Laboratorio N°1:
Paradigma Orientado a Objetos

Tomás Rando, Enzo Palau, Julián Montaño

Paradigmas de la Programación
Licenciatura en Ciencias de la Computación
Facultad de Ingeniería, Universidad Nacional de Cuyo

20 de Septiembre de 2023

Abstracto:
En el presente trabajo abordaremos el problema de la creación de un software para el sistema de control de una bodega, a continuación se encontrará el encare de la solución propuesta, así como también las conclusiones obtenidas.

1. Introducción

En el presente informe se pretende dar solución al problema planteado en el Laboratorio N°1 de la cátedra de Paradigmas de la programación a cargo del profesor Dr. Pablo J. Vidal.

En dicho problema se plantea aplicar los conceptos del paradigma orientado a objetos a través de una aproximación de un programa para gestionar el sistema de trazabilidad de una bodega.

En el apartado de Metodología se describe cuál fue la aproximación e interpretación que se tuvo para este problema, explicando de forma detallada cada clase y sus métodos así como la interpretación de la misma. Además se encuentra un sub-apartado en el cual explicamos cuáles fueron los cambios y problemas encontrados durante el modelado de la solución propuesta.

En cuanto al apartado de Resultados hace referencia a como fueron cumplidos los requisitos del problema correspondiente, además una guía de cómo se ejecuta el .jar y cómo utilizar el programa.

2. Metodología

Para la realización de este problema comenzamos con una abstracción del problema, para ello utilizamos el lenguaje de modelado UML, lo que resultó en el siguiente diagrama de clase:

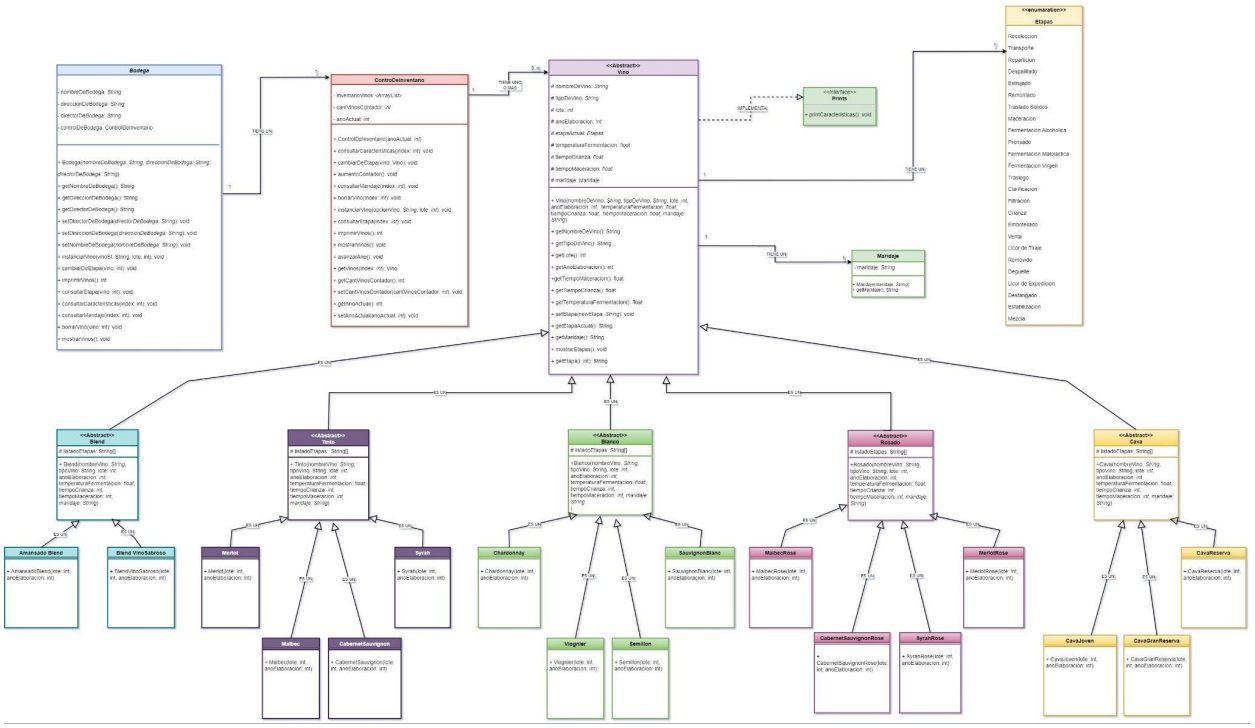

Figura 1: Diagrama de clases del modelado.

