

De vuelta a los clásicos



Alumno: Miguel Fernández Huerta

Centro: Escuela de ingeniería informática del software (uniovi)

DNI: 34313073-V

UO: 287577

Índice

1. Introducción	(pág 3-4)
2. Desarrollo	(pág 5-32)
2.1. Fase de diseño	(pág 5-9)
2.2. Fase de implementación	(pág 10-32)
2.2.1. Lógica	(pág 10-16)
2.2.2. Interfaz	(pág 17-25)
2.2.3. Pruebas	(pág 26-32)

1. Introducción

Se trata de elaborar un juego de estrategia que será instalado en una máquina de una tienda de videojuegos con el fin de darles a los clientes la posibilidad de jugar al juego si disponen de un ticket de compra de la tienda con un importe igual o superior a los 20€.

Dicho ticket tiene dos códigos a introducir en la máquina, es el código de la tienda y el código del ticket.

Una vez validada la información del ticket introducida en la máquina, aparecerán una serie de pantallas a medida que el usuario vaya avanzando en ellas antes de llegar a la pantalla donde comenzará la partida.

Las primeras pantallas con las que el usuario se encontrará, son destinadas a introducir la información acerca del usuario y del ticket junto con mostrar un texto con las reglas del juego. Estas pantallas componen la primera clase interfaz de la aplicación.

Las posteriores pantallas serán las del juego y una pantalla de canje de premios accesible para el usuario si éste logra terminar el juego con una puntuación final superior a 0 puntos.

La aplicación cuenta con dos clases de lógica, una encargada de toda la lógica del juego (clase Juego) y otra encargada con el resto de funcionalidad de la aplicación (clase Service). La clase Service es la clase principal de la aplicación ya que es la que contiene los métodos de validación de tickets, de inicialización del juego, de modelo de premios, etc.

Las clases de la interfaz de usuario cuentan con clases internas receptoras de eventos que, mediante enlaces con ciertos componentes de las interfaces, permiten realizar determinados eventos en la ejecución de la aplicación sin gastar excesivos recursos en las creaciones de dichas clases internas como haría WindowBuilder. Además, esto nos permite reutilizar ciertos métodos usados en esas clases internas para la construcción de nuevos métodos o incluso para que varios componentes tengan el mismo objeto receptor de un evento común entre ellos (Esto implica una optimización en el uso de recursos).

Aunque en el Wireframe V2 habían pantallas en las que se podría cambiar el idioma de la aplicación (Internacionalización), he optado por no internacionalizar la aplicación porque no estaba muy seguro de cómo iba a ser el resultado final de la aplicación sumado a que no era algo que dominase por completo.

En el Wireframe V2 también se podían ver que en ciertas pantallas se activaban botones en función de si ciertos campos de texto no eran vacíos. Normalmente, en clase, era costumbre hacer un chequeo de los campos de texto y si estaban vacíos pues que saltase un aviso al usuario que indicase que los campos debían ser rellenados.

No me pareció mala idea pero quise hacerlo de forma que quedase igual que como se indicó en el Wireframe, por ende, cree una clase interna (clase `ProcesaCamposDeTexto` en la clase `VentanaInicio`) receptora de un evento de tipo `KeyListener` que extendiera la clase `KeyAdapter` para no tener que reescribir cada método de la interfaz `KeyListener`.

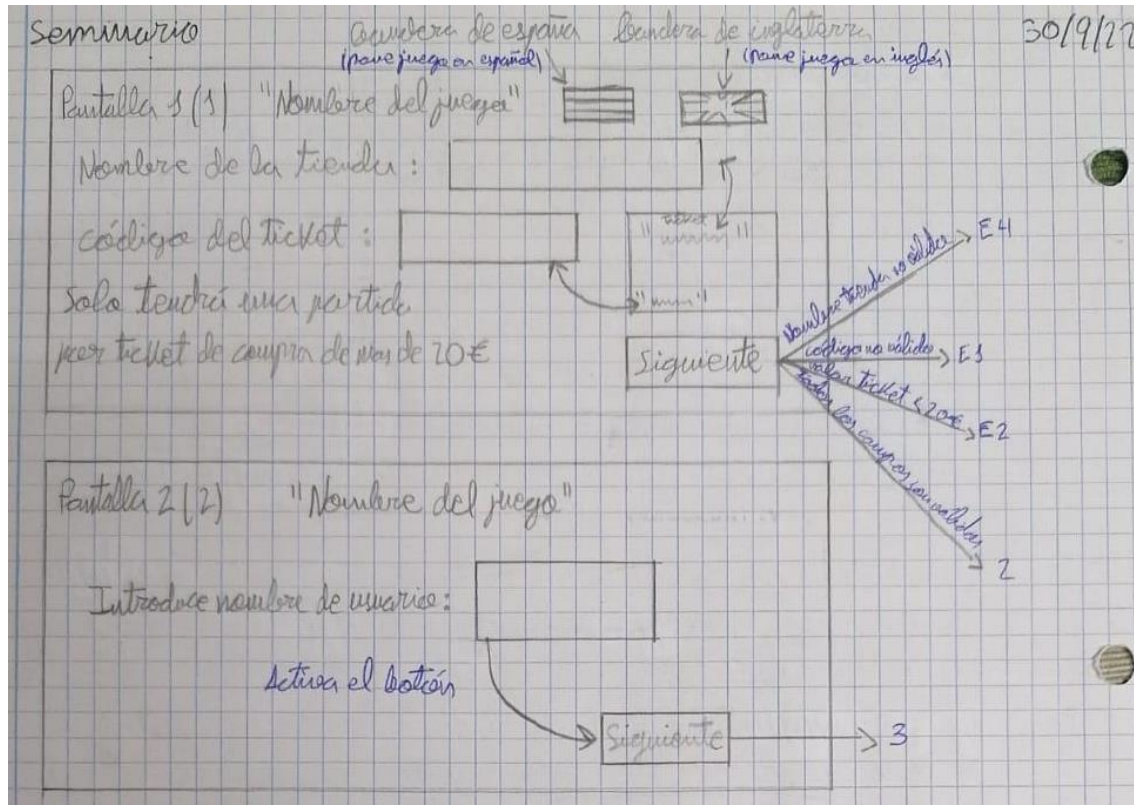
El método que nos permitiría activar el botón siguiente al tener los campos de texto no vacíos sería el `keyReleased`.

Eso sería lo más relevante de las clases de la interfaz de usuario de la aplicación.

En cuanto al resto de clases, se ha requerido la creación de dos clases que extendieran las clases de ciertos componentes de las interfaces de usuario con el fin de facilitar la implementación de la funcionalidad del juego en la aplicación. Dichas clases son la `MyJButton` (que extiende la clase `JButton`) y `MyJSpinner` (que extiende la clase `JSpinner`).

