Memoria Escrita: *Práctica 1 - Grupo 9*
"Cárcel Apuestera"

- **Descripción general del dominio:**

El dominio de nuestra aplicación se centra en un sistema de información sobre una cárcel muy particular. El gobernador (*warden,* en inglés) de esta cárcel administra todos los aspectos sobre los prisioneros, guardianes, celdas, etc. Lo que hace especial a esta cárcel, es que regularmente se organizan eventos donde prisioneros pueden pelear entre sí, y los demás prisioneros y guardianes apuestan a un ganador. Todo este sistema de apuestas es administrado por el gobernador.

- **Descripción de las clases:**

La aplicación está diseñada para que el (único) gobernador de la cárcel la utilice, y pueda administrar tanto los aspectos normales (información sobre prisioneros, celdas y guardias), como las apuestas y peleas.
Cada prisionero es modelado como un objeto de la clase 'Prisionero', y de este se guarda su nombre, su identificación, su género, la fecha de inicio de su condena, la fecha de cuándo termina esta, la lista de todos los delitos con los que se condenó, la lista de las actividades especiales que ofrece la prisión de las que ha participado el prisionero en cuestión, y la celda en que reside. Estas actividades se conocen como "Antidelitos".

Cada delito se modeló como un objeto de la clase "Delito", y de este interesa guardar un identificador único del delito, el nombre con el que se conoce, una descripción, un nivel que indica la severidad del delito y el tiempo en meses que este delito y su severidad tienen como condena.

Como se mencionó previamente, la cárcel cuenta con una serie de actividades especiales que tienen como propósito preparar a los prisioneros para reintegrarse a la sociedad. Algunos ejemplos son: clase de yoga, de meditación, de pintura, etc. Estas actividades se conocen dentro de la cárcel como "antidelitos" y se modelan por medio de objetos de la clase "Antidelitos". De los antidelitos interesa guardar un identificador único del antidelito, el nombre con el que se conoce, una descripción y el tiempo en meses que este antidelito rebaja a la condena del prisionero.

Cada guardia se modela como un objeto de la clase 'Guardian', y de este nos interesa guardar su nombre, identificación, saldo, la lista de celdas que tiene encargado vigilar, y un registro de todos los traslados de celda que ha realizado (la funcionalidad del traslado de celdas se explica más adelantes en *funcionalidades*).

Cada celda se modela como un objeto de la clase 'Celda'. De estas interesa información como las dimensiones de la celda, el género de la celda, la capacidad máxima de prisioneros, y la lista de prisioneros que actualmente residen en ella.

El género de los prisioneros y las celdas se modeló por medio de una clase 'Enumeración', donde los valores de sus constantes son FEMENINO y MASCULINO.

Como se presentó en la descripción general del dominio, el gobernador administra peleas entre prisioneros, y tanto prisioneros como guardias pueden apostar a una pelea. Para modelar este sistema, se procedió de la siguiente manera:
Cada pelea se modeló como un objeto de la clase 'Pelea', y de estas interesa guardar un código de identificación única de cada pelea, el género de la pelea, ambos luchadores que conforman la pelea y el arma que usa cada uno de los luchadores. Una vez se termine una pelea, también interesa registrar quién fue el ganador.

Cada pelea que se registre en el sistema, está relacionada con un objeto de la clase 'Apuesta'. Estos objetos almacenan la siguiente información: un código de identificación único (que será el mismo código de la pelea con el que está relacionado) y una lista con todos los apostadores de esta pelea. Esta última lista debe almacenar información que permita identificar quién es el apostador, a quién está apostando y cuánto va a apostar. Esta clase va a almacenar además otros atributos que serán explicados detalladamente en *funcionalidades.*

Como prisioneros y guardias pueden apostar, se crea una superclase abstracta llamada 'Apostador' de la cual heredan las clases 'Prisionero' y 'Guardian'. Esta superclase va almacenar la información del nombre, identificación y saldo del apostador. Esta clase además cuenta con métodos que permiten manipular el saldo de cada apostador y obliga sus subclases a implementar el método toString().

- **Descripción de las relaciones y el diagrama UML:**

Los métodos de cada clase se explicarán con detenimiento en *funcionalidades.*
Las relaciones que modela el diagrama UML son las siguientes:

Un prisionero debe residir en una celda, pero una celda puede contener a varios prisioneros.
Los prisioneros residentes en una celda se guardan en un diccionario dentro de 'Celda' (atributo de instancia), donde la llave es la identificación única del prisionero que se guarda como valor.
Los delitos y antidelitos de un prisionero se guardan en 'Prisionero' como diccionarios (atributos de instancia), donde las llaves son los códigos de los objetos guardados como valores, respectivamente.
Un prisionero debe tener al menos un delito para ser encarcelado, y puede tener varios o no tener anti-delitos registrados.