La implementación de este software fue a través del lenguaje de programación Java en su versión 11.

A continuación describiremos detalladamente cada clase y su funcionamiento.

2.1) Bodega

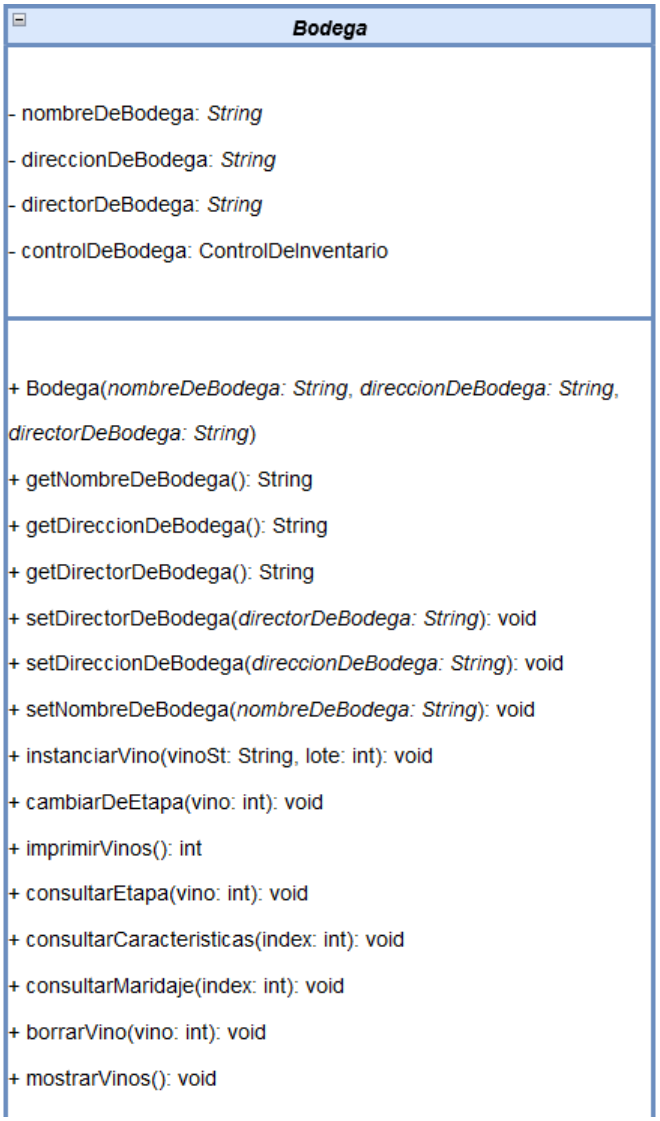

Figura 2: Diagrama de la clase Bodega.

La clase de Bodega se encarga de dar un primer nivel de abstracción a nuestro problema, ya que brinda datos iniciales como los atributos de nombre,dirección y director.
Además, lo más importante es su atributo el cual tiene un objeto de la clase ControlDeInventario. Esto nos servirá para poder hacer uso de las funciones de esta, ya que los métodos “instanciarVino”,”cambiarDeEtapa”,” imprimirVinos”,”consultarEtapa”, “consultarCaracteristicas”,”consultarMaridaje”,”mostrarVinos” y ”borrarVino” solo hacen una llamada a los métodos de igual nombre de el ControlDeInventario.