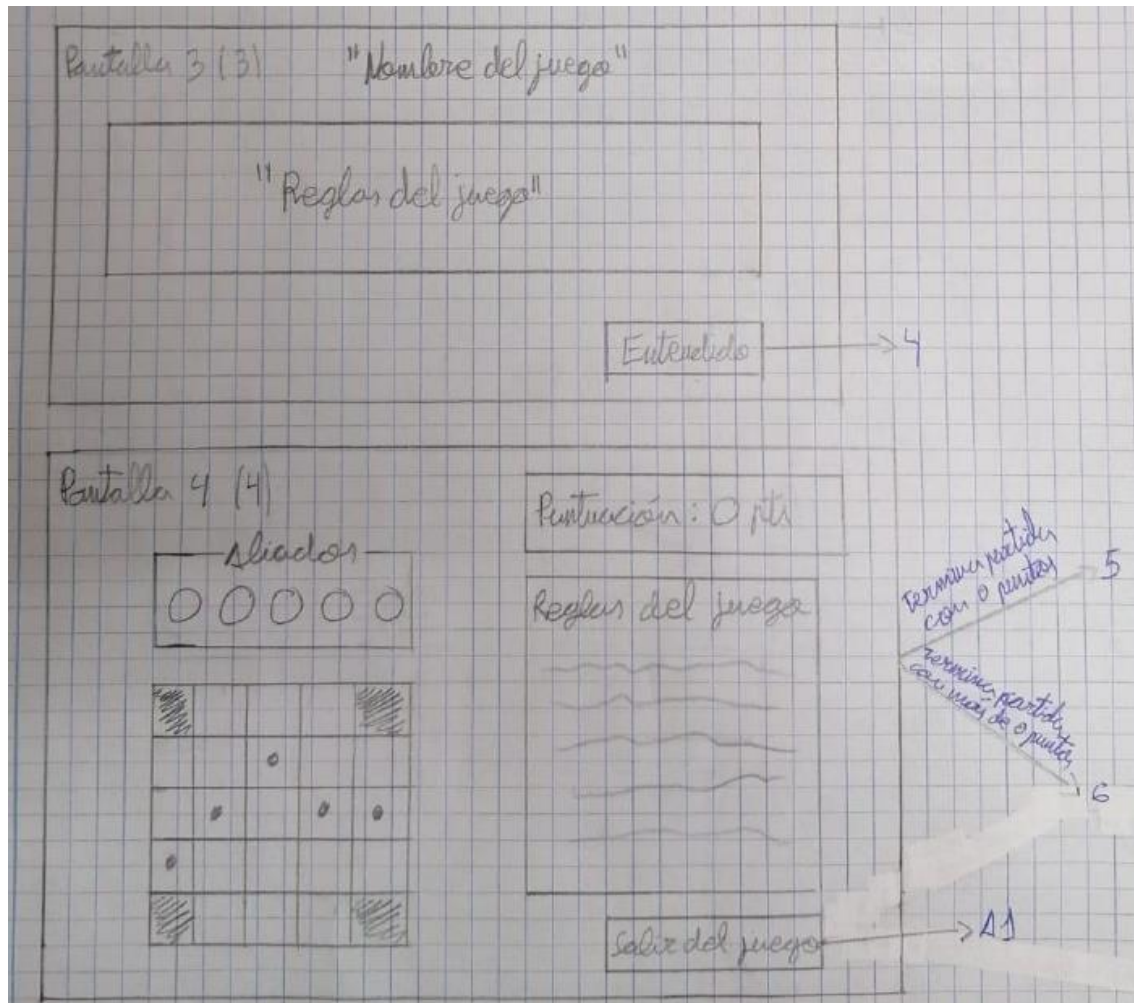
2. Desarrollo

2.1 Fase de diseño

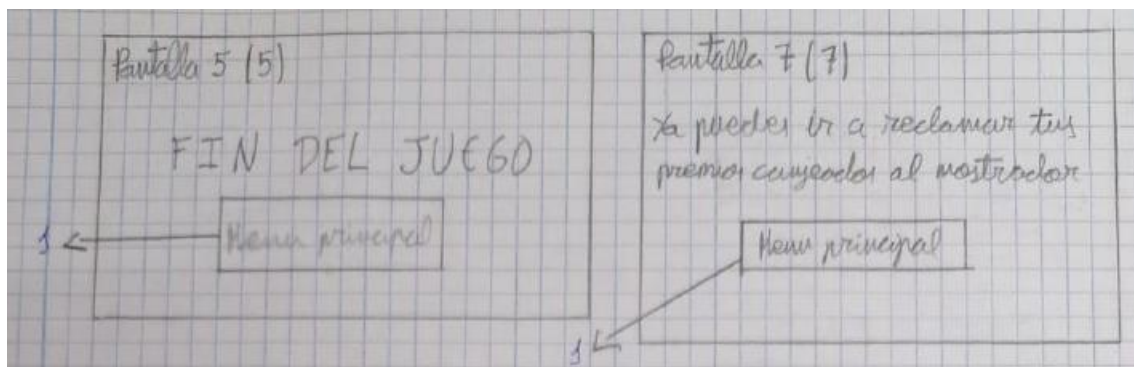


En la pantalla 1 lo únicos cambios que se han realizado son que se eliminaron los botones de la parte superior derecha que permitían cambiar el idioma de la aplicación, como no se ha optado por la internacionalización, estas opciones no eran necesarias.

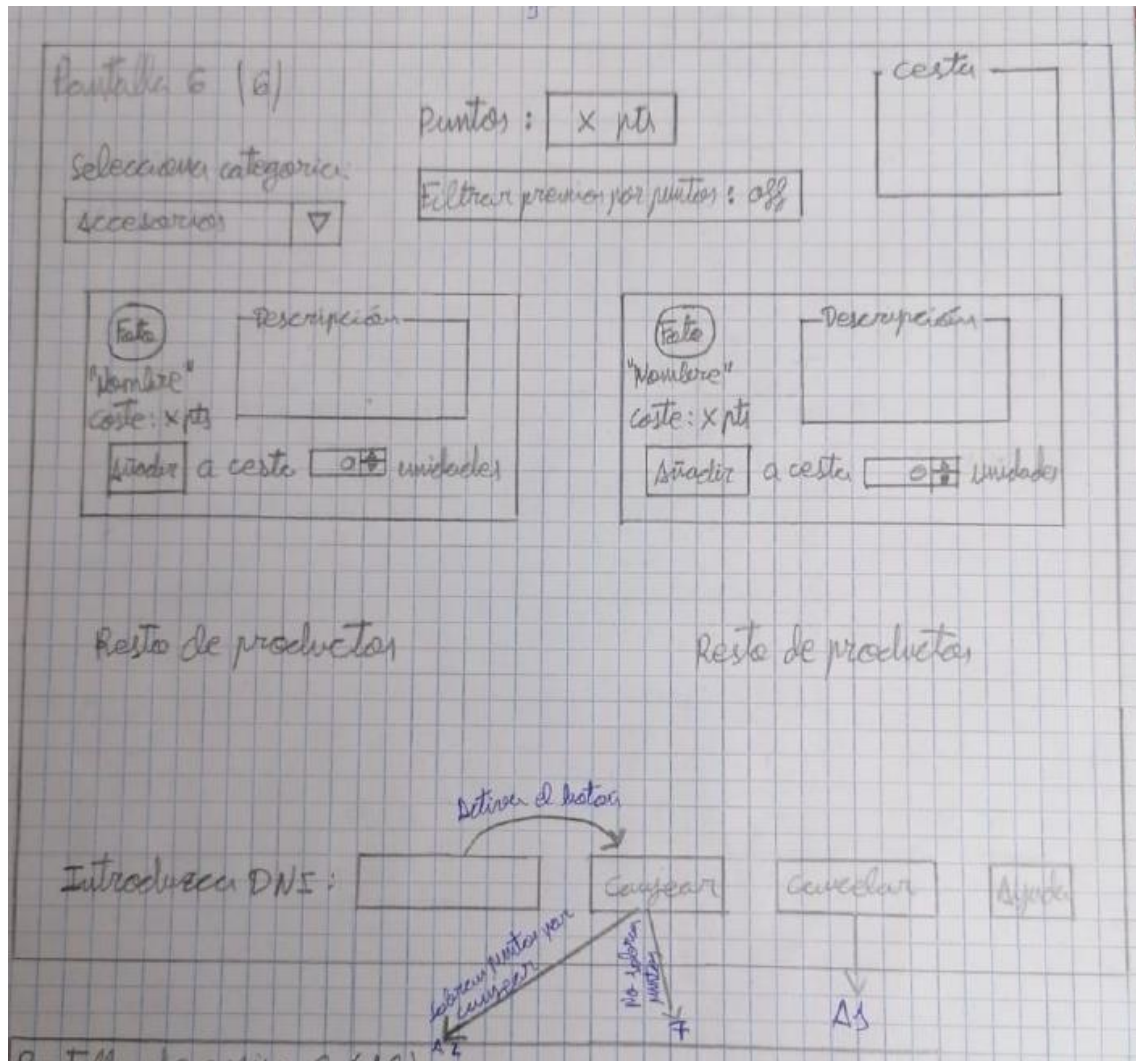
En la misma pantalla también se cambiaron los textos que se sitúan por delante de los campos de texto de introducir el código de la tienda y el del ticket para adecuarse mejor a los nombres de los datos que el usuario debía introducir en la aplicación.



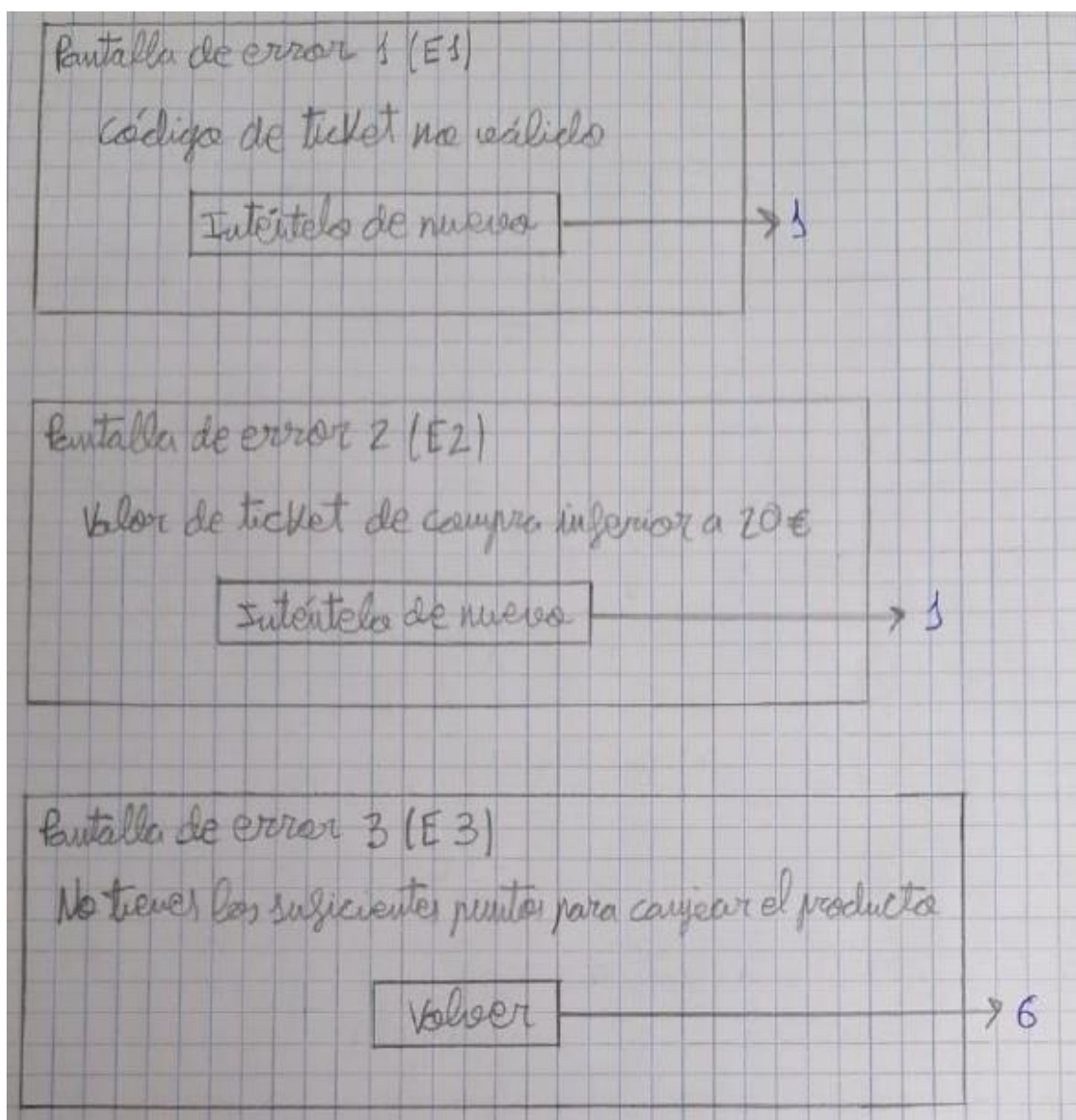
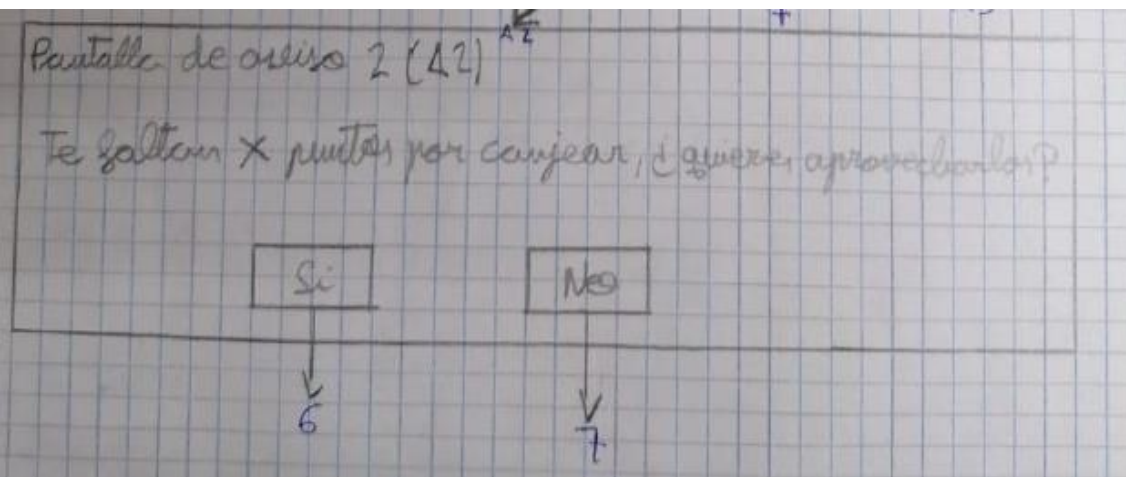
En la pantalla 4 se ha agregado un campo de área de texto que representa una leyenda con las iniciales de los invasores del tablero. También se agregó una etiqueta y un campo de texto para mostrar los turnos en los que se van desarrollando las partidas.