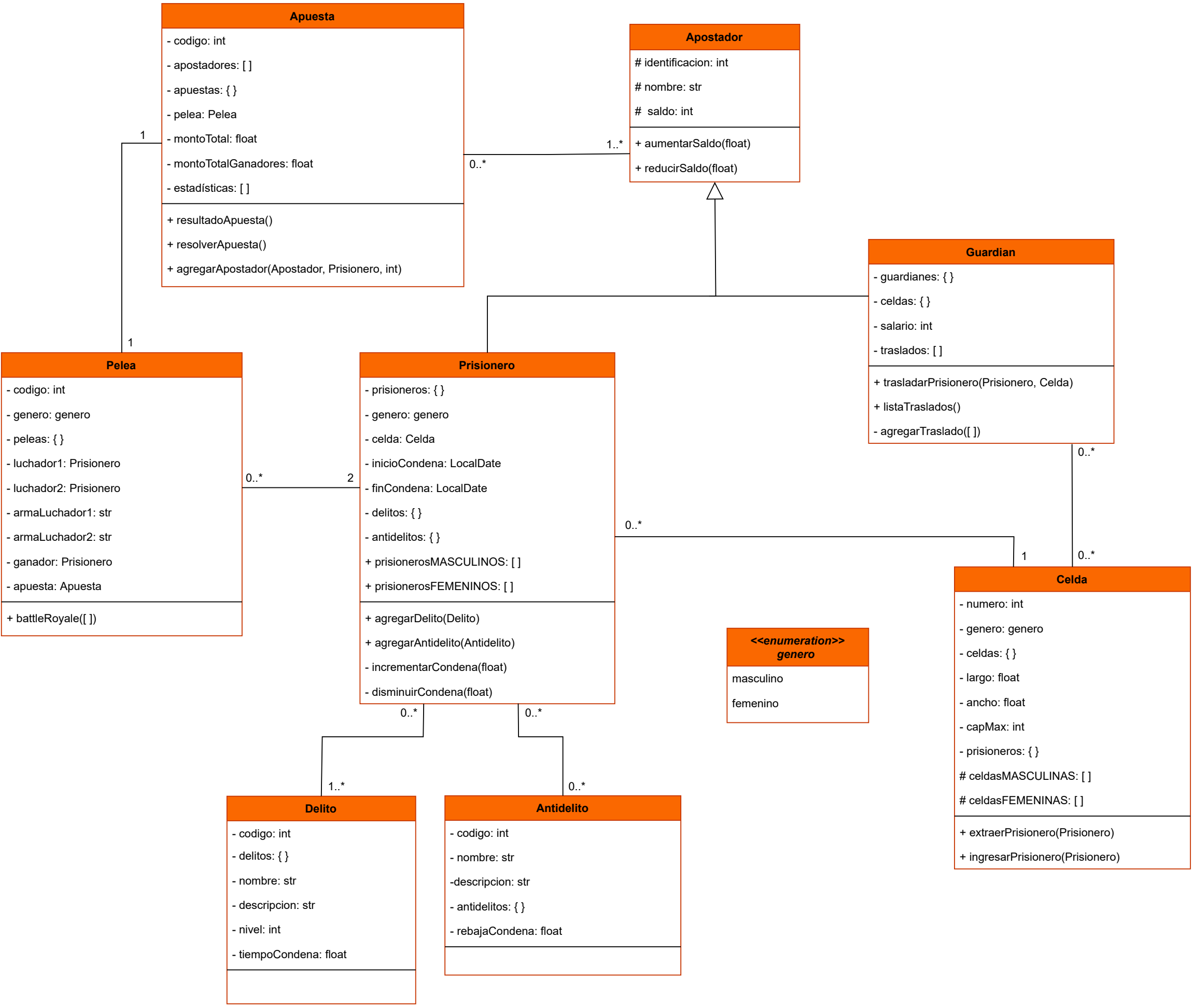
Un guardián puede tener varias celdas bajo su vigilancia, y estas se guardan en el atributo *celdas* de clase diccionario dentro de 'Guardian' (atributos de instancia) donde la llave es el respectivo número de la celda que se guarda como valor. Dos guardianes diferentes pueden vigilar la misma celda.
El atributo de instancia *traslados* de 'Guardian' almacena todos los traslados que el guardián ha realizado por medio de un arreglo, donde cada elemento del arreglo almacena a su vez otro arreglo estático donde se guarda la celda de origen, el prisionero y la celda de destino.

'Prisionero' y 'Guardian' heredan de la clase Apostador.

Por otro lado, una Pelea está conformada por dos prisioneros, los cuales corresponden a los atributos *luchador1* y *luchador2.* Una vez se conoce el resultado de una pelea, el atributo *ganador* es inicializado con el prisionero ganador.
Un mismo prisionero puede haber participado en varias peleas.
Cada instancia de 'Pelea' está relacionada con una instancia de 'Apuesta', y viceversa. La Apuesta de una Pelea puede estar relacionada con varios apostadores. Esta información se guarda en el atributo de instancia *apostadores* dentro de Apuesta por medio de un arreglo, donde cada elemento del arreglo almacena a su vez otro arreglo donde se guarda el apostador, el prisionero por el que apuesta y la cantidad de dinero que apuesta.

El atributo estático de la clase 'Prisioneros' *prisioneros* de tipo diccionario guarda todos los prisioneros que están registrados en el sistema, donde la llave es la identificación del prisionero guardado como valor.
De igual forma, los atributos estáticos *delitos, antidelitos, celdas, guardianes, peleas* y *apuestas* de clase diccionario guardan todos los objetos de las clases Delito, Antidelito, Celda, Guardian, Pelea y Apuesta respectivamente que han sido creados en el sistema.

