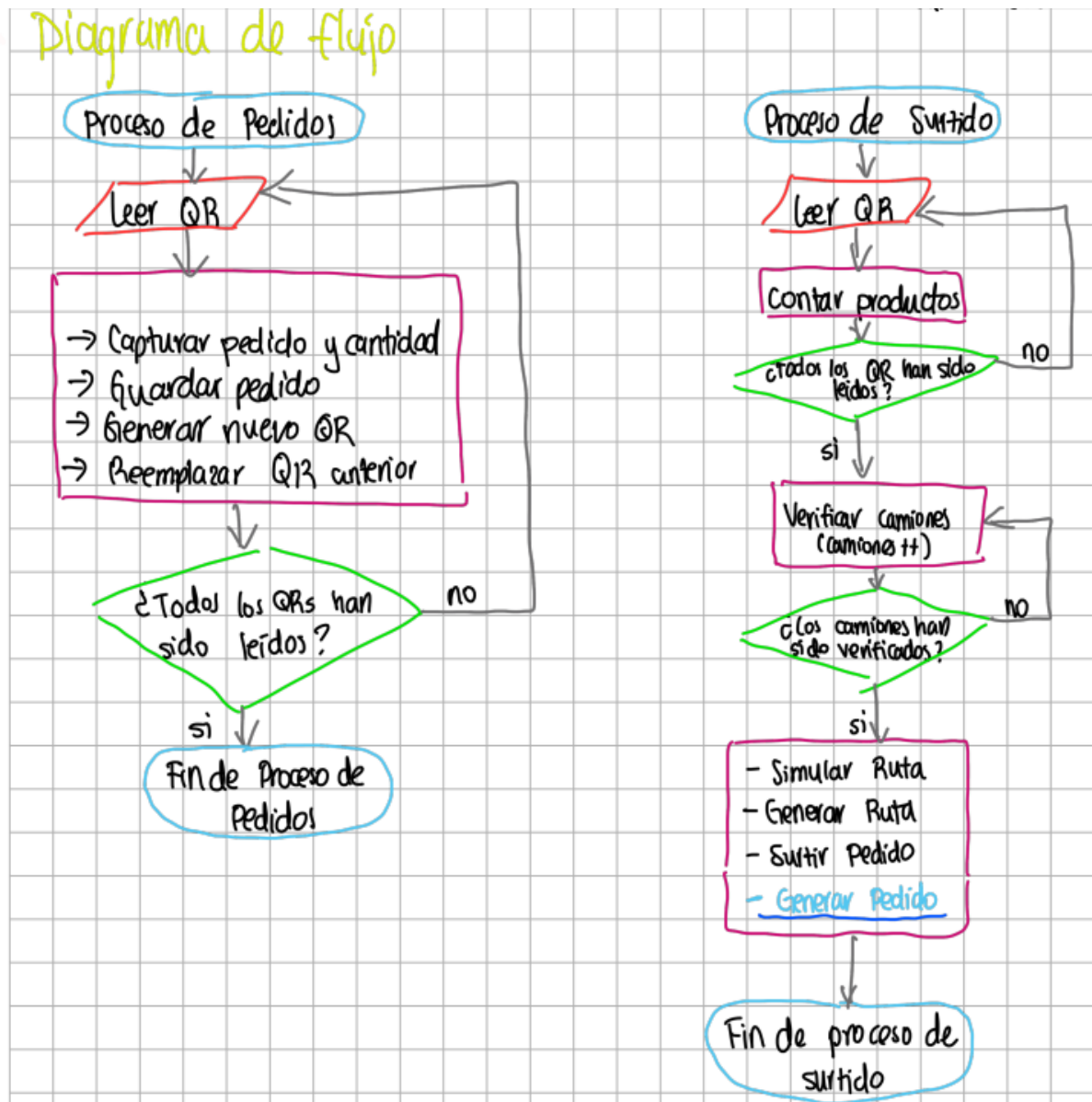
16/05/2022

6to Semestre

Proyecto Final

El proyecto final consiste en realizar una aplicación con Windows Forms, el cual simule los procesos de pedir y surtir pedidos. Para ello se tiene el flujo que seguirá la aplicación. Se pueden distinguir tres procesos principales:

1. Pedir productos.
2. Surtir pedidos.
3. Generar bitácora.



Los patrones a utilizar son:

1. Pedido: Builder (creacional).
2. Surtido: Facade (estructural).
3. Bitácora: Strategy (de comportamiento).

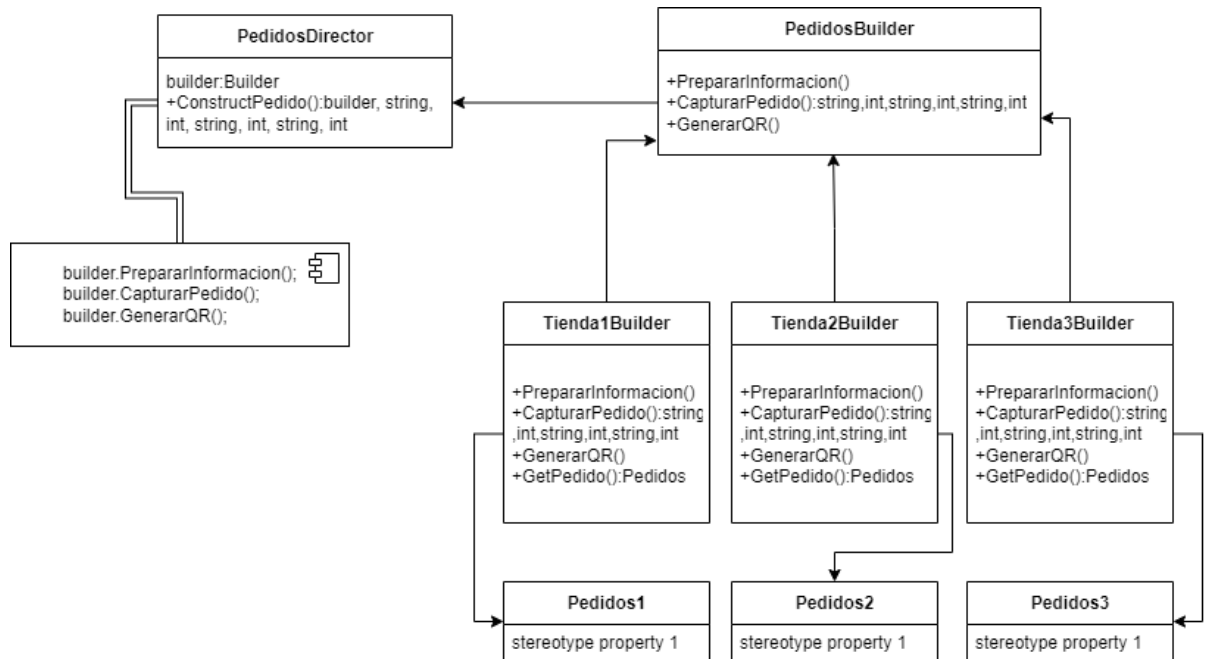
Justificación de Patrones de Diseño

1. ¿Por qué usar **Builder**?

Al leer el problema, se observó que para generar los pedidos de las tiendas, se tenía que seguir una secuencia de pasos:

1. Leer un QR
2. Capturar productos y cantidad
3. Guardar QR

Tomando en cuenta lo anterior, se realizó un builder como se muestra en el siguiente diagrama UML.



Ahora bien, tomando el anterior diagrama UML del patrón Builder se tienen las siguientes clases:

- a) **PedidosDirector**: corresponde a la clase Director el cual define el orden que va a seguir el Builder: LeerQR(), CapturarPedido() y GenerarQR().
- b) **PedidosBuilder**: corresponde a la clase abstracta Builder, la cual contiene los métodos abstractos anteriores, además de la obtención del producto, para obtener el resultado.
- c) **Concrete Builder**: corresponde a las clases concretas para cada una de las tiendas (Tienda1Builder, Tienda2Builder, Tienda3Builder) con sus respectivos métodos implementados.
- d) **Product**: corresponde a la clase que tiene el resultado. En este caso el resultado simplemente arroja un mensaje de “success” correspondiente a la realización de cada uno de los métodos, ya que dentro de la implementación se guardan los JSONs y los QRs de cada tienda.