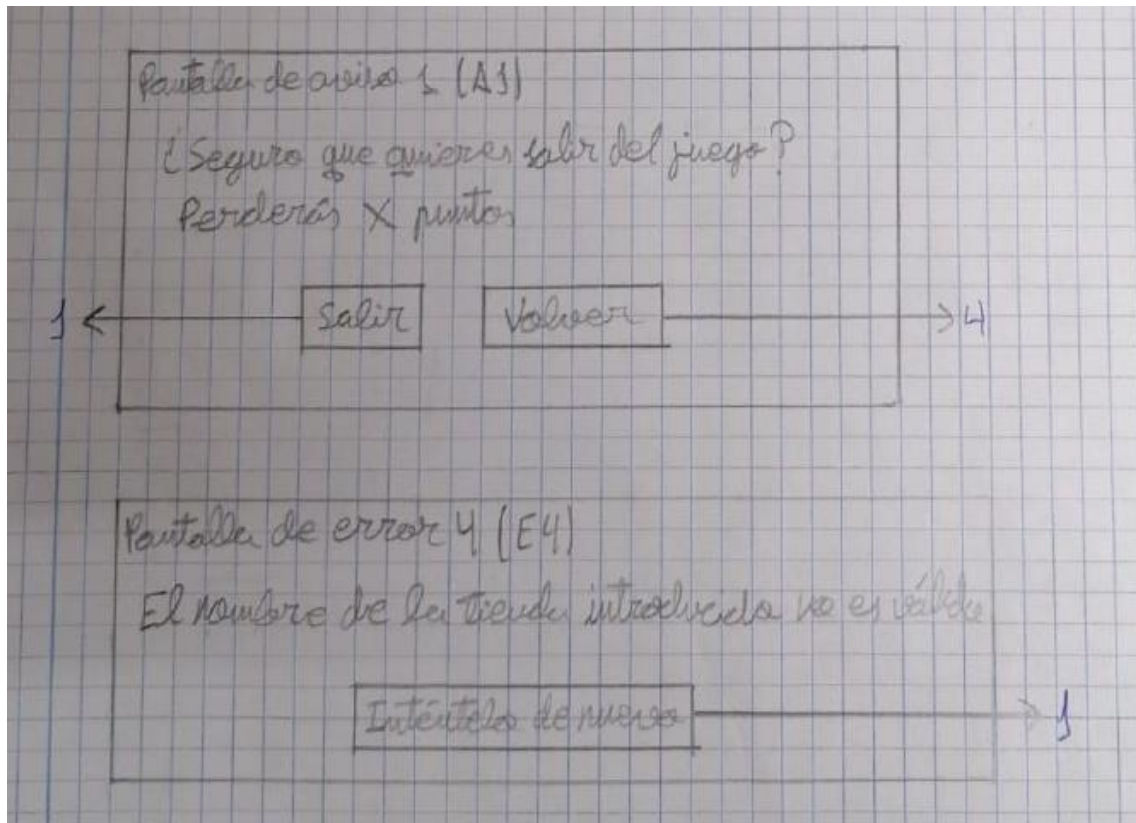


Esas dos pantallas los únicos cambios que han sufrido han sido un cambio de texto en el botón que pone “Menú principal”. También el texto “FIN DEL JUEGO” de la pantalla 5 ha sido sustituido por diferentes textos en función de los diferentes finales de las partidas.



En la pantalla 6 lo único que ha cambiado ha sido la organización de los elementos que conforman los premios (la foto, nombre, botón añadir, etc) y se ha añadido en cada premio el botón eliminar premio.





De las pantallas mostradas, los únicos cambios que han sido implementados son un cambio en el texto de los botones que las componen junto con añadir una ventana de error más para la eliminación de un premio que no se encuentra en el pedido del usuario.

2.2 Fase de implementación

2.2.1 Lógica

Clase Main:

Es la clase encargada de ejecutar la aplicación por completo, cuenta con una constante “DEBUG” que, en función de los valores que guarde, cambiará la forma de funcionar el juego en las partidas únicamente.

Clase Juego:

Es la clase destinada a manejar toda la lógica referente al juego, los métodos que más destacan de esta clase son los siguientes:

- `controladorEliminacionDeEnemigos()`
Este método es el encargado de ir eliminando los invasores que hayan sido colocados en un turno y de ir actualizando los puntos del usuario en función de los tamaños de las colonias que se vayan eliminando.
- `getTextoDeLasReglasDelJuego()`
Este método es destinado a mostrar en la interfaz de la aplicación las reglas del juego de forma dinámica.
- `getTextoDeLasReglasDelJuegoResumidas()`
Este método es destinado a mostrar en la interfaz de la aplicación las reglas del juego resumidas de forma dinámica.
- `getTextoDeLaLeyenda()`
Este método es destinado a mostrar en la interfaz de la aplicación el texto de la leyenda de las iniciales de los invasores del tablero de forma dinámica.
- `generacionInvasoresEnTablero()`
Este método es el encargado de generar los 5 invasores de cada turno y colocarlos en el tablero de forma aleatoria.

- `generacionDelInvasorEnBaseAlDebugMode()`
Este método es usado en el anterior mencionado para retornar invasores que serán colocados en el tablero en posiciones aleatorias, con un tipo de invasor que será determinado en base al valor guardado en la constante `DEBUG` de la clase `Main`. Si el valor es 0, los invasores que se generan son de tipos aleatorios; si el valor es 1, los invasores que se generan son sólo de tipo cabecilla (Capitán General); y si el valor es 2, los invasores que se generan son de cualquier tipo MENOS cabecilla (Capitán General).

Clase Block:

Es una clase que extiende a la clase `Casilla` y que sus objetos representan a aquellas casillas del tablero de juego que son inaccesibles para el usuario.

No destaca ningún método en esta clase.

Clase Casilla:

Es la superclase de todas las clases destinadas a representar un tipo de casilla en el tablero de juego.

No destaca ningún método de esta clase.

Clase Espacio:

Es una clase que extiende a la clase `Casilla` y que sus objetos representan a aquellas casillas del tablero de juego que no están ocupadas por un invasor.

No destaca ningún método de esta clase.

Clase Invasor:

Es una clase que extiende a la clase `Casilla` y que sus objetos representan a los invasores del juego.

De esta clase solo destaca el siguiente método:

- `getTipoInvasorAleatorio()`
Este método retorna un tipo de invasor aleatorio basándose en un valor tomado de la constante `TiposInvasor` de forma aleatoria.

Clase Tablero:

Es una clase destinada a representar el tablero de juego.

No destaca ningún método de esta clase.

Clase MyJButton:

Es una clase que extiende a la clase JButton y es creada con el fin de facilitar la implementación de algunas mecánicas del juego y de la funcionalidad de la ventana de canje de premios, ya que se le añaden tres atributos de los cuales 2 guardan las coordenadas de un botón del tablero de juego para poder pasarle a la capa de la lógica una posición del tablero con la que se haya interactuado de forma exacta.

El otro atributo añadido guarda un objeto de la clase MyJSpinner con el fin de “asociar” un objeto MyJButton con un MyJSpinner, esto es usado en la pantalla de canje de premios para saber cuántas unidades de x premio hay que añadir o eliminar de un pedido.

No destaca ningún método de esta clase.

Clase MyJSpinner:

Es una clase que extiende a la clase JSpinner y es creada con el fin de añadir a los objetos de esta clase el atributo actionPerformed como si se tratase de un JButton con el fin de asignarle un identificador en la interfaz de usuario de canje de premios para que, a la hora de añadir o eliminar un premio del pedido, la aplicación sepa de qué MyJSpinner sacar las unidades de las que se deban añadir o eliminar un premio en el pedido.

No destaca ningún método de esta clase.

Clase CartaPremios:

Es una clase destinada a cargar la lista de premios de la aplicación que el usuario pueda canjear. Dichos premios son cargados del fichero “premios.dat”.

De esta clase solo destaca el siguiente método:

- `getArrayPremios()`
Es un método que, a partir de la lista de premios que se haya cargado del fichero antes mencionado, retorna un Array con la misma información que contenía la lista de premios. Esto de cara a la interfaz resulta de gran utilidad porque tenemos toda la información de cada premio cargado del fichero mencionado anteriormente convertida en objetos Premio que, a su vez, son guardados en un Array.

Clase PedidoPremios:

Es una clase destinada a manejar toda la lógica de la pantalla de canjeo de premios, ya que cuenta con un listado de todos los premios que se añaden o eliminan del pedido actualizado con cada interacción del usuario en la interfaz de canjeo de premios (Es decir, se actualiza la lista cada vez que se añade o elimina un premio del pedido).

Destacan los siguientes métodos:

- `add()`
Encargado de añadir a la lista del pedido un premio que no haya sido añadido al pedido anteriormente. En caso de haber sido añadido anteriormente, se modifica el premio del pedido de tal forma que aumenta el número de unidades de este.
- `remove()`
Encargado de eliminar de la lista del pedido un premio existente en dicho pedido. Si las unidades del premio a eliminar superan a las unidades existentes de dicho premio en el pedido, se elimina directamente el premio del pedido; en caso contrario, se disminuyen las unidades del premio del pedido tantas veces como unidades se hayan indicado por parte del usuario en la interfaz.