Carcel Apuestera

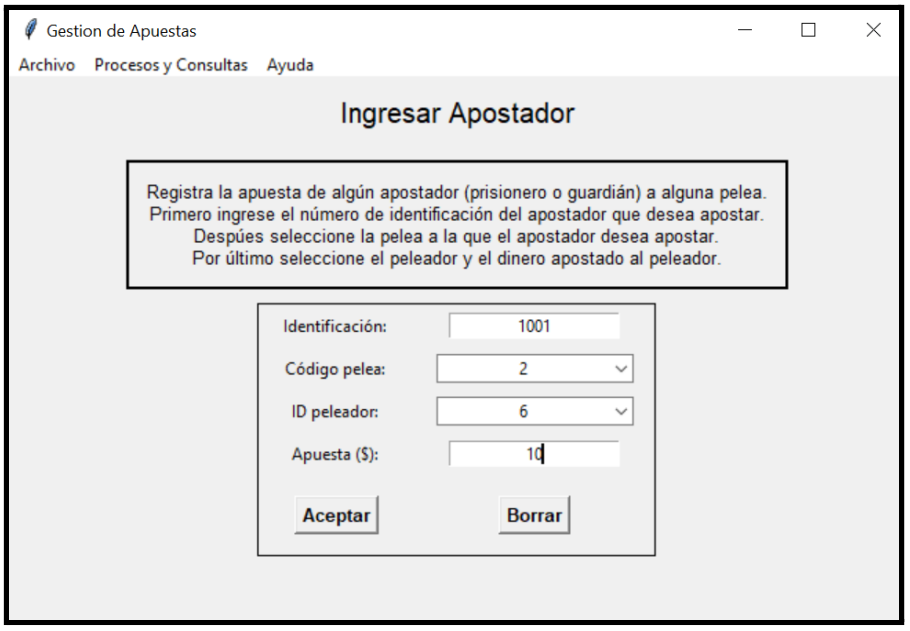


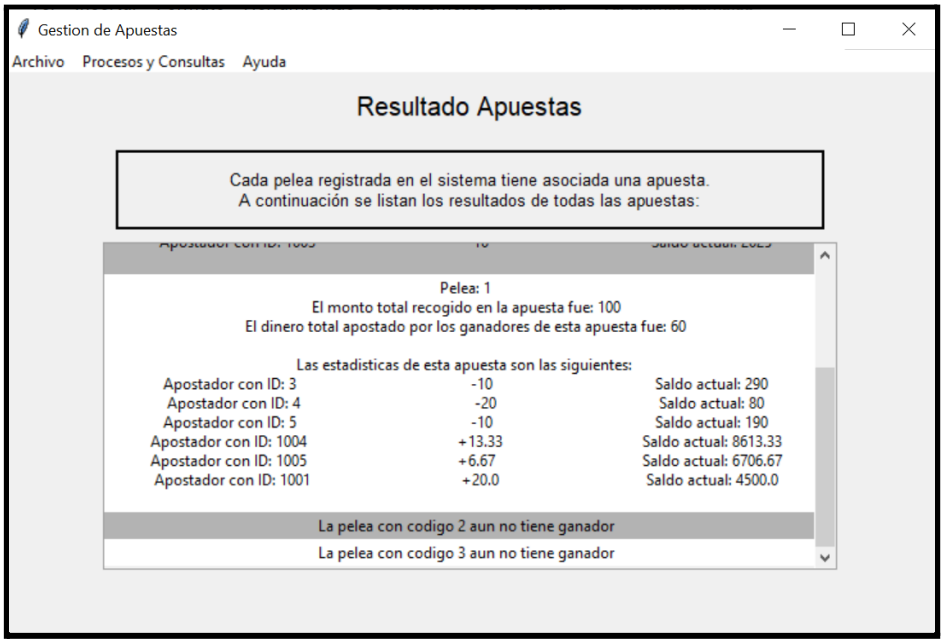
- **Descripción de funcionalidades:**

1. Resolver una apuesta: El objetivo final de esta funcionalidad es que a cada apostador ganador de una pelea se le consigne a su saldo la ganancia respectiva. El flujo de esta funcionalidad es como sigue:
El gobernador registra una pelea nueva en el sistema con toda la información pertinente. Para efectos de la ilustración, llamemos a esta pelea Pelea1.
Automáticamente se crea un objeto de la clase Apuesta relacionado a esta pelea. Llamemos a este objeto Apuesta1.
Si un Apostador (Prisionero o Guardian) desea apostar a Pelea1, debe comunicárselo al gobernador, para que éste lo registre. Este proceso se lleva a cabo usando el método Apuesta1.*agregarApostador*(Apostador apostador, Prisionero prisionero, Integer apuesta), el cual se encarga de agregar 'apostador' a la lista de apostadores del objeto Apuesta1, junto con el 'prisionero' luchador al cual le apuesta y el monto que 'apuesta' . Este método también es el que se encarga de descontar del saldo del Apostador el dinero que se apostó.
Se pueden registrar muchos Apostadores a Apuesta1.
Cuando la pelea haya terminado, el gobernador se encarga de registrar el ganador implementando el método de instancia Pelea1.*setGanador*(Prisionero prisionero). Este método, a su vez, llama al método Apuesta1.*resolverApuesta*().
Este último método lleva a cabo varias acciones:

- Calcular el monto total apostado por todos los Apostadores e inicializa el atributo Apuesta1.*montoTotal* con esto.
- Calcular el monto total apostado por todos los Apostadores que ganaron la apuesta e inicializa el atributo Apuesta1.*montoTotalGanadores* con esto.
- Entre los Apostadores ganadores de Apuesta1 se reparte Apuesta1.*montoTotal* de modo que la ganancia de cada quién sea proporcional al dinero que apostó. Para aumentar el saldo de cada ganador se usa la función de instancia de la clase Apostador *aumentarSaldo*(double dinero).
- Por cada Apostador, se registra en un Strings cuánto ganó (+) o perdió (-) y el saldo actual. Cada String se agrega a un ArrayList<String> y finalmente se inicializa Apuesta1.*estadisticas* con este Array.

Finalmente, se imprime por pantalla lo que devuelve la función Apuesta1.*resultadoApuesta()*. Este método hace uso los atributos *montoTotal, montoTotalGanadores* y *estadisticas* de Apuesta1.

