Justificación taller 1

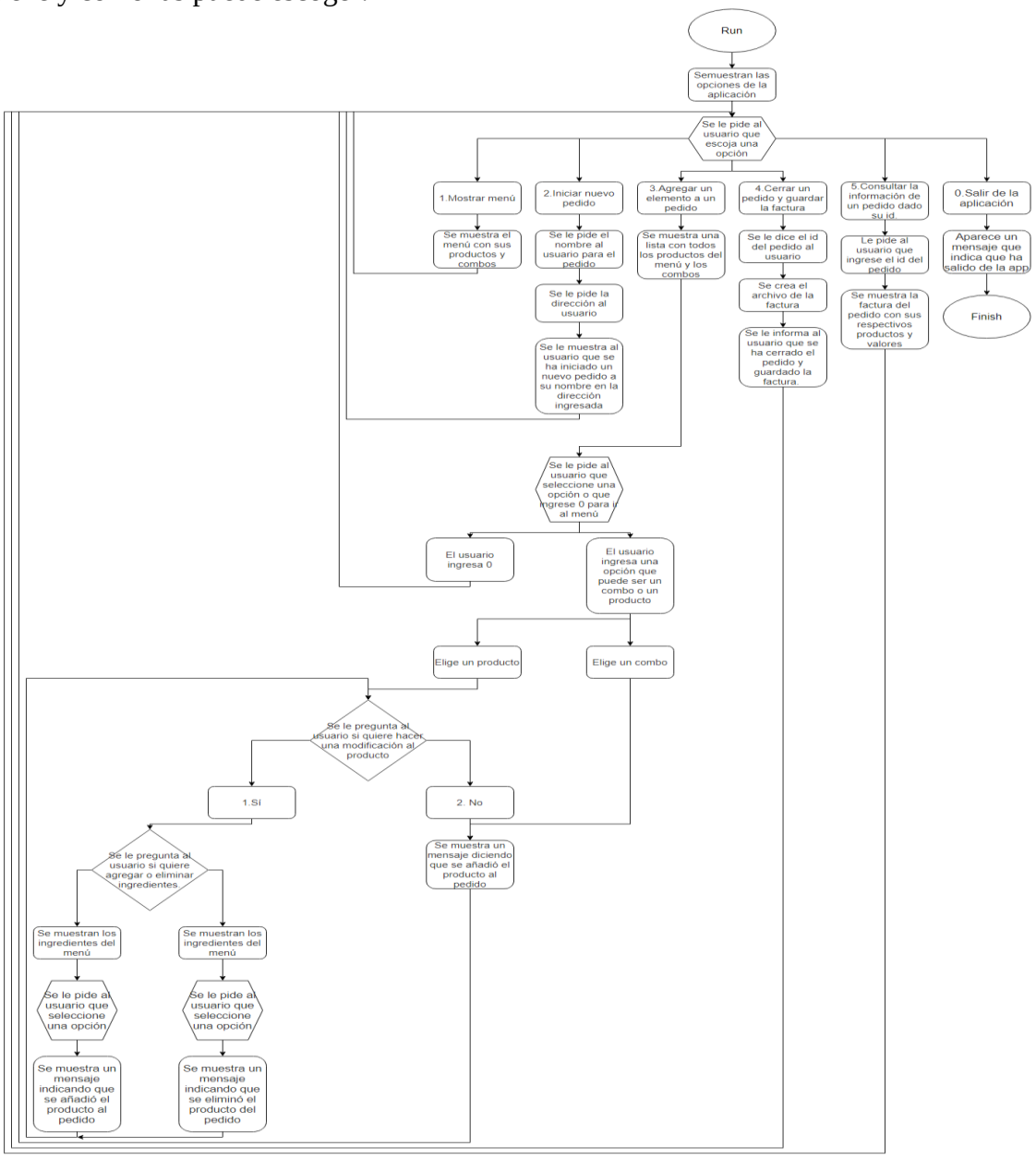
Alejandro Hernández - 202111716

Pablo Pedreros - 202112491

Luis Ernesto Tejón – 202113150

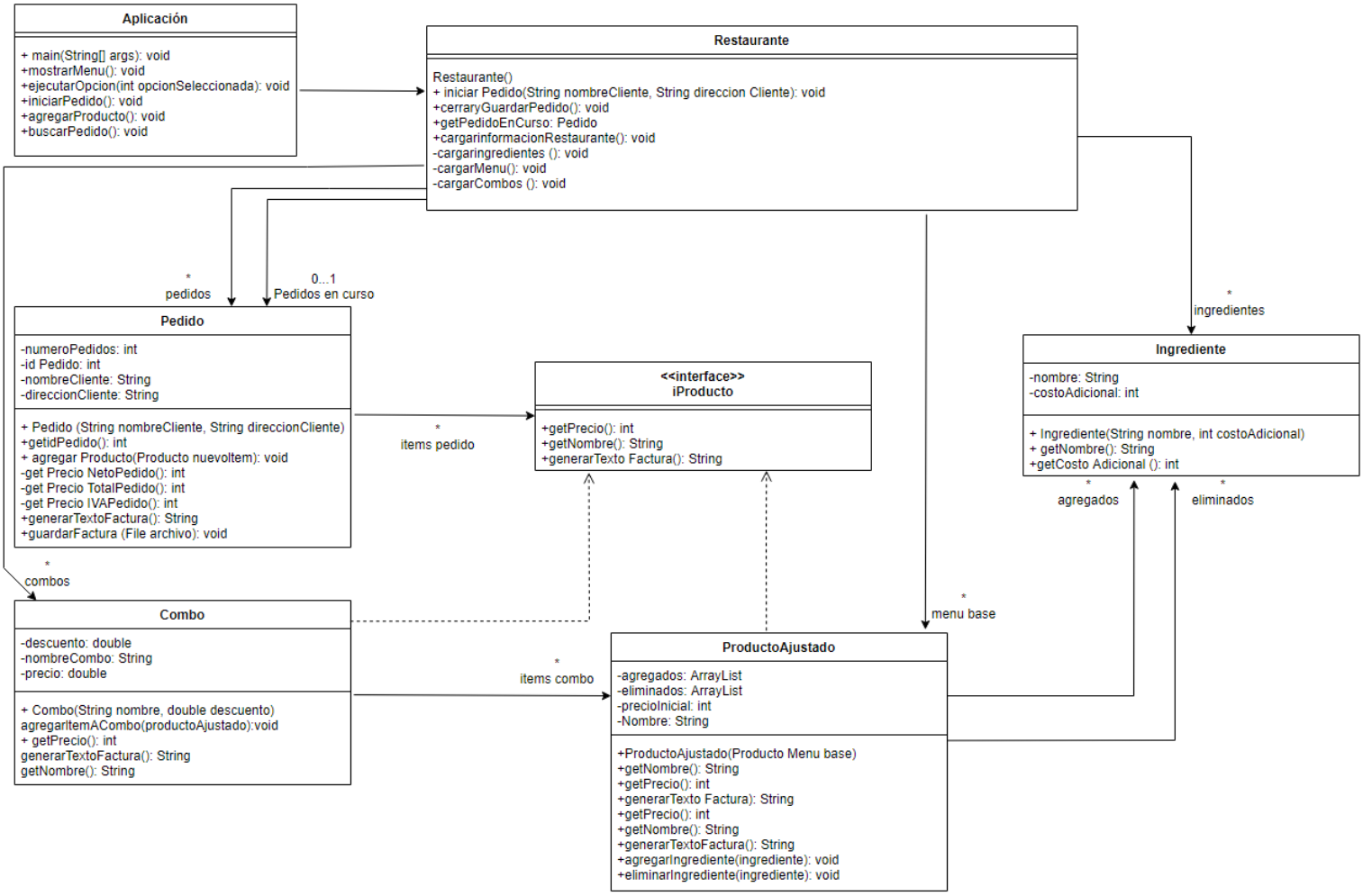
1. **<u>Mapa de opciones de usuario</u>**:

El siguiente mapa muestra cómo se relaciona el usuario con el programa, cuáles son las opciones qué tiene y cómo las puede escoger:



2. **<u>Modelo UML del proyecto</u>**:

A continuación, se muestra el modelo UML que explica cómo funcionan y se relacionan las clases del proyecto:



3. **<u>Modificaciones respecto al modelo original</u>**:

A continuación, se listan las modificaciones que hicimos en la implementación del proyecto respecto al modelo propuesto en el taller:

- <u>Aplicación.java</u>: en este archivo se mantuvieron los mismos métodos que se proponían en el modelo original, pero agregando los métodos **iniciarPedido**, **agregarProducto** y **buscarPedido**, de forma que casi toda la interacción del usuario con la aplicación para

ingresar datos suceda en este archivo de la consola y habilitando la opción de buscar un pedido por su ID, que no parecía estar en el modelo original.

- Restaurante.java: en este archivo eliminamos los métodos **getMenuBase** y **getIngredientes**, pues nunca les dimos uso durante la implementación, y cambiamos el nombre de la variable menuBase por productosMenu. Además, les quitamos los parámetros a los métodos **cargarInformaciónRestaurante**, **cargarIngredientes**, **cargarMenu** y **cargarCombos**, pues los archivos de texto con la información siempre serán los mismos.