Clase Premio:

Es una clase cuyos objetos representan los premios de la aplicación.

No destaca ningún método de esta clase.

Clase Ticket:

Es una clase cuyos objetos representan los tickets que sean, generalmente, de la tienda de donde se instalará la aplicación.

No destaca ningún método de esta clase.

Clase TicketsLoaded:

Es una clase destinada a cargar la lista de tickets de compra de la tienda de un fichero que se va actualizando con cada compra realizada en la tienda. El fichero del que se cargan los tickets es el llamado “tickets.dat”.

Destacan los siguientes métodos en la clase:

- **removeTicket()**
Es un método encargado de eliminar de la lista de tickets cargados, un ticket con el que se haya accedido al juego. Tras eliminar el ticket de la lista, el fichero de tickets de la tienda (el “tickets.dat”) es actualizado con el cambio de no presentar el ticket usado en la máquina para una partida.
- **actualizarFicheroTickets()**
Es el método encargado de actualizar, llamando al método `saveToUpdateFile()` de la clase `FileUtil`, el fichero de tickets (el “tickets.dat”) tras haber eliminado un ticket que haya sido usado para acceder al juego de la aplicación.

Clase ValidTickets:

Es una clase destinada a cargar la información descrita en el fichero “config.dat” con el fin de obtener el código de la tienda de donde se instalará la aplicación y así poder verificar si los códigos de los tickets, que son pasados por la máquina al tratar de acceder al juego por parte de los usuarios, son de la tienda de la aplicación a instalar.

De esta clase no destaca ningún método.

Clase Service:

Es la clase encargada de contener toda la lógica de la aplicación.

De esta clase destacan los siguientes métodos:

- **validarCodigoTienda()**
Permite validar o no el código de la tienda de un ticket que haya sido introducido en la interfaz de la aplicación. Si el método retorna true, significa que el código de la tienda del ticket introducido es válido; retorna false en caso contrario.
- **validarCodigoTicket()**
Permite validar o no el código de un ticket que haya sido introducido en la interfaz de la aplicación. Si el método retorna true, significa que el código del ticket introducido es válido; retorna false en caso contrario.
- **validarImporteTicket()**
Permite verificar si el importe de un ticket que haya sido introducido en la interfaz de la aplicación es de al menos 20€. Si el método retorna true, significa que el importe del ticket introducido es de al menos 20€; retorna false en caso contrario.
- **getArrayPremiosPorCategoria()**
Permite filtrar la lista de premios a poder canjear en la aplicación en función de la categoría de premio que es pasada como parámetro. También permite filtrar los premios en función de los puntos que posea el usuario en la pantalla de canje si la opción de activar el filtro está activada (dicha opción se activa en la interfaz).

Clase Fileutil:

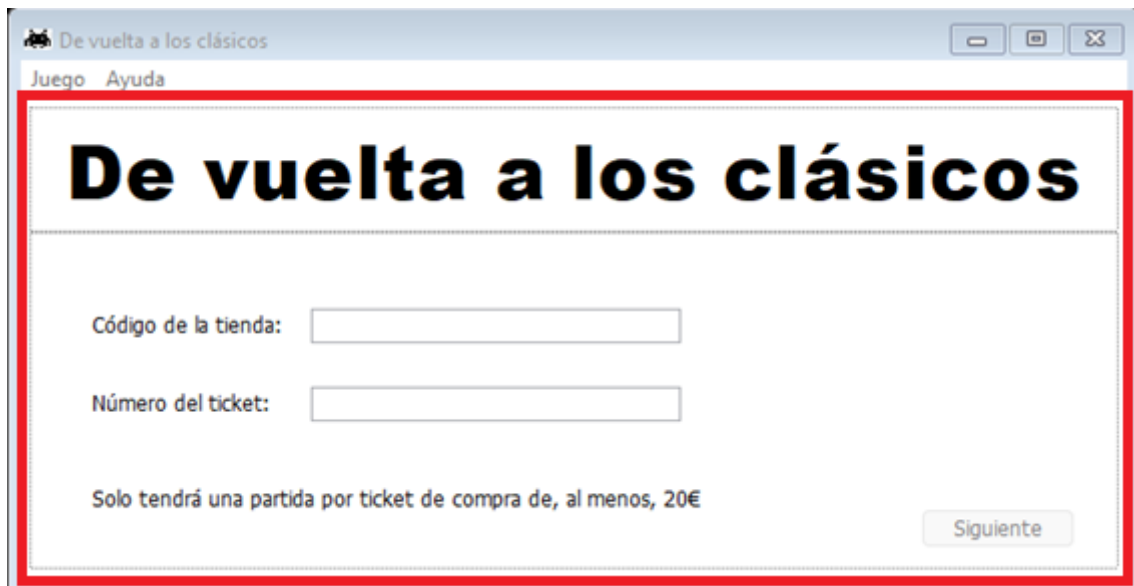
Es la clase dedicada a aportar la funcionalidad de lectura y escritura de ficheros de la aplicación

Destacan los siguientes métodos en esta clase:

- **loadValidTicketsFile()**
Es un método destinado a la carga de información de un fichero para la clase ValidTickets.

- `loadTicketsFile()`
Es un método destinado a la carga de información de un fichero para la clase `TicketsLoaded`.
- `loadPremiosFile()`
Es un método destinado a la carga de información de un fichero para la clase `CartaPremios`.
- `saveToFile()`
Es un método destinado a la descarga de información a un fichero. Dicha información es dada por la clase `PedidoPremios` para guardar la información acerca del pedido de un usuario en un fichero denominado "entregas.dat".
- `saveToUpdateFile()`
Es un método destinado a actualizar el fichero de tickets tras haber sido eliminado un ticket de dicho fichero porque ha sido usado para acceder a la aplicación.

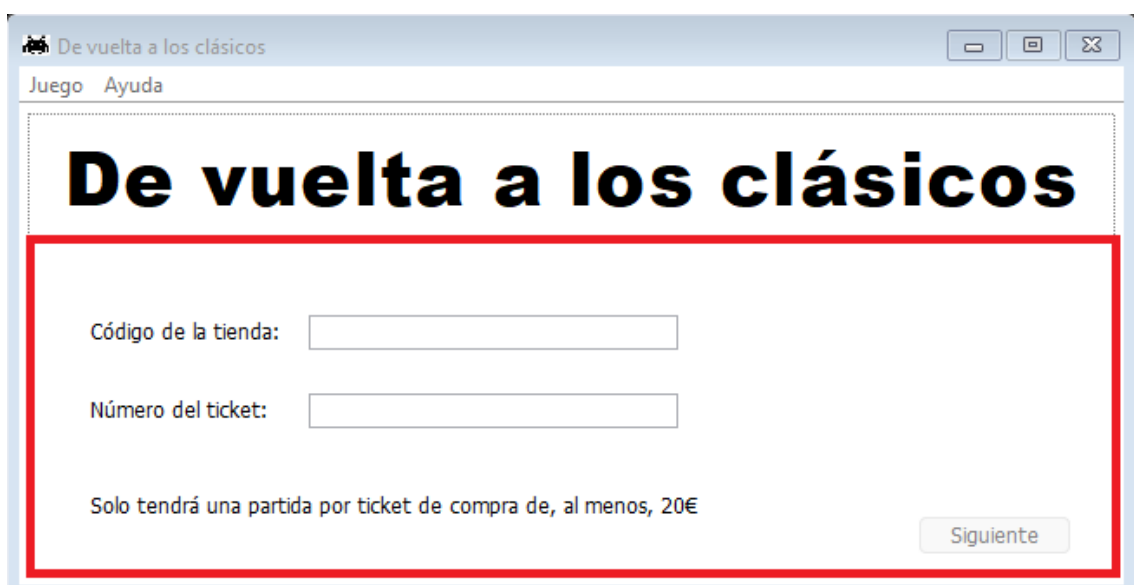
2.2.2 Interfaz



De la primera captura, el primer componente más relevante es el uso de un panel principal que dispone de un BorderLayout que contiene:

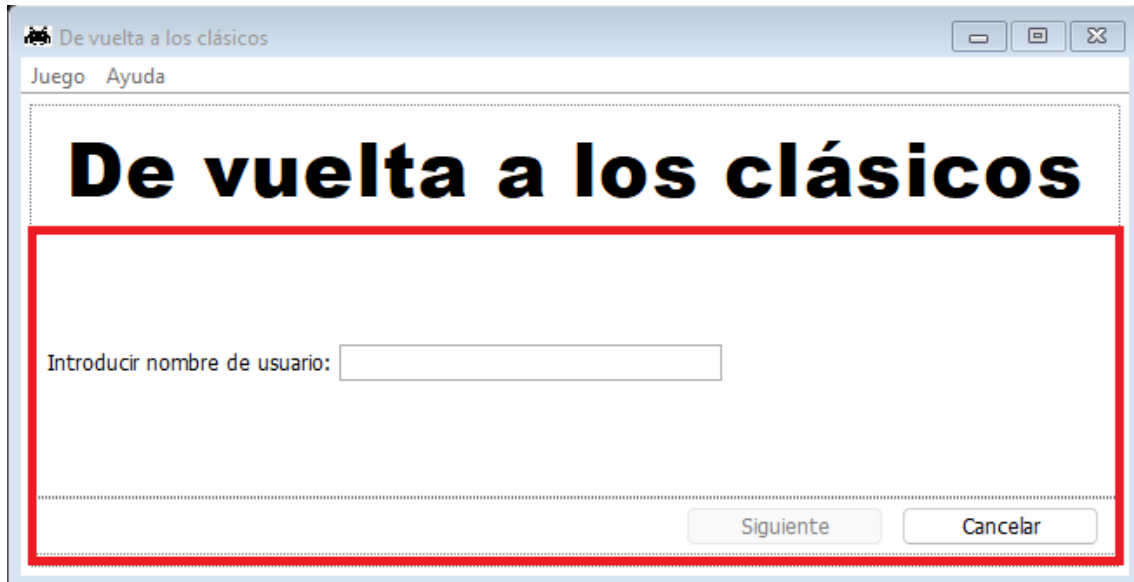
- En el norte, un panel con un FlowLayout conteniendo una label con el título del juego.
- En el centro, un panel con un CardLayout conteniendo unos tres “subpaneles” con los que, mediante ciertas acciones con ciertos componentes en cada subpanel, podremos pasar de un “subpanel” a otro sin tener que destinar una clase de interfaz de usuario por cada ventana que tenga la aplicación.