2.2) ControlDeInventario

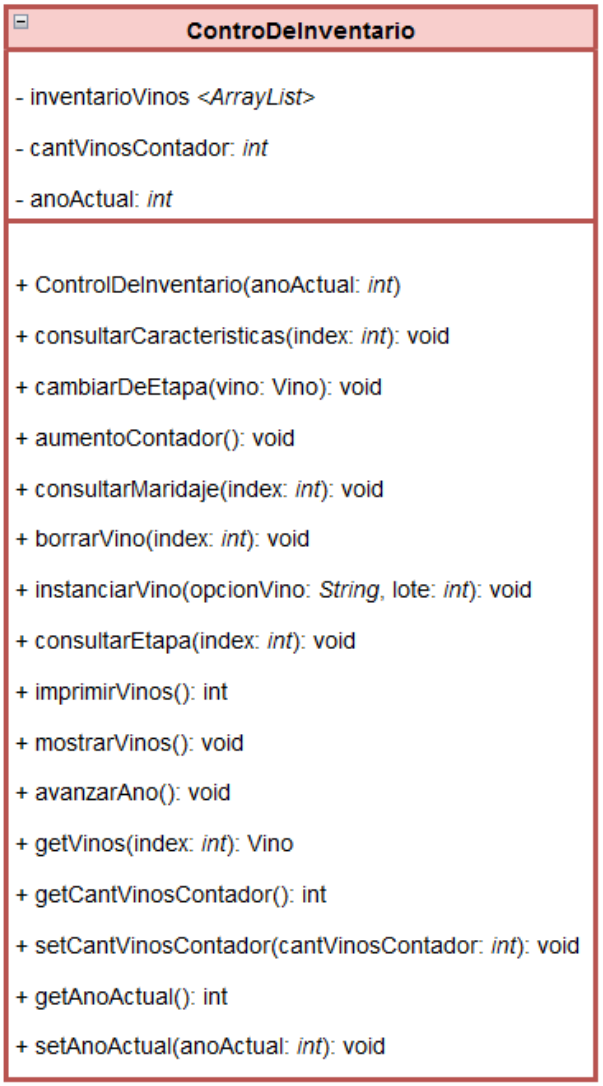

Figura 3: Diagrama de la clase ControlDeInventario.

Estamos ante probablemente la clase más importante de nuestro modelado, ya que esta se encarga de lo que tiene que ver con la gestión de los vinos,crearlos,almacenarlos y hacer consultas a los mismos.

Los atributos de esta clase son: “inventarioVinos”, el cual es un arraylist que almacena los vinos creados o en producción, “cantVinosContador” que almacena el número de vinos y “añoActual” para cambiar el año actual con el método “avanzarAño”.

En cuanto a los otros métodos, para crear vinos tenemos el método “instanciarVino” el cual crea un objeto de cualquiera de las subclases que hereden de vino, luego se almacena en el inventario y se aumenta el contador con el método “aumentoContador”, de esta forma se podrá almacenar en el sistema los diferentes vinos. El método “borrarVino” hace el proceso inverso.

Para el tema del cambio de etapas de los vinos tenemos “cambiarDeEtapa”, el cual es un método polimórfico, este recibe un objeto “Vino” pero este vino puede ser de cualquiera de los vinos que ofrecemos (subclases de vino) y ahí decidir a qué etapa pasar; cabe aclarar que podemos consultar la etapa con “consultarEtapa”.

Finalmente otras funciones como consultar el maridaje, características y mostrar los vinos se hacen con los métodos “consultarMaridaje”,”consultarCaracteristicas” y “imprimirVinos” respectivamente, estos métodos harán uso de nuestro Arraylist de inventario para ver los vinos previamente instanciados.