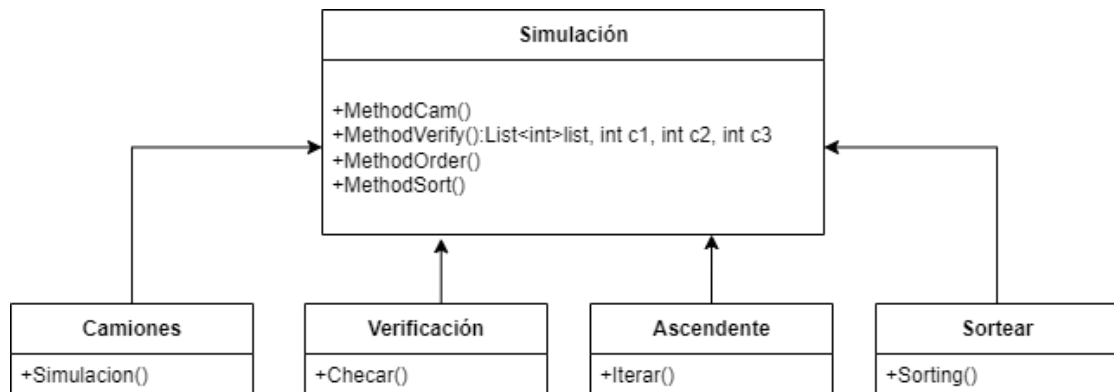
Entonces, en pocas palabras, el Builder ayuda a generar los pedidos por tienda, generar sus respectivos JSONs y sus QRs.

2. ¿Por qué usar **Facade**?

Para el segundo proceso de la Aplicación (simulado y surtido) se consideró utilizar un Facade que pudiera implementarse para cada una de las fases:

1. Simulación de ruta (verificar camiones).
2. Generar Ruta (ordenar en orden descendente).
3. Surtir pedidos.

Como considero que es un grupo complejo de clases que trabaja con JSONs, Listas y Diccionarios, lo que podría usarse es precisamente un Facade.



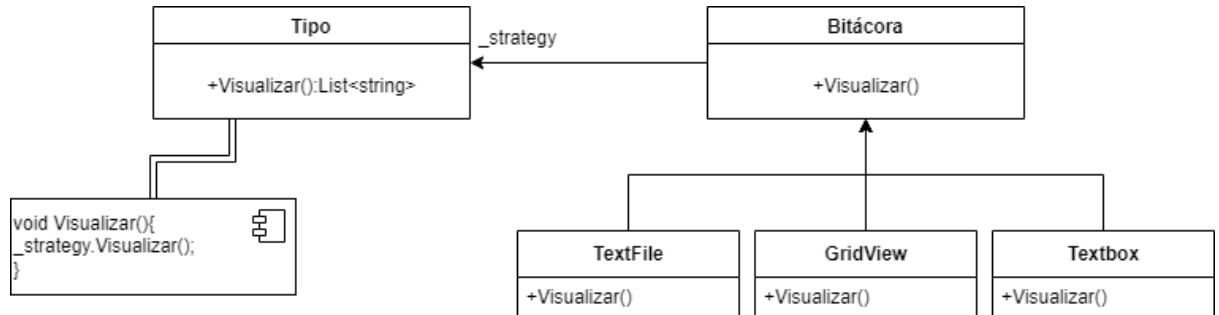
Observando el diagrama UML, se puede observar que tenemos 4 clases secundarias o subsistemas de la clase Simulación (funciona como **Facade**). Dentro de cada clase lo que se hace primordialmente es adecuar la información de las tiendas para poder realizar la simulación:

- a) **Clase Camiones:** devuelve una lista de enteros (List<int>) para poder visualizar los productos que van a ir en los camiones.
- b) **Clase Verificación:** devuelve una lista de booleanos (List<bool>) que verifica que los productos se repartan en los camiones y ayuda a determinar cuántos camiones de cada producto van a salir a la ruta.
- c) **Clase ascendente:** devuelve una lista de strings (List<string>) que ayuda a visualizar los productos de cada tienda y la ganancia total.
- d) **Clase Sortear:** devuelve un diccionario con llave string y valor entero (Dictionary<string,int>) que acomoda las tiendas en orden descendente de acuerdo a su ganancia total. Esto para determinar la ruta que van a seguir los camiones.

En conclusión, el método Facade nos ayuda a encerrar la llamada a los métodos para hacer la simulación de pedidos. Una vez que la simulación está en orden, prosigue al proceso de la Bitácora.

3. ¿Por qué usar Strategy?