El primer “subpanel” de la primera ventana es el que se muestra a continuación:



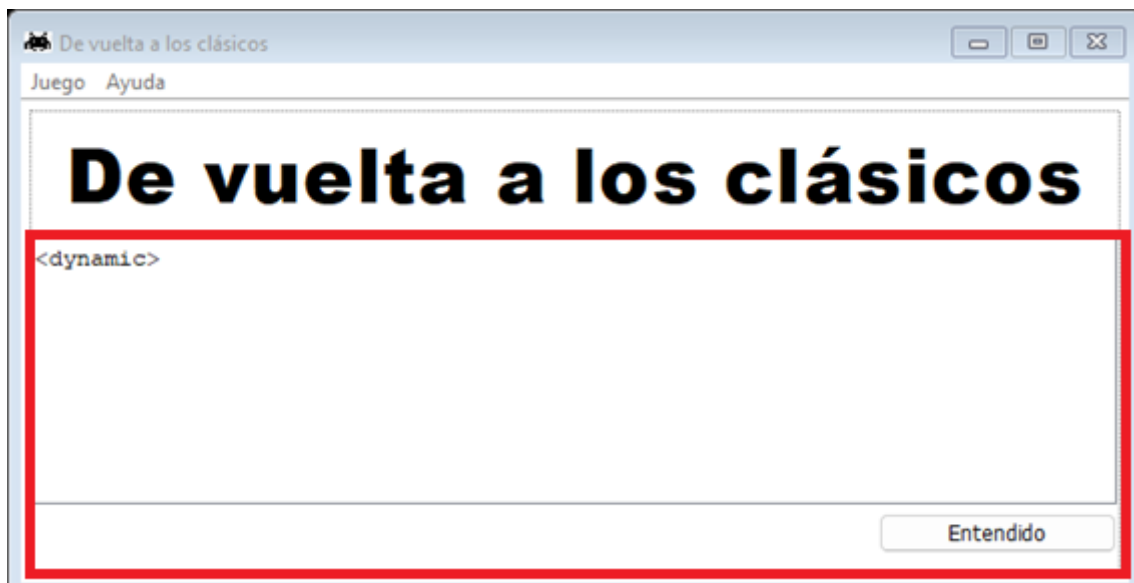
Del primer “subpanel” destacan dos campos de texto en los que se introducen el código de la tienda y el número del ticket. Estos dos componentes activarán el botón “Siguiente” una vez dejen de ser campos vacíos en tiempo de ejecución.

El botón “Siguiente” permitirá al usuario acceder al siguiente “subpanel” que se muestra a continuación:



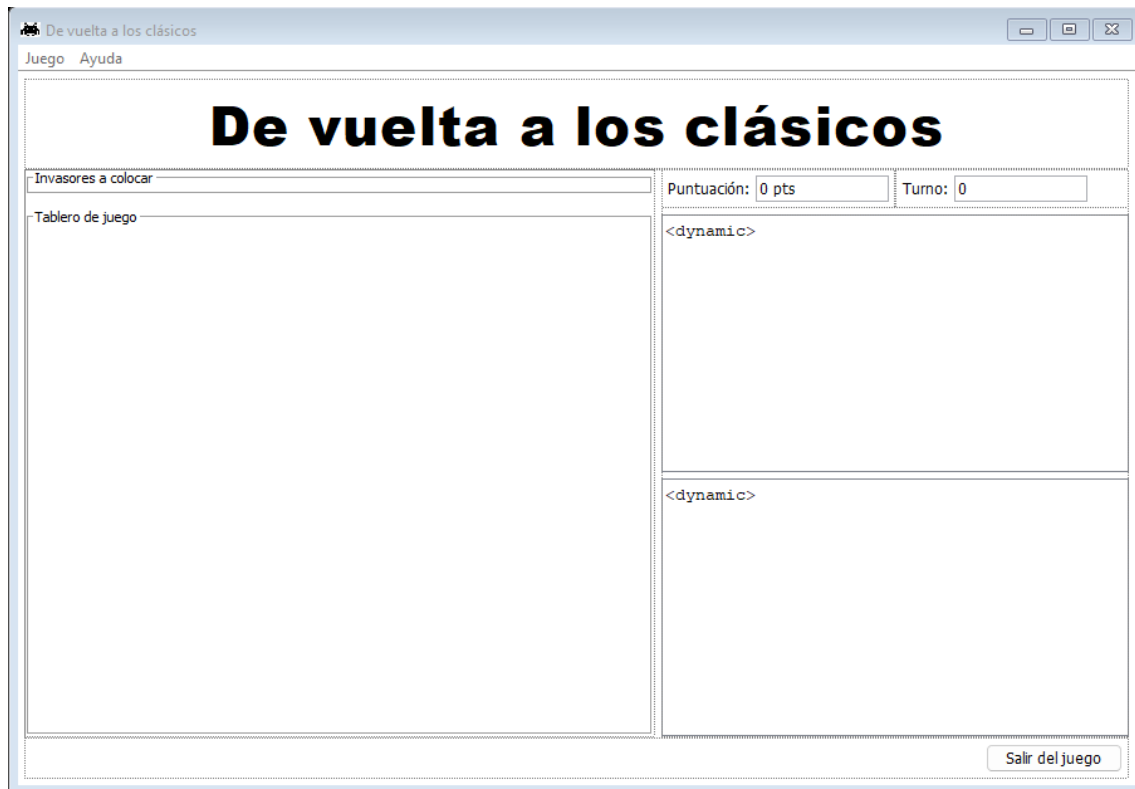
Del segundo “subpanel” destaca un campo de texto en el que el usuario introducirá su nombre como jugador. Una vez que este campo no esté vacío, se activará el botón “Siguiente” que permitirá al usuario pasar al tercer y último “subpanel” del CardLayout.

Dicho tercer “subpanel” es el que se muestra a continuación:



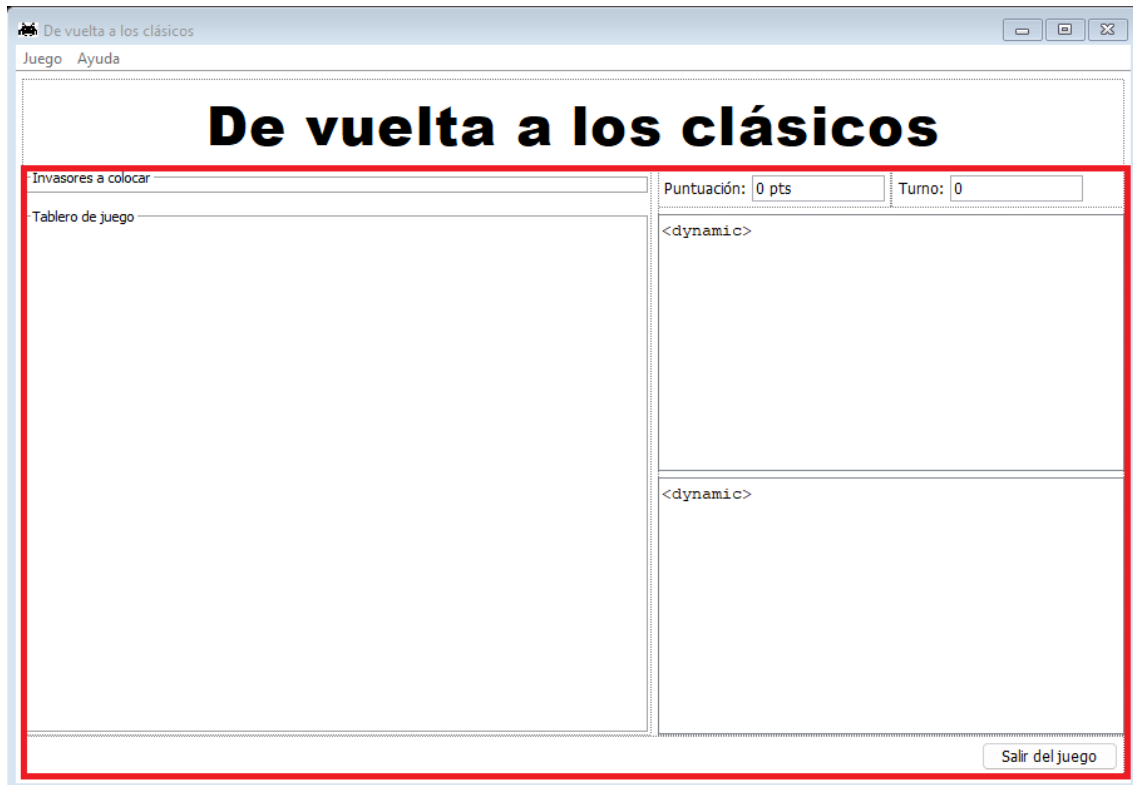
Del tercer “subpanel” destaca un área de texto en el que se mostrará en tiempo de ejecución, un texto con las reglas del juego al completo. El botón “Entendido” permitirá al usuario acceder a la primera ventana del juego de la aplicación.