- Pedido.java: en este archivo volvimos público el método **generarTextoFactura** para que se pueda usar desde el paquete de la consola a la hora de buscar la información de un pedido del historial.
- ProductoMenu.java: se eliminó por completo esta clase, pues se podía cumplir su función en el programa haciendo unas pequeñas modificaciones en la clase **productoAjustado**, y usando esta última para reemplazarla a la vez que cumple su función original.
- ProdcutoAjustado.java: como esta clase cumplirá las funciones de la anterior clase **prodcutoMenu**, ya no recibirá un producto por parámetro, sino un nombre y un entero que servirá como **precioInicial**. Así, el atributo que se tendrá es el del precio original del producto, y al momento de implementar el método **getPrecio**, se hará un ciclo por todos los ingredientes de la lista de **agregados**, sumando su precio al precio original para retornarlo. De forma parecida, la función **getNombre** iterará por las listas de **agregados** y **eliminados**, verificando si hay que agregar al nombre la especificación de que se agregó o eliminó un ingrediente. Por último, para agregar ingredientes a estas dos listas se crearon los métodos **agregarIngrediente** y **eliminarIngrediente**.
- Combo.java: por último, en este archivo se agregó el atributo **precio**, que se inicializará en 0 e irá aumentando conforme se agreguen los productos. Al ya no existir la clase **productoMenu**, el método **agregarProductoACombo** tendrá como parámetro un objeto de la clase **productoAjustado**, pero funcionará de la misma forma que el original.