2.3) Vino, etapas, maridaje e interfaz

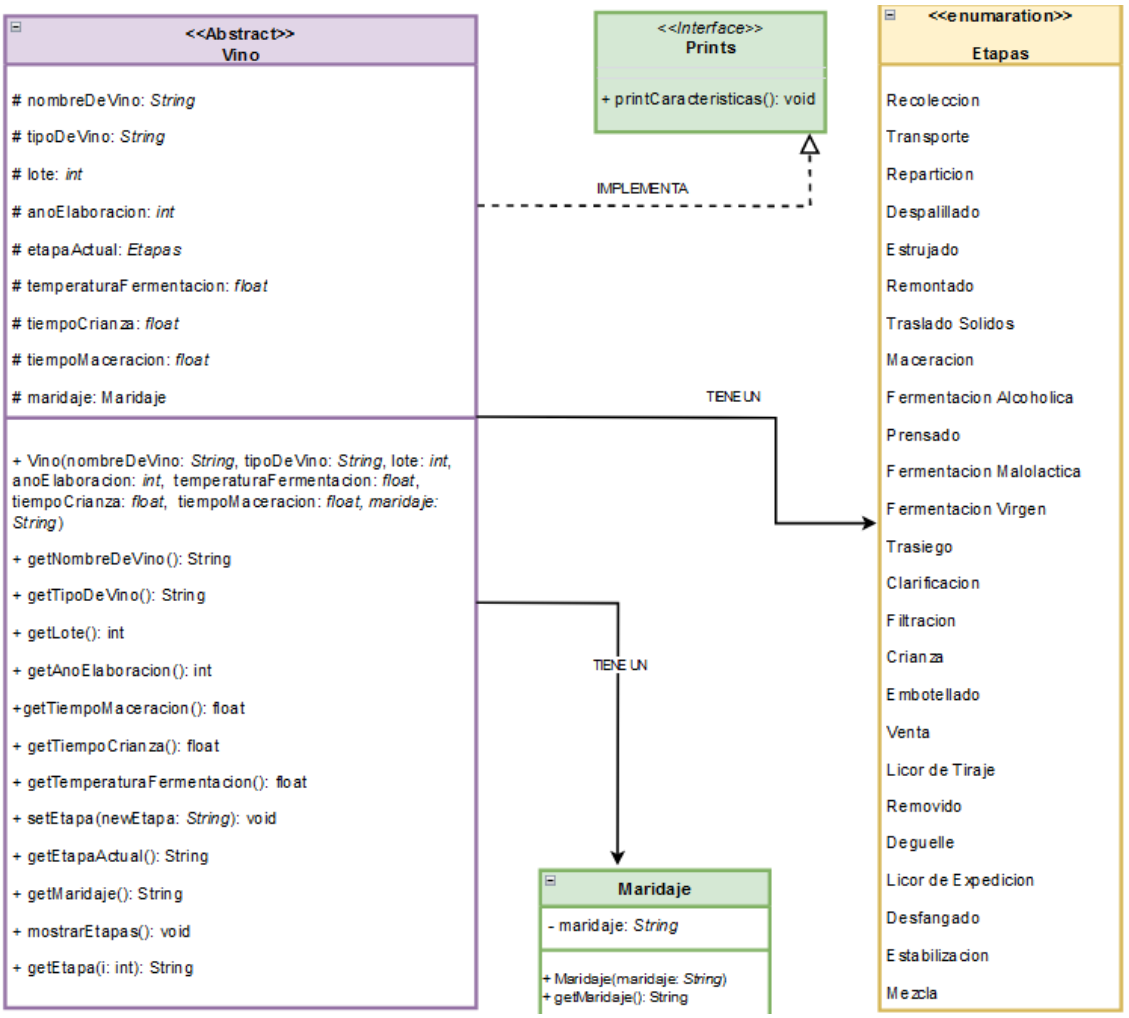

Figura 4: Diagrama de la clase Vino así como su relación con maridaje , el enumeration “Etapas” y la interfaz “Prints”.
La clase Vino es una clase abstracta que nos da un nivel de abstracción para crear los diferentes vinos, ya que los diferentes tipos de vinos comparten ciertos atributos como su nombre, su tipo, lote, año de elaboración, la etapa en la que se encuentran, su temperatura de fermentación, tiempo de maceración y crianza, y por último su maridaje.

El maridaje dentro del mundo del vino y la gastronomía, se corresponde, como su nombre indica, con el proceso de casar un alimento con un tipo de vino determinado. En este caso “maridaje” es un atributo el cual tiene un objeto de la clase Maridaje, el cual le dará en formato de String las comidas con las que se lleva mejor dicho vino.

La interfaz “Prints” la implementa “Vino” y es de utilidad para imprimir las características de los vinos particulares que heredan de Vino.

Por último las etapas son puestas en relación de asociación a un enumeration donde se listan las etapas del proceso de elaboración.