Dicha primera ventana es la que se muestra a continuación:



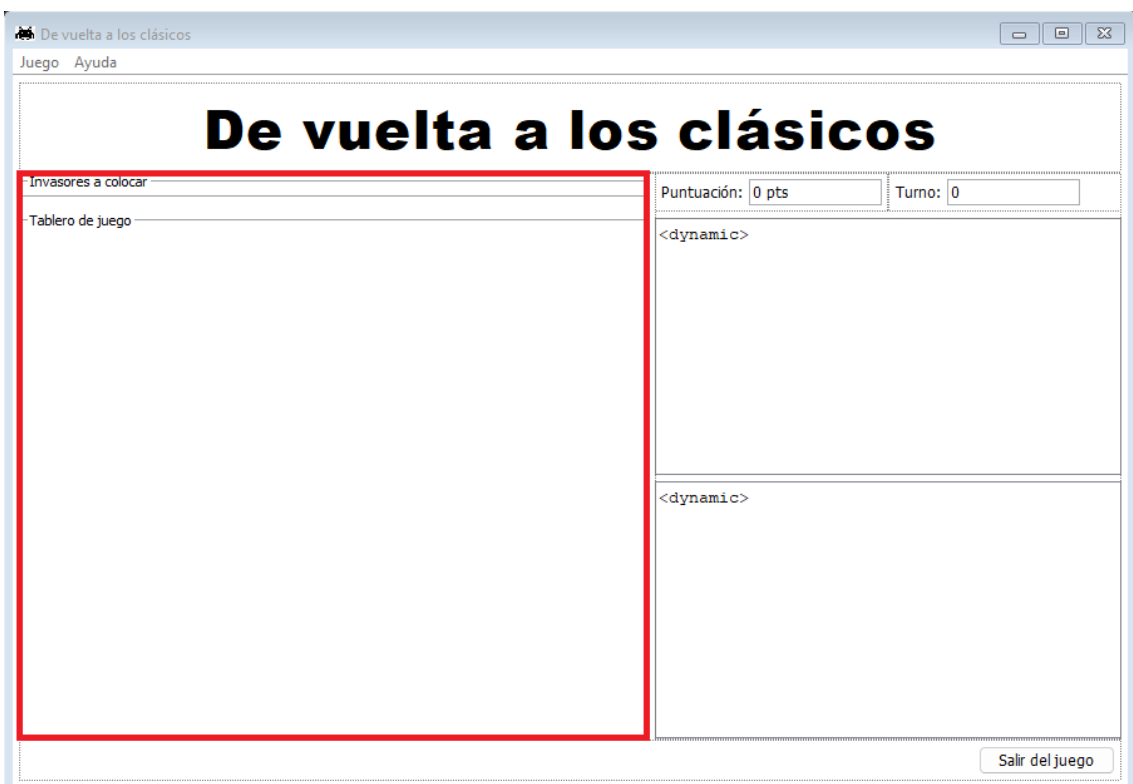
De esta ventana de juego destaca, principalmente, un panel que con un BorderLayout que contiene:

- En el norte, un panel con un FlowLayout que contiene una etiqueta con el título del juego.
- En el centro, un botón “Salir del juego” que permite al usuario abandonar la partida en caso de desearlo, y un panel secundario con un CardLayout que contiene dos “subpaneles”, el primer “subpanel” es el del juego de la aplicación, siendo éste el que se muestra a continuación:

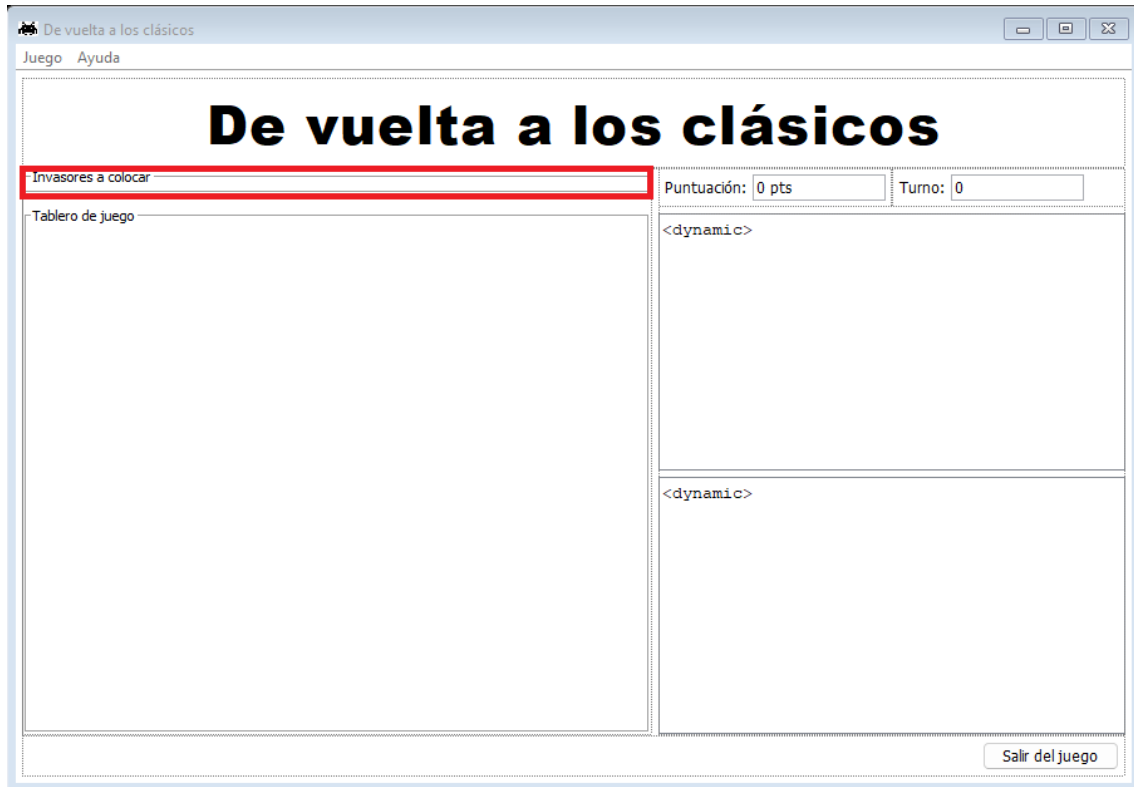


Dentro de éste primer “subpanel” destaca un panel que contiene una serie de componentes ordenados jerárquicamente en función de lo que contienen cada uno:

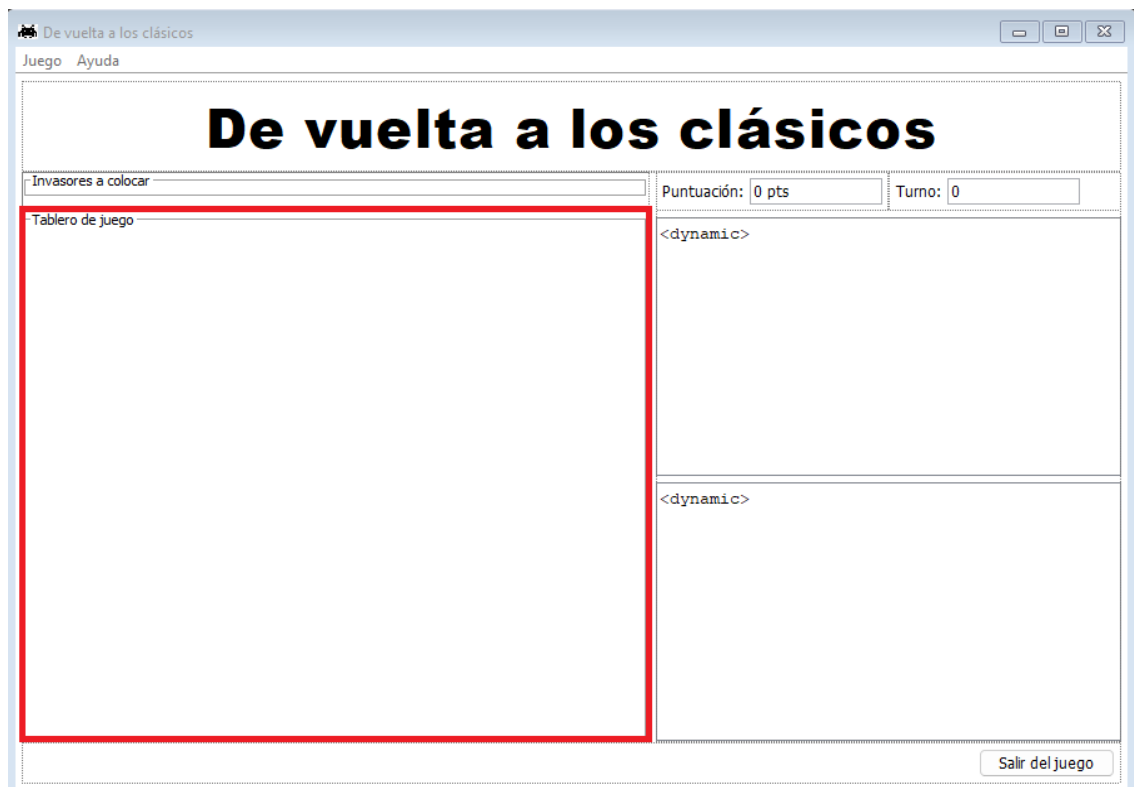
- Un panel con un BorderLayout que contiene los elementos relacionados con el tablero de juego.



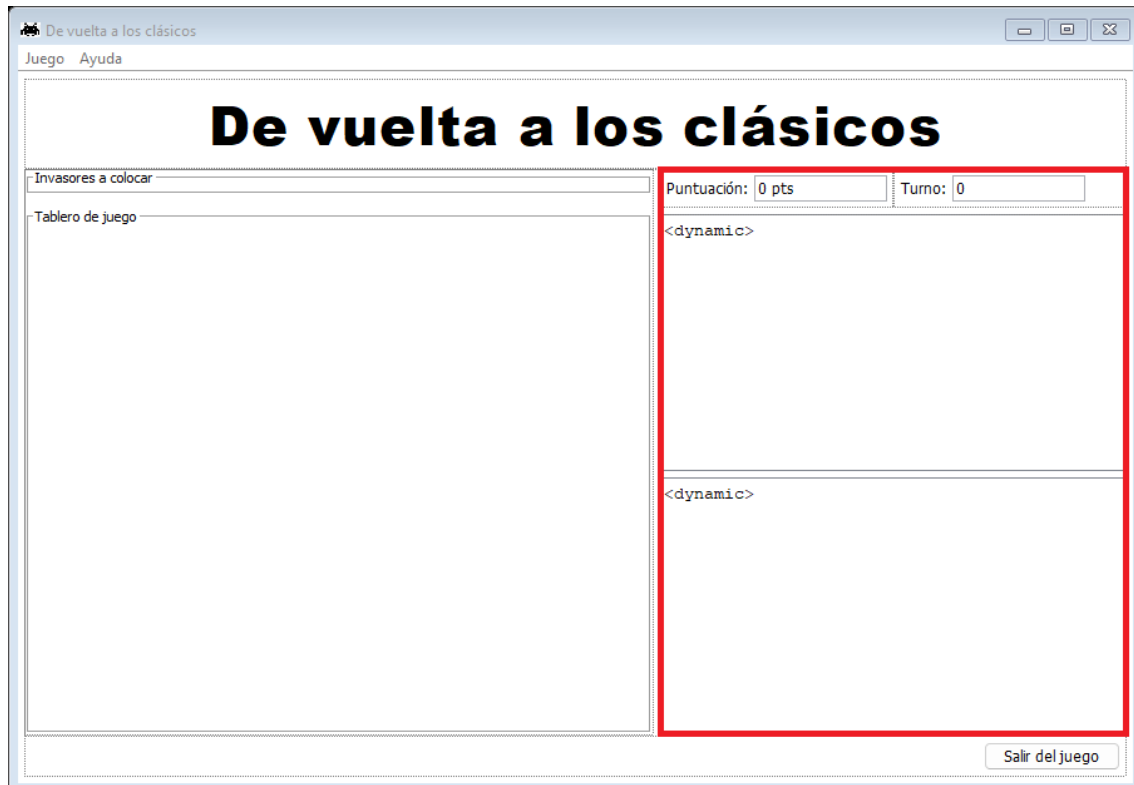
Dichos elementos son dos paneles, uno que contendrá a los invasores a colocar por parte del usuario en el tablero durante un turno.