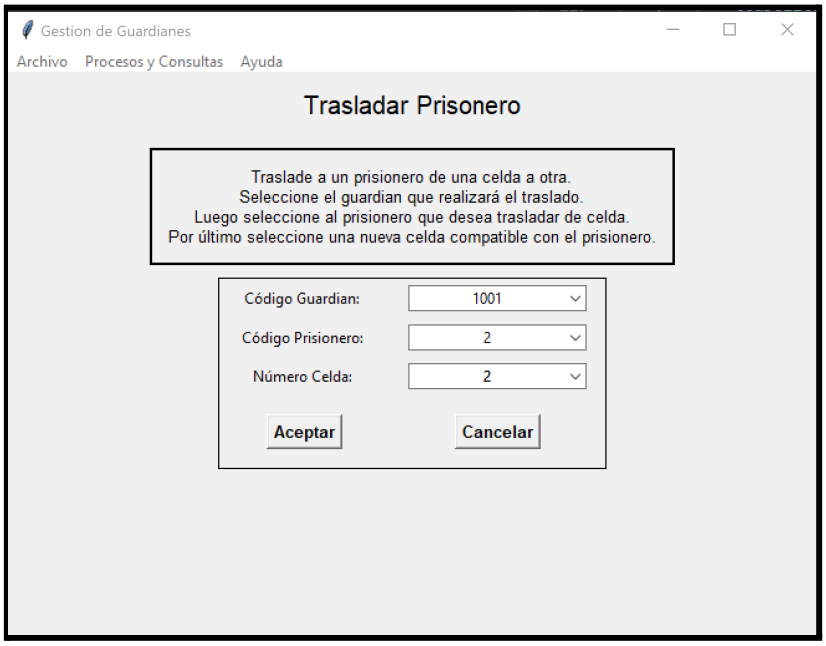




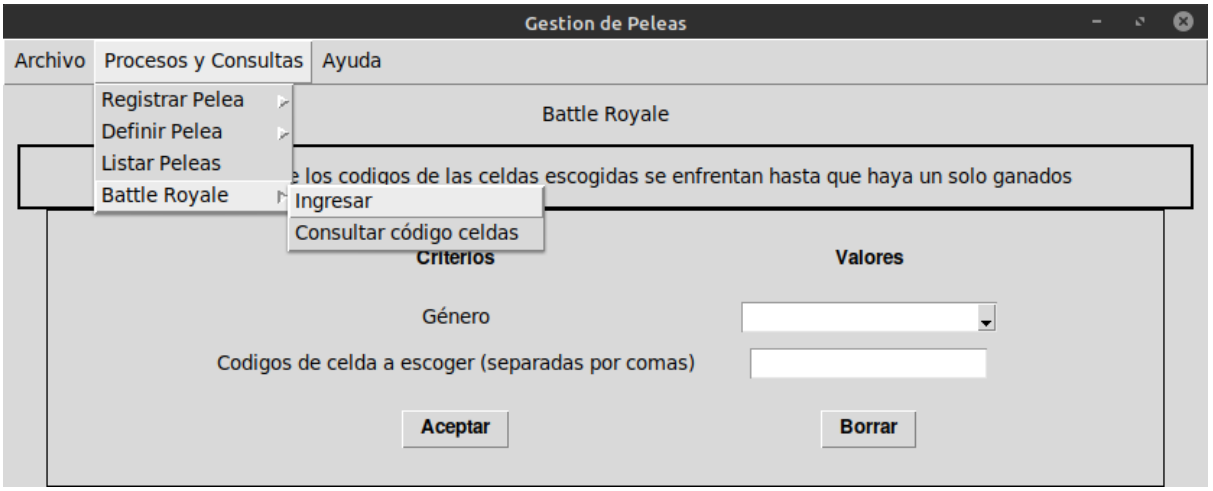
2. Trasladar un prisionero de una celda a otra: Esta funcionalidad recrea el traslado de un prisionero que se encuentra en una celda a otra celda por parte de un guardia seleccionado.
Requiere la participación de tres clases: 'Guardian', 'Prisionero' y 'Celda'.
Se implementa de la siguiente manera: Se selecciona el prisionero el cual se quiere trasladar y la celda nueva a la que se desea trasladar, luego, el encargado de realizar esta tarea es un guardia el cual también puede ser elegido. La selección de las partes involucradas las hace el gobernador (Administrador) por medio de la UI.
La funcionalidad hace uso de los siguientes métodos:

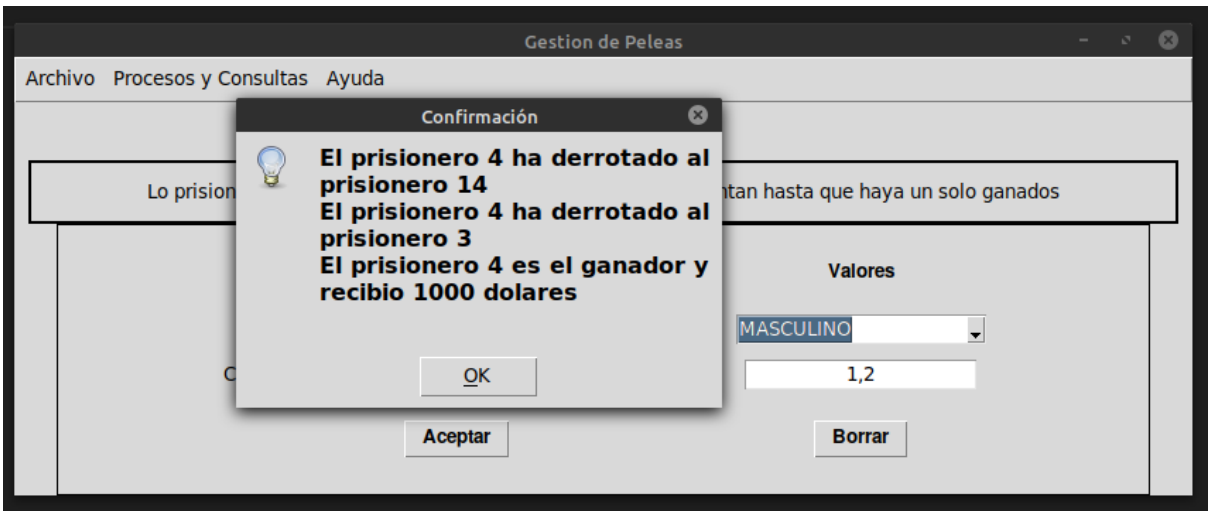
- **trasladarPrisionero(prisionero: Prisionero, celda: celda):** que se encuentra en la clase 'Guardian', la cual recibe como parámetros el prisionero a trasladar y la celda donde va a ser trasladado. El guardián verifica en qué celda se encuentra el prisionero y lo extrae de la lista de prisioneros que la celda en cuestión posee (extraerPrisionero(prisionero: Prisionero)), es decir, lo extrae de la celda en la que se encuentra. Luego, toma la nueva celda a la que se desea trasladar el prisionero y lo agrega a la lista de prisioneros que esta nueva celda posee (ingresarPrisionero(prisionero: Prisionero)). Finalmente, el registro del traslado realizado se almacena en una lista que cada guardia posee (traslados).
- **extraerPrisionero(prisionero: Prisionero):** Este método se encuentra en la clase 'Celda'. La clase 'Celda' posee un atributo el cual permite identificar que prisioneros residen en ella (**prisioneros**). Cumple la tarea de borrar un prisionero a **prisioneros**, es decir, se extrae un prisionero de la celda.
- **ingresarPrisionero(prisionero: Prisionero):** Este método se encuentra en la clase 'Celda'. La clase celda posee un atributo el cual permite identificar que prisioneros residen en ella (**prisioneros**). Cumple la tarea de ingresar un prisionero a **prisioneros**, es decir, se ingresa un nuevo prisionero a la celda.
- **agregarTraslado(objetos):** Recibe como parámetro una lista de tamaño 3 de diferentes tipos de objetos los cuales se encuentran en el siguiente orden:
  - El primer elemento representa la celda de origen del prisionero.
  - El segundo elemento representa el prisionero que quiere ser trasladado.
  - El tercer elemento representa la nueva celda a la cual se quiere trasladar al prisionero.

  Toda esta información se almacena en un atributo que posee 'Guardian' el cual está encargado de almacenar los traslados que un guardia ha realizado (**traslados**). Este método agrega dicho traslado realizado a **traslados**.
- En la UI, se comprueba si la celda a la que se va a trasladar un prisionero es compatible con su género y si posee espacio en ella.