2.4) Tipos de vino y sus subclases

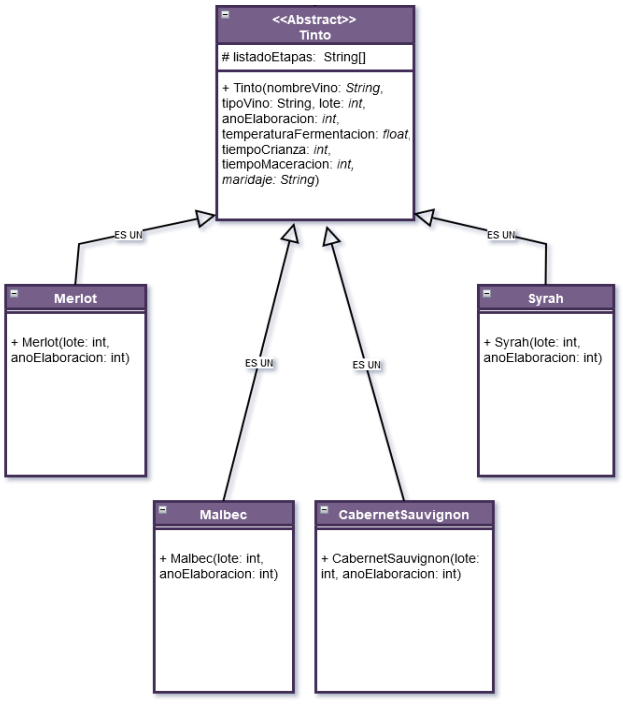

Figura 5: Un ejemplo de tipo de vino en general y sus tipos más específicos .
En esta sección de clases tenemos los vinos ordenados por tipos (tinto, blanco, rosado, blend y cava) en relación de herencia con la clase abstracta “Vino”, ya que en los vinos de un mismo tipo notamos que tienen las mismas etapas, de ahí la decisión de por ejemplo, como se ve en la imagen (Figura 5**),** la clase “Tinto” tiene un “listadoEtapas” en forma de array.

Por último en relación de herencia con cada tipo, tenemos a los subtipos más específicos como en la imagen (Figura 5**),** por ejemplo, la clase “Malbec”, estas son útiles para poder instanciar cada vino con sus respectivas características.

2.5) Herramientas utilizadas

Utilizamos para la creación del código el lenguaje de programación Java en su versión JDK 11.0.21, el entorno de programación IntelliJ, y otras herramientas útiles como github y git.
Para la creación del diagrama de clases utilizamos páginas como Miro y Drawio.

2.6) Cambios durante el trabajo y acotaciones

Para llegar a este esquema final pasamos por muchas fases de modelado. Primeramente, se intentó usar a la clase Bodega como la clase principal que se ocupaba de no solo dar la información inicial a la bodega (su nombre, dirección, etc), sino también de la creación y gestión de los vinos y de ahí derivar a dos clases principales, una referida a las etapas, la cuál agrupaba a todas sus subetapas y otra referida a los vinos, la cuál contenía a los tipos de vino (Blanco, Rosado, Tinto y Cava).

Luego del primer control, cambiamos el modelado pensado hasta el momento, puesto que nos dimos cuenta que hacer las etapas como

clases no tenía sentido, ya que lo que se pedía era hacer un sistema de trazabilidad de la bodega y las etapas podían repetirse entre distintos vinos. Por ello, borramos completamente la clase "Etapas" y creamos String Arrays con las etapas en la clase Vino.
Luego borramos las clases referidas a los tipos de vino, y creamos clases (las cuales heredan de la clase abstracta Vino) por cada vino que nuestra bodega produzca (clases como por ejemplo Malbec, Merlot etc).

Por último, nos pareció útil y mejor para el modelado del problema recortar la clase Bodega y desligarla de las responsabilidades de las funciones de crear vinos y gestionarlos. Decidimos que estas tareas podrían constituir una clase aparte. De ello nació "controlDeInventario".