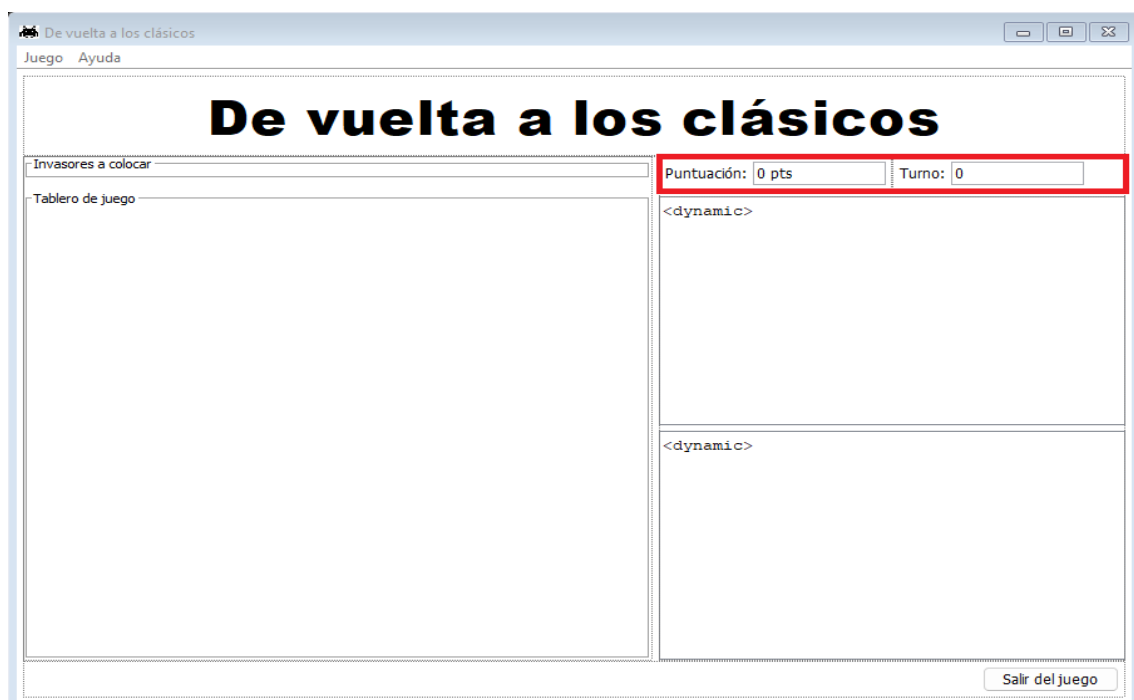
El segundo panel es el que muestra el tablero de juego que se va actualizando conforme el usuario coloque los invasores y se pase de turno.



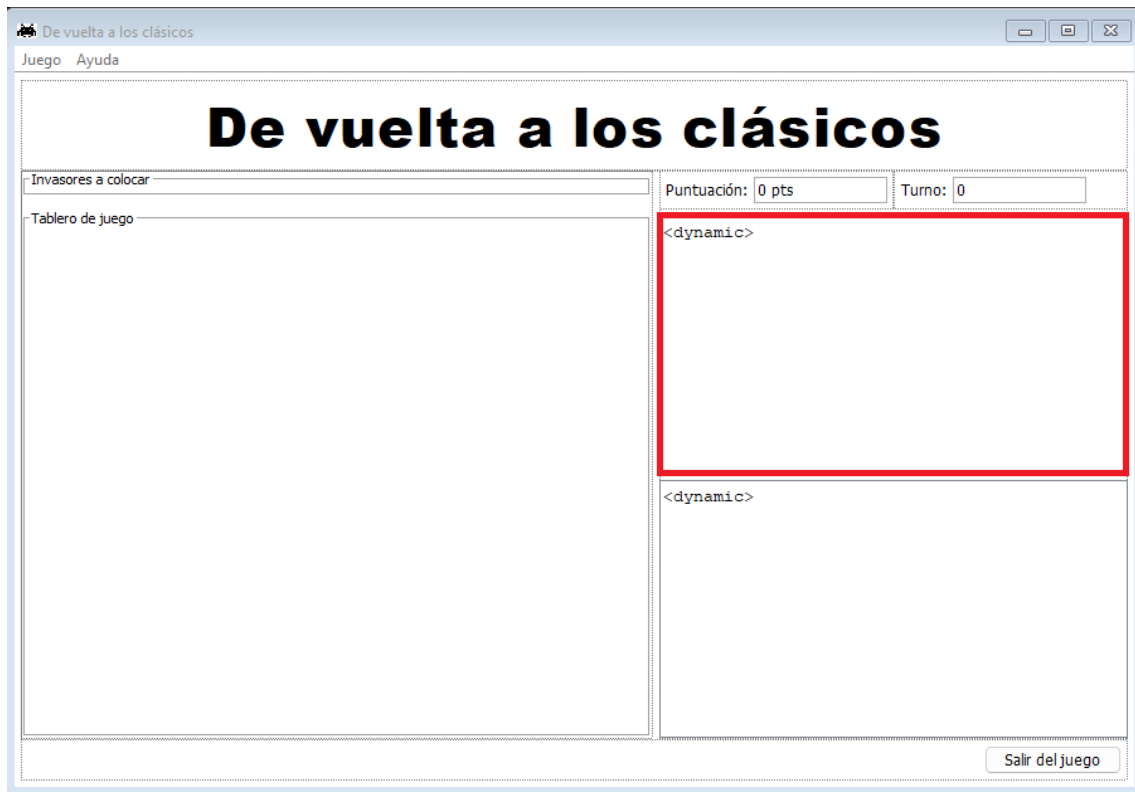
- Otro panel con otro BorderLayout que contiene los elementos relacionados con la información acerca del juego tales como un resumen de las reglas, una leyenda con las iniciales de cada tipo de invasor en el tablero de juego, el turno en el que se desarrolla cada momento de la partida, y los puntos que va consiguiendo el jugador a medida que transcurre la partida.



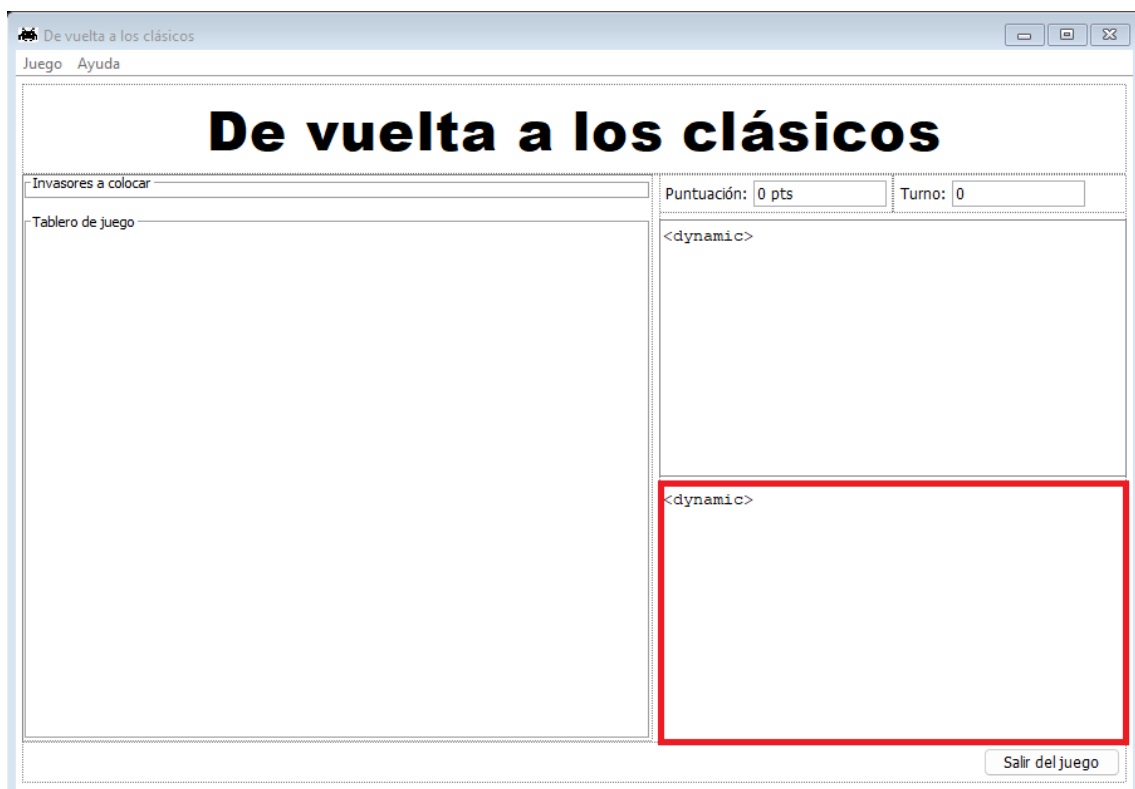
Los campos que se muestran a continuación son los que contienen el número de puntos del jugador en la partida y el turno en el que se desarrolla ésta.