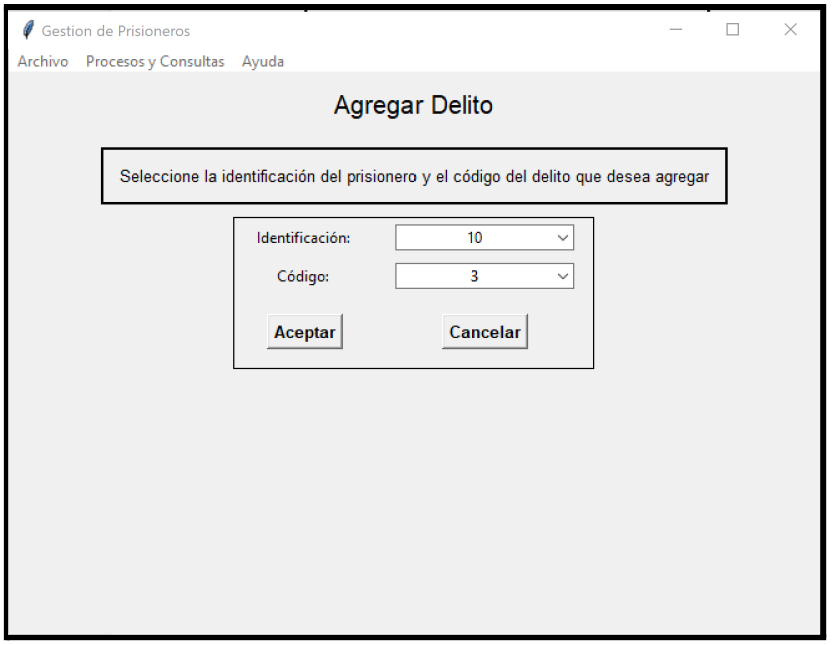


3. Battle Royale: Esta funcionalidad simula varias peleas entre todos los prisioneros de unas celdas en específico. Para esto, el gobernador ingresa los códigos de las celdas elegidas separados por coma y la identificación de cada prisionero perteneciente a estas celdas se agrega a otra lista llamada luchadores. Luego se eligen dos prisioneros aleatoriamente, se eliminan de la lista y se enfrentan en combate, el ganador será agregado de nuevo a la lista de luchadores. Se agregará la narración de cada pelea a la lista combates hasta que solo quede un ganador; después de la pelea final se agregarán la identificación del ganador y el monto recibido. El ganador final y la lista combates serán agregados al resultado que será el retorno de esta funcionalidad, además el prisionero ganador tendrá un premio de US 1000, el cual se adicionará al saldo de este. Esta funcionalidad es un método estático de la clase Pelea y emplea las clases Prisionero y Celda, y los métodos getPrisioneros() y aumentarSaldo().



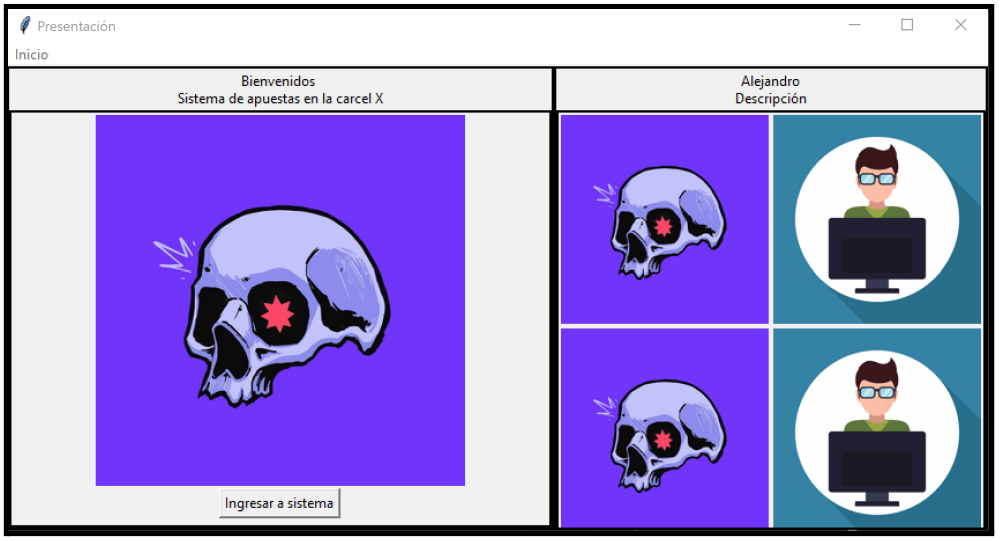


4. Incrementar Condena: Esta funcionalidad permite, a la hora de ingresar un nuevo prisionero, escoger todos los delitos que este cometió y calcular la fecha en la que termina su condena. También, si un prisionero cometió un delito mientras se encuentra en la prisión, se le condena por este delito y su condena se extiende automáticamente. En el constructor de Prisionero se recorre la Hashtable delitos, se suma el tiempo de condena de cada delito, se establece la fecha de inicio de la condena y fin condena con el método incrementarCondena(); al agregar un nuevo delito,el gobernador ingresará el delito cometido, este se agregará a la Hashtable delitos del prisionero y se enviará el tiempo de condena de este al método incrementarCondena(), cambiando automáticamente la fecha final de la condena. En esta funcionalidad interactúan las clases Prisionero y Delito.

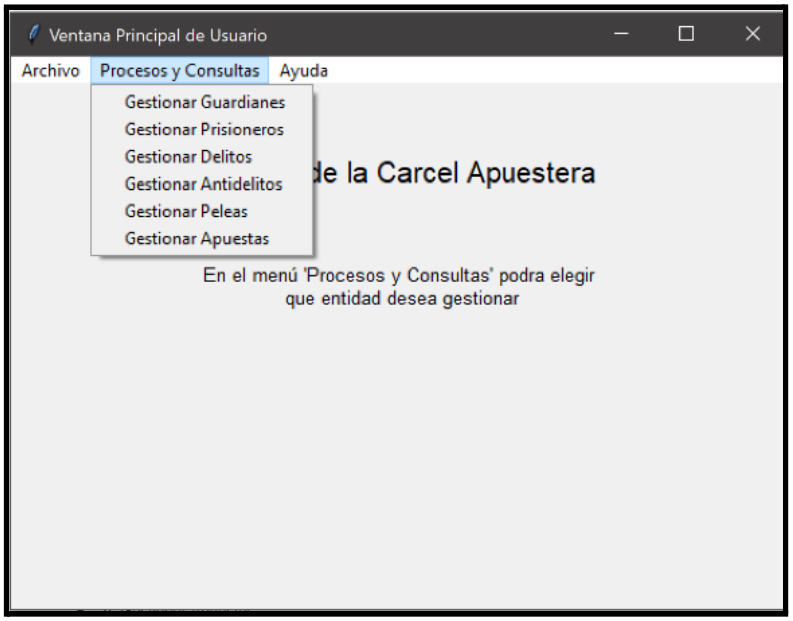


5. Disminuir Condena: Esta funcionalidad permite recalcular la fecha en la que termina la condena del prisionero. Si un prisionero realiza un antidelito mientras se encuentra en la prisión, su condena se rebajará automáticamente, el gobernador ingresará el antidelito en el método agregarAntidelito(), este se agregará a la Hashtable antidelitos del prisionero y se enviará el tiempo de disminución de condena de este al método disminuirCondena(), cambiando automáticamente la fecha final de la condena. En esta funcionalidad interactúan las clases Prisionero y Antidelito.

