Perseverance

Solitario Apolo

Diseño del Sistema

Autores:

- ➢ Espósito, Javier
- ➢ Sartori, Gastón Emilio
- ➢ Tántera, Julián
- ➢ Zanotti, Emiliano

Fecha: 4 de Junio de 2021

Versión: 1.0.1

Historial de Cambios

Versión	Fecha	Cambios
1.0.0	4 / 6 / 2021	Documento inicial
1.0.1	13 / 6 / 2021	Cambios en sección 3
1.0.2	17 / 6 / 2021	Cambios en sección 2.2

1. Introducción