Posterior al segundo control, repensamos la clase de maridaje. En su comienzo, estaba en relación de dependencia con bodega, sin embargo, la repensamos para que esté en relación de dependencia, pero con la clase Vino. Además, volvimos a agregar las clases referidas a los tipos de vino (Tinto, Blanco, entre otros) ya que observamos que las etapas entre los mismos tipos eran en su mayoría las mismas.

Por último, se nos pidió agregar la categoría de "Blend", a lo que luego de una investigación, encontramos que con los vinos que habíamos incorporado en la bodega se hacían mayormente dos Blends conocidos, por lo que los agregamos, creando también una clase llamada "Blend" que heredaba de Vino y agrupaba a estos nuevos vinos.

3. Resultados

3.1) Cumplimiento de los requerimientos

Los requerimientos fueron los siguientes:

1. Modelar cómo sería su aproximación para brindar una solución que permita hacer funcionar el sistema de bodegas.
2. Generar una bodega con al menos 15 vinos diferentes.

3. Imprimir el estado actual de los diferentes vinos, en qué estado se encuentra y las características del mismo.
4. Imprimir con base en una lista de características del maridaje que vinos pueden ser las mejores opciones.
5. Implementar una interfaz en el diagrama de clases.
6. Debe existir al menos algún método polimórfico.

Los requisitos fueron cumplidos en su totalidad.
-Generamos 15 vinos distintos y además sumamos dos vinos “Blend”.
-Logramos a través de la clase controlDeInventario que imprima nuestro inventario, su etapa actual, así como otras funciones útiles.
-El usuario puede consultar el maridaje de los distintos vinos incluidos los “blend”.
-Implementamos la interfaz “Prints”, útil para imprimir las características de los vinos.
-Tenemos un método polimórfico, “cambiarDeEtapa(Vino)” que recibe un objeto Vino, el cual puede representar un objeto de las clases de vino que ofrecemos (Malbec,Syrah, etc). Luego, en base a esto se muestran y cambian las etapas dependiendo del tipo.

3.2) Planificación de actividades

La organización para desarrollar el proyecto comenzó tan pronto como las consignas fueron entregadas.
En un principio se comenzó con el modelado del diagrama de clases en el cual todos los integrantes del grupo tuvieron presencia y aportaron ideas y posibles cambios. Luego, con el diagrama ya encaminado a su versión casi final, se dividieron las clases y cada integrante codificó las clases que le fueron asignadas para después subirlas al repositorio de github.
Luego del tercer control con la confirmación del diagrama se decidió asignar tareas a cada integrante, referidas al informe; código; diagrama y presentación. Si bien todos participamos ayudando en cada parte, cada uno individualmente se dedicó a específicamente una parte de cada una de las anteriormente mencionadas.
Finalmente luego de la realización de estas tareas se hizo una puesta en común para la revisión grupal del trabajo de cada uno.

3.3) Cómo ejecutar el archivo

En el archivo .zip se encuentra, además de este mismo informe se incluye el diagrama de clases completo en mayor resolución y una carpeta llamada “SistemaBodega” en la cual se encuentran los archivos “SistemaBodega.jar” y “SistemaBodega.bat”.

Se puede ejecutar el programa de dos maneras:
> Haciendo doble clic en el archivo “SistemaBodega.bat”(opción solo para windows)
> Desde una terminal deberá acceder al directorio correspondiente y ejecutar el archivo .jar con el comando “java -jar SistemaBodega.jar”, cabe aclarar que se necesita tener Java 8, versiones anteriores podrían tener alguna incompatibilidad.

Una vez ejecutado el archivo se encontrará con el siguiente menú.

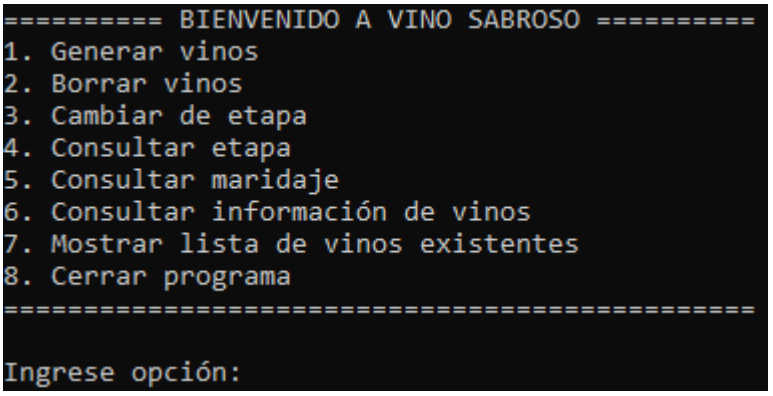

Figura 6: Menú principal.