- **Manual de Usuario:**

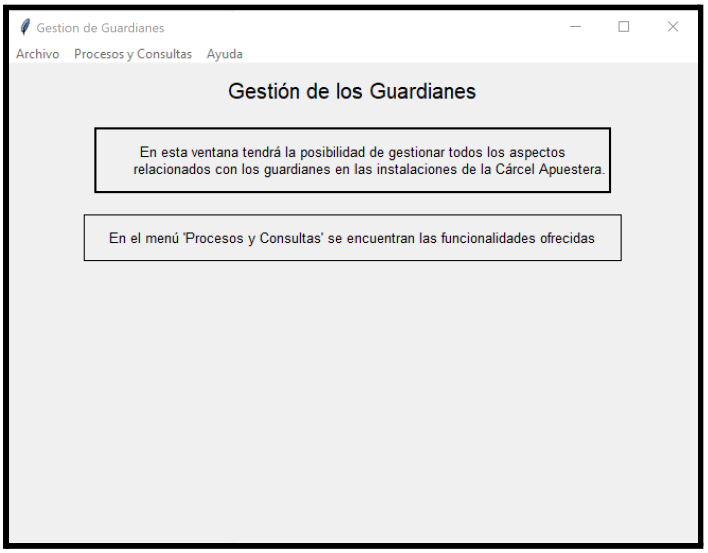


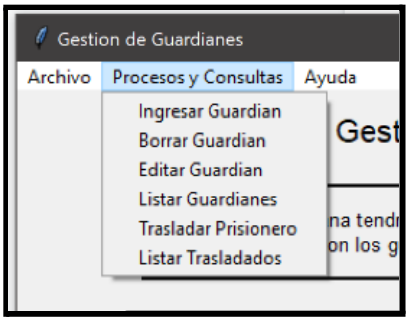
El sistema es de uso abierto. Al ejecutar la aplicación, encontrará una ventana de bienvenida. Presione 'Ingresar a sistema' para dirigirse al menú principal:



En el menú de 'Procesos y Consultas' encontrará las diferentes entidades que se pueden gestionar:

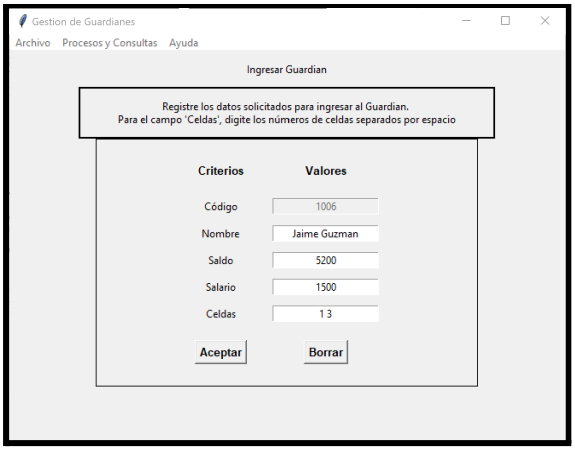
Gestionar Guardianes:

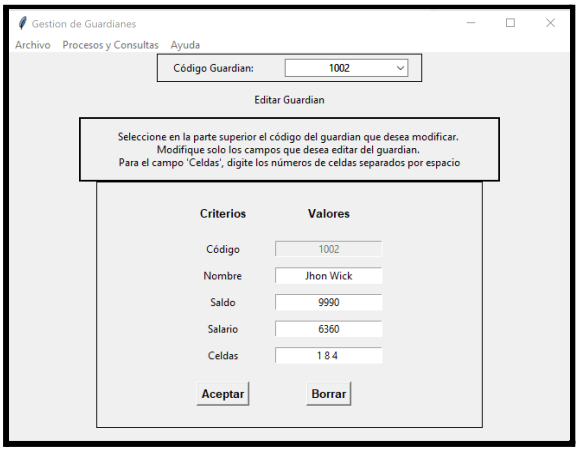


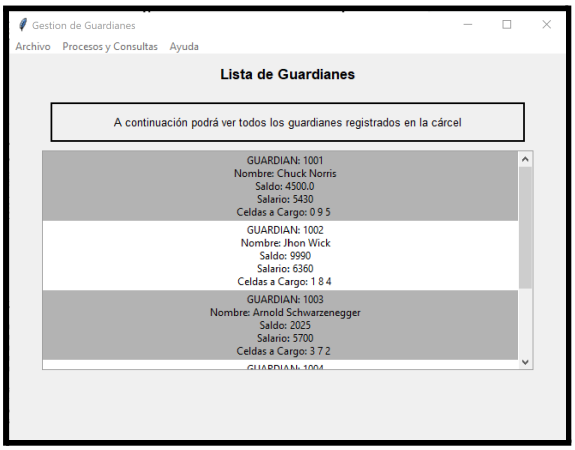


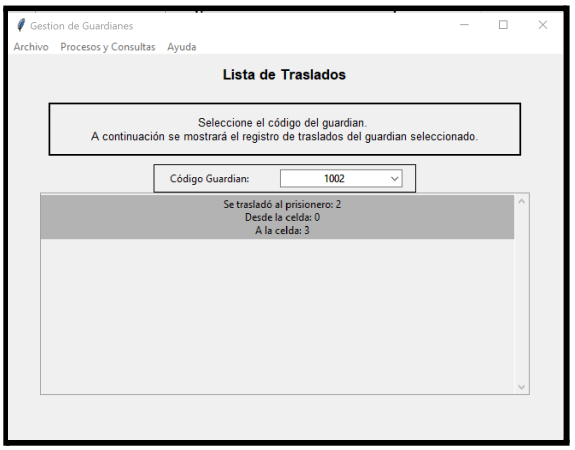
Seleccione la opción que desea ejecutar dentro de Guardianes. Rellene los campos necesarios y oprima aceptar para ejecutar la acción seleccionada.