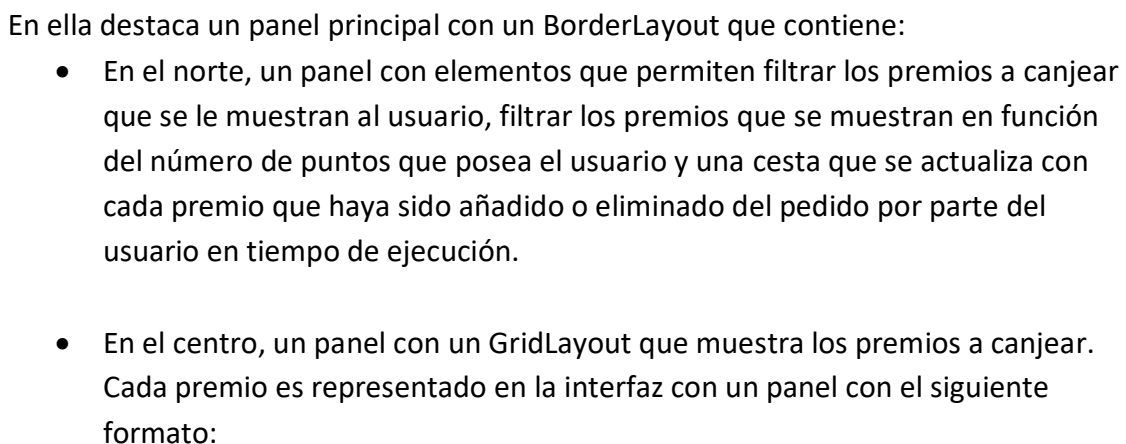
El siguiente campo es un área de texto que muestra la leyenda de las iniciales de cada tipo de invasor en el tablero de juego:



El último campo que se muestra es otro área de texto con un resumen de las reglas del juego:



Esta segunda “subpantalla” es la que permite al usuario canjear una serie de premios en función del número de puntos que haya conseguido en la partida. Dicha “subpantalla” es la que se muestra a continuación:



Nombre: Auriculares Turtle Beach

Precio: 150 pts



Sumergete en tus juegos con el potente sonido amplificado y el potenciador de graves de los Recon 200

Unidades:

- En el sur, un panel con un GridLayout que contiene dos “subpaneles”. El primer “subpanel” es que el contiene un campo de texto en el que se introduce el DNI del usuario. Una vez que este campo deja de estar vacío, el botón “Canjear” del segundo “subpanel” se activa.
El segundo “subpanel” contiene una serie de botones con los que el usuario puede finalizar el canjeo de premios, salir del juego o acceder a la ayuda de la aplicación.

2.2.3 Pruebas

- Escenario 1:

Nombre: Ana

Edad: 42

Ocupación: Asesora contable

Escenario: Ana ya ha jugado en otra ocasión, así que no tiene dificultades para comenzar la partida. Introduce los datos del ticket y tienda que le pide el terminal y se validan sin problema. Juega y obtiene 5500 puntos. Va directamente a la sección de Consolas, que identifica claramente, para ver qué puede elegir. Comienza a añadir artículos y cuando alcanza la suma de 5300 puntos en regalos intenta añadir uno de 300 de la sección de Videojuegos, pero no es posible. Intenta confirmar los regalos elegidos por valor de 5300 pero el sistema le avisa de que perdería 200 de los puntos obtenidos. Así que devuelve un regalo por valor de 100 puntos y coge el que quería de 300 de manera que ya no le sobra ningún punto. La aplicación le confirma que se han procesado la selección, así que se va muy contenta.



El resultado obtenido es el esperado, la aplicación informa sobre la cantidad de puntos sobrantes a la hora de canjear si estos existen y, a su vez, no permite añadir un premio cuyo valor sea superior a la cantidad de puntos disponibles para canjear que tenga el usuario.

- **Escenario 2:**



Nombre: Antonio

Edad: 83

Ocupación: Jubilado

Escenario: A Antonio le acaban de comentar al pagar su compra que tiene opción a jugar en el TPV. Muy contento y algo asustado pq nunca interactuó con algo similar, se acerca al terminal. Tiene que meter dos códigos: uno el del ticket y otro el de la tienda. Por las pistas visuales que le da la interfaz entiende perfectamente dónde localizarlos en su ticket y lo introduce sin problemas. Llega a la pantalla donde le aparece el tablero de juego y no sabe qué tiene que hacer ahora; agradecería una pequeña introducción al juego. Comienza a jugar y acumula un total de 1200 puntos. Tiene 3 nietos así que le gustaría elegir 3 premios iguales. Ve claramente las diferentes categorías de regalos pero cree que le va a costar mucho trabajo navegar por todas para localizar artículos de un máximo de 400 puntos. Así que le gustaría que hubiese una opción para que, en base a su puntuación, viese todos los regalos a los que podría optar, esto es, los que valiesen igual o menos que los puntos acumulados y así poder hacer la selección de una manera más sencilla. Así elegiría 3 unidades de alguno de los premios de 400 puntos o menos.

El resultado obtenido es el esperado, la aplicación proporciona pistas visuales de la información del ticket a introducir en la máquina (en la ayuda), una guía con los controles (en la ayuda) y reglas del juego, y la opción de filtrar los premios a canjear en base a los puntos acumulados que tenga en usuario.

- **Escenario 3:**

Nombre: George

Edad: 27

Ocupación: Profesor de inglés

Escenario: George acaba de hacer una compra en la tienda y se acerca al terminal del juego a probar suerte. Recuerda que tiene otro ticket de una tienda de Sevilla, así que decide probar suerte con éste también. Como está de viaje por España y todavía no domina el idioma, espera que, o bien la interfaz sea lo suficientemente intuitiva por los controles e imágenes utilizados o bien que sea posible cambiar al idioma inglés. Se acerca a la máquina e introduce los datos que le piden del ticket de Sevilla. Mediante la información que le proporciona la aplicación entiende que no puede jugar con ese ticket ya que el código de la tienda que ha proporcionado no coincide con el de la tienda en la que está intentando jugar. Prueba suerte de nuevo con el ticket de la tienda. Tampoco consigue jugar. Esta vez también le queda claro que es debido a que no ha alcanzado el importe mínimo de 20 euros, así que se va frustrado pero teniendo claro cuáles son los requisitos para poder jugar en otra ocasión.



El resultado obtenido es el esperado, la aplicación no permite jugar al usuario si el ticket introducido no es de la tienda de la máquina o, en el caso de que el ticket sea de la tienda, que la máquina le avise al usuario que no puede jugar por tener un ticket cuyo importe es inferior a 20€ y que no le deje; también le avisa al usuario acerca de los dos problemas mencionados antes en caso de que ocurran. También se muestra mediante imágenes cómo son los controles del juego.

- Escenario 4:



Nombre: Sara

Edad: 30

Ocupación: Empresaria

Escenario: Sara viene a España desde Estados Unidos a cerrar un negocio empresarial y, tras cerrar el negocio, se da una vuelta por la ciudad hasta encontrar la tienda y decide comprar una figura de un videojuego como recuerdo por 35€. Al ver que se le ofrece jugar a la máquina por tener un ticket de compra superior a 20€, decide jugar una partida.

Al ver la interfaz de la máquina se da cuenta de que tiene la opción de cambiar el idioma del juego a inglés para que no lo tenga en español, ya que no sabe nada de español.

Gana la partida con una puntuación final de 4 puntos pero al llegar a la pantalla de recompensas se da cuenta de que no le alcanza la puntuación para reclamar cualquiera de los premios disponibles. Ante esta situación Sara intenta reclamar cualquiera de los premios pese a no tener la puntuación suficiente pero la máquina le muestra un mensaje diciendo que no tiene la puntuación suficiente, así que se va de la tienda y vuelve a Estados Unidos.

El resultado obtenido es más o menos el esperado.

Pese a no contar con el posible cambio de idioma de la aplicación, ésta sí que cuenta con la funcionalidad para impedir que se canjeen premios cuyos precios sean superiores a la puntuación obtenida por el usuario.

- Escenario 5:



Nombre: Paco

Edad: 79

Ocupación: Jubilado

Escenario: Paco esta dando una vuelta con sus amigos y entran en la tienda para verla y se encuentran con la máquina. Los amigos de Paco le animan a que eche una partida para divertirse un poco aunque Paco temía de que las reglas de juego fueran demasiado difíciles.

Al intentar empezar la partida, la máquina le salta un mensaje diciendo que necesita un ticket de compra de mínimo 20€ para poder jugar. Entre todos intentaron comprar algo de la tienda para tener un ticket de compra superior a 20€ pero no les llegaba el dinero, así que se van de la tienda sin poder echar una partida.

El resultado obtenido es el esperado, la aplicación cuenta con la funcionalidad para impedir que se juegue una partida con un ticket cuyo importe sea inferior a 20€, y también cuenta con unas reglas resumidas y simplificadas para explicar al jugador. Dichas reglas son visibles antes de empezar la partida y durante esta.

- Escenario 6:



Nombre: Juan

Edad: 48

Ocupación: Abogado

Escenario: Juan llega a la tienda y le compra un juego a su hijo para su cumpleaños. Se gasta 15€ en el juego e intenta echar una partida en la máquina pero al ser un ticket de compra de inferior a 20€ no le deja echar la partida. Juan compra otro juego por unos 5€ para tener entre los dos tickets una compra en total de 20€. Vuelve a la máquina pero al escanear el segundo ticket le salta el mensaje de que el ticket de compra es inferior a 20€. Al ver que no puede acumular el dinero de dos compras, vuelve a casa con dos juegos como regalo de cumpleaños para el hijo

El resultado obtenido es el esperado, la aplicación permite jugar solo a aquellos usuarios que introduzcan unos datos de un ticket cuyo importe sea de, al menos, 20€. No permite la acumulación de varios importes de varios tickets hasta dar con un importe final de, al menos, 20€.

- Escenario 7:



Nombre: Natalia

Edad: 14

Ocupación: Estudiante

Escenario: Natalia llega a la tienda tras salir de clase y compra una consola por 45€ como recompensa de sacar notas muy altas en los exámenes. Natalia juega una partida en la máquina aprovechando que tiene un ticket de compra superior a 20€ y gana la partida con una puntuación final de 90 puntos.

Al llegar a la pantalla de recompensas elige un juego para su nueva consola por 25 puntos y un mando por 80 puntos pero al reclamar los dos productos, la máquina le dice que no tiene los puntos suficientes para comprar ambos productos, así que decide comprar el mando y una camiseta que le costó 10 puntos.

El resultado obtenido es el esperado aunque la aplicación no avisa al usuario de que no puede canjear una serie de premios o un premio en particular si la suma de los precios de estos supera a los puntos del usuario. En vez de ello, directamente avisa al usuario cuando trata de añadir una o varias unidades del premio (suponiendo que el usuario no dispone de los puntos necesarios para canjear el premio en concreto) de que no se puede añadir el premio seleccionado al pedido porque no tiene los puntos suficientes para añadirlo al pedido.