3.3.1) En la opción de generar los vinos se encontrará otro submenú en el cual pedirá el tipo de vino, una vez seleccionado se pasa a elegir el vino, donde se pide también el número de lote y su etapa actual.

```
Seleccione el vino que desea generar:
========== Ingrese el tipo de vino: ==========
1. Tintos
2. Blancos
3. Rosados
4. Cavas
5. Blends
==============================================
Ingrese opción:
2

1. Chardonnay
2. Semillon
3. Viognier
4. Sauvignon Blanc
Ingrese opción:
3
Viognier
Ingrese lote:
123
123
Este es el listado de etapas de los vinos blancos:
```

```
========== LISTADO DE ETAPAS ==========
0. Recoleccion
 1. Transporte
 2. Reparticion
 3. Despalillado
 4. Desfangado
 5. TrasladoSolidos
 6. Maceracion
 7. FermentacionAlcoholica
 8. FermentacionVirgen
 9. Trasiego
 10. Clarificacion
 11. Estabilizacion
 12. Crianza
 13. Embotellado
 14. Venta
 ======================================
Por favor, seleccione la etapa de su vino
Ingrese índice:
```

Figura 7: ejemplo de generación de vinos.

3.3.2) En esta opción el usuario tendrá que luego de que se muestre los vinos en producción, seleccionar cual de estos quiere borrar del sistema.

3.3.3) En la sección de cambiar de etapa naturalmente se requiere que al menos un vino está en producción. Si se tiene vinos cargados

en el sistema se mostrarán por pantalla para que el usuario luego elija el vino a cambiar de etapa. Una vez elegido se le pedirá que con un menú (igual al de cuando se cargó el vino por primera vez) elija la etapa a la que quiere cambiar.

3.3.4) En el apartado de consultar etapa luego de que se muestre los vinos cargados en el sistema, se podrá indicar a cual vino se desea consultar la etapa actual.
Se le mostrara algo asi por pantalla:

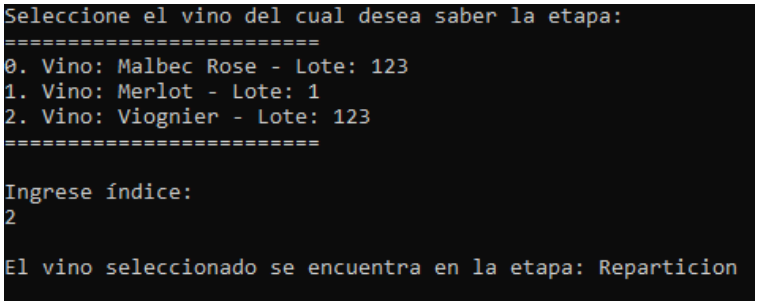

Figura 8: ejemplo de consulta de etapa

3.3.5) En Consultar maridaje el usuario podrá obtener el maridaje de los vinos que tenga cargados en el sistema.

3.3.6) Si se desea conocer más acerca de los vinos que se tienen en el sistema, Consultar información de vinos es la opción indicada, ya que mostrará características del vino que seleccione el usuario.

3.3.7) Para consultar los vinos que el usuario tiene cargados en el sistema,

3.3.8) Por último la opción de cerrar dará fin a la ejecución del programa.

4. Conclusiones

En conclusión pensamos que hemos realizado con éxito el desafío de crear un sistema de control para una bodega, aplicando los conocimientos de PDP, obviamente encontramos problemas/dificultades ya que el mayor reto fue tratar de modelar un

problema de la vida real como puede ser una bodega debido a que requiere una mayor inmersión en el terreno del problema y tratar de llevar esos conceptos al modelado del problema.