A continuación se muestran los formularios de cada opción:

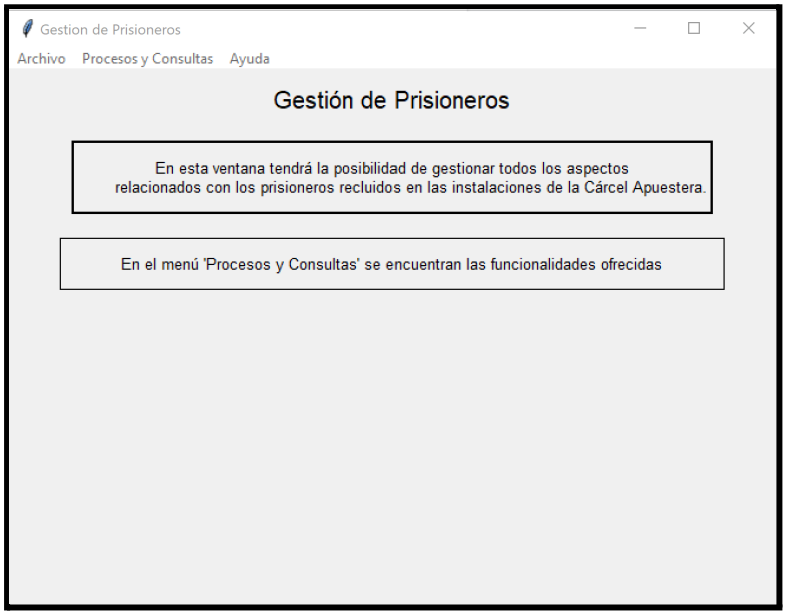


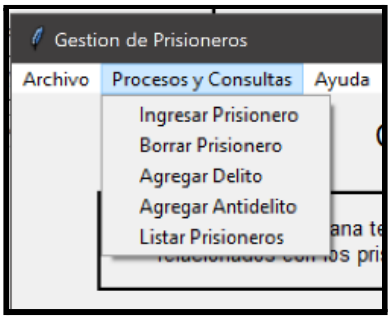




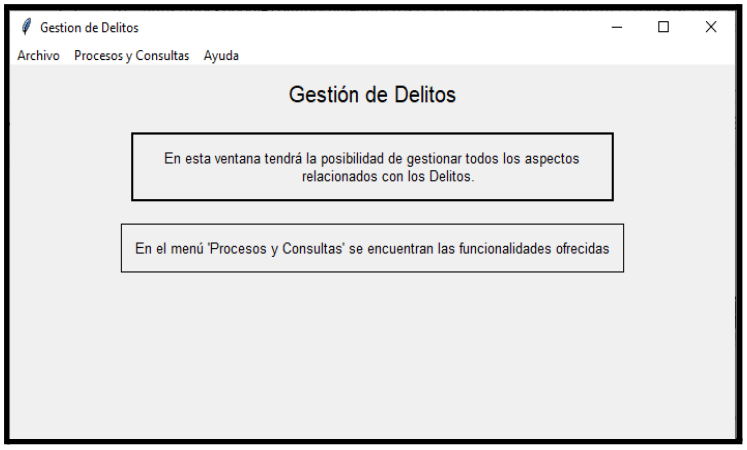


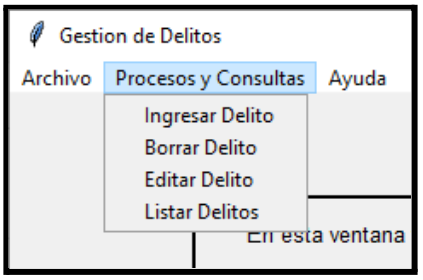
Gestionar Prisioneros:





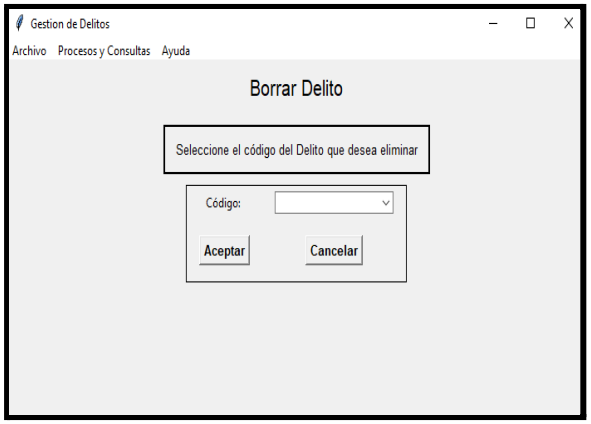
Gestionar Delitos:

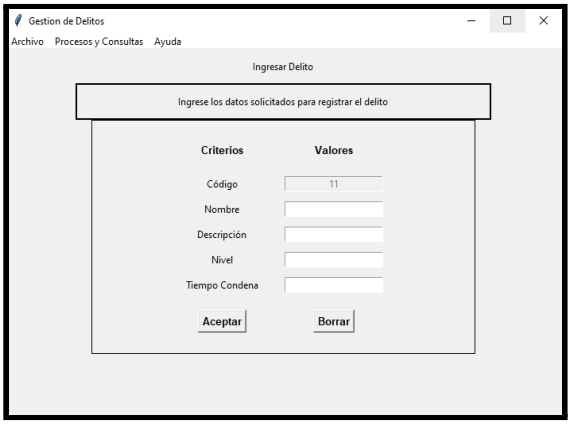


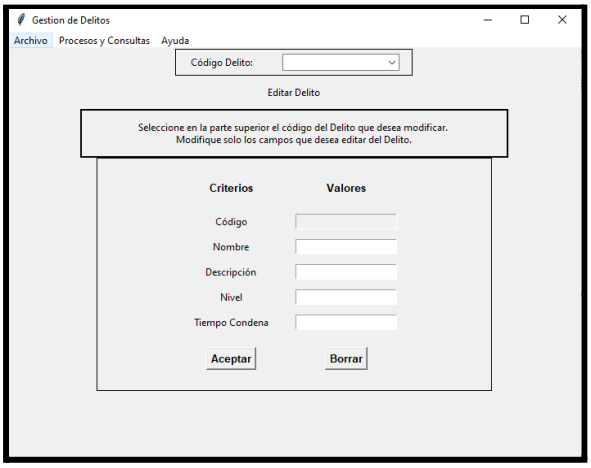


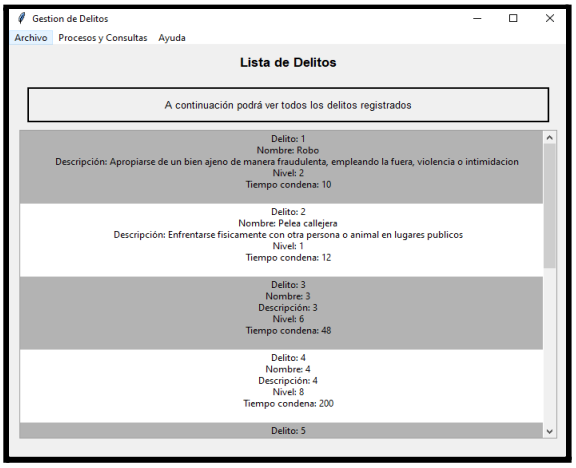
Seleccione la acción que desea realizar. Rellene los campos necesarios y oprima el botón Aceptar.