Para poder llevar a cabo el último proceso de la Aplicación, se utilizó un Strategy. Esto porque lo que hace la aplicación simplemente es llevar la bitácora y obtenerla de distintas formas (archivo de texto, Grid View, Text Box). El Strategy en este caso lo que hace es preparar la información correspondiente a la bitácora, y la despliega de distintas formas. A continuación se muestra el diagrama UML hecho específicamente para la Bitácora.


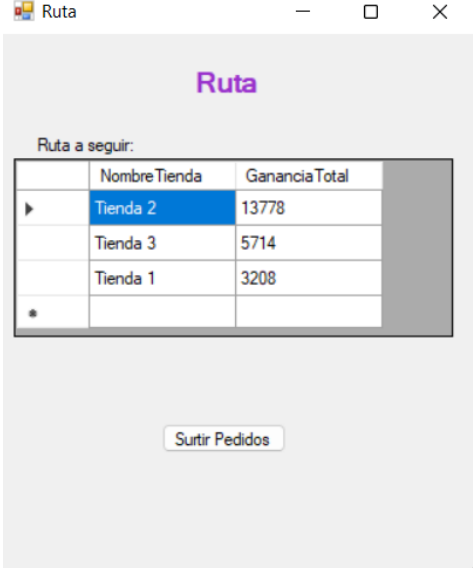



Como se puede observar, tenemos la clase Tipo (funciona como el **Contexto**) que va a mandar llamar a la clase abstracta Bitácora (funciona como la **Estrategia**). Y ésta última a su vez, tiene sus clases concretas, lo que ayuda a desplegar la información como se pide. En esta clase, tenemos nuestro método principal que es Visualizar, el cual regresa una Lista de tipo string con toda nuestra información lista para ser procesada.

Vista de la Aplicación

La aplicación cuenta con 5 Formularios de Windows como se muestran a continuación:

| | |
|---|--|
| <p>The 'LeerQR' window displays a red 'BIENVENIDO USUARIO' header. It has two input fields labeled 'Nombre Tienda:' and 'ID Tienda:'. Below them is a large empty box. At the bottom, it asks '¿Qué desea hacer?' with two buttons: 'Simulación' and 'Leer QR'.</p> | <p>The 'CapturaPedidos' window has a blue 'CAPTURA DE PEDIDOS' header. It shows 'ID TIENDA: 1' and 'NOMBRE TIENDA: Tienda 1'. Below is a table with two columns: 'PRODUCTO' and 'CANTIDAD'. The table lists 'Refrescos' (max 360), 'Pan Fresco (pieza)' (max 810), and 'Verdura Congelada (1kg)' (max 285). Each row has a spinner control for the quantity. A 'Generar Pedido' button is at the bottom.</p> |
| <p>Formulario para leer QRs, capturar pedidos e ir a la simulación.</p> | <p>Formulario para capturar los pedidos dependiendo de la tienda que se elija.</p> |

| | |
|--|--|
|  |  |
| <p>Formulario para simular la ruta de pedidos en donde se muestra la cantidad total de los productos a repartir.</p> | <p>Formulario para visualizar la ruta a seguir dependiendo de la ganancia.</p> |
|  | |
| <p>Formulario para la obtención de la bitácora de las 3 formas diferentes que se piden.</p> | |

La aplicación sigue los siguientes pasos:

1. Seleccionar un QR de entre los 3 que hay en la carpeta para leer.
2. Una vez seleccionado, aparecerán el nombre e id de la tienda en la parte superior y el usuario puede ir a capturar el pedido de la tienda.
3. Una vez capturados los pedidos por las 3 tiendas disponibles, el usuario procede a ir a la simulación (a pesar de que el usuario en cualquier momento puede ir a la

simulación, es recomendable capturar todos los pedidos antes, ya que si no, se tomará el pedido anterior).

4. En la simulación, el usuario puede ver la cantidad de cada producto a surtir, por lo que deberá seleccionar hasta 3 camiones por producto para surtir la mercancía.
5. Una vez validados los camiones, se procede a generar la ruta en orden descendente de acuerdo a la ganancia por tienda.
6. Posteriormente se redirecciona a la obtención de la bitácora, donde el usuario puede seleccionar el formato en que la quiera obtener (txt, grid view, text box).
7. Una vez que termina este proceso, el usuario da click en terminar y regresa al formulario de lectura de QRs.