A continuación se muestran los formularios para cada opción:

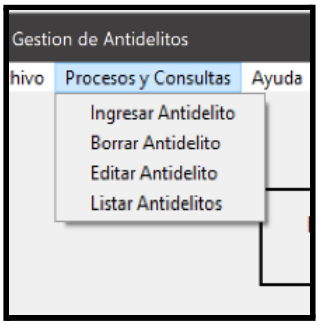






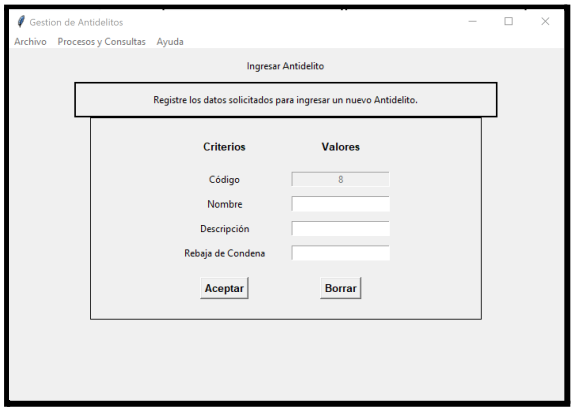


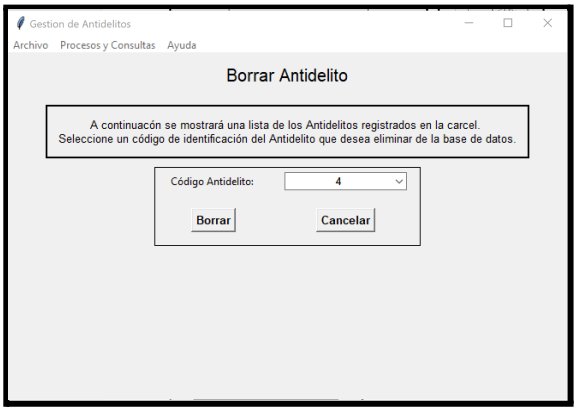
Gestionar Antidelitos:

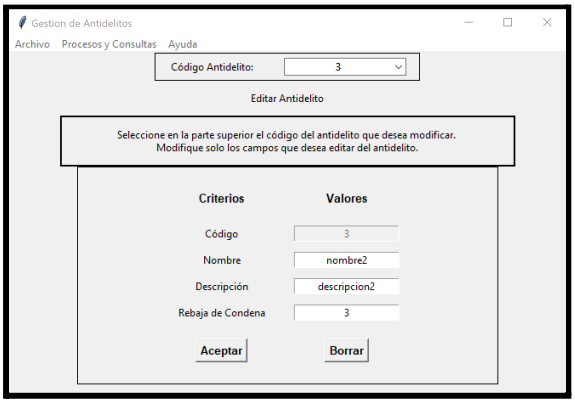


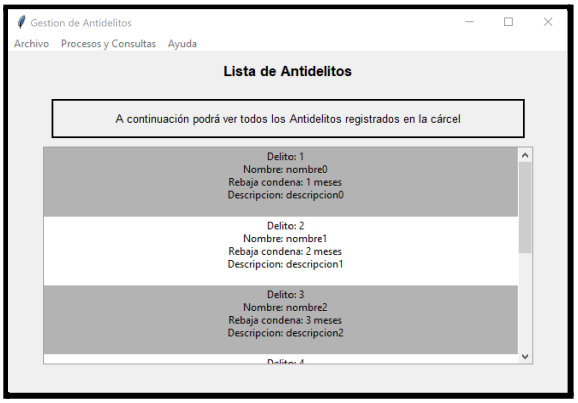
Seleccione la opción que desea ejecutar dentro de Antidelito. Rellene los campos necesarios y oprima aceptar para ejecutar la acción seleccionada.

A continuación se muestran los formularios de cada opción:

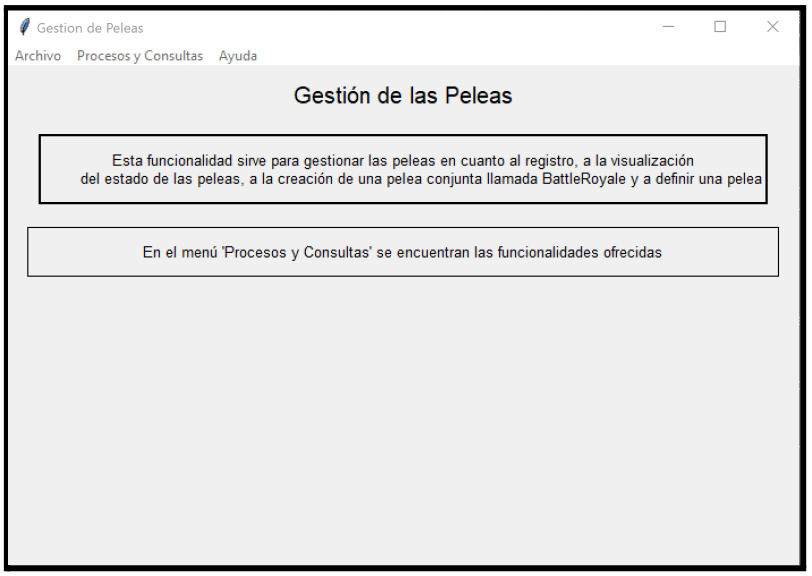


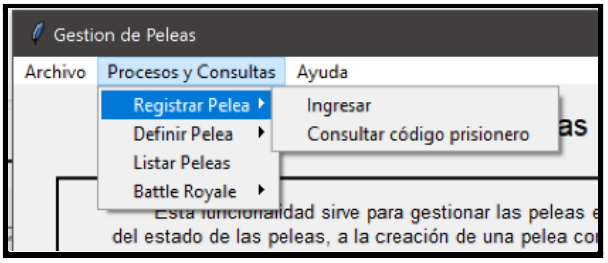






Gestionar Peleas:
Esta sección permite registrar y modificar aspectos de las peleas creadas.
Posee la función de Battle Royale en donde se enfrentan los prisioneros de las celdas seleccionadas por un premio.





Gestionar Apuestas:
Esta sección permite registrar la apuesta de algún apostador y consultar los resultados de apuestas ya resueltas o aún por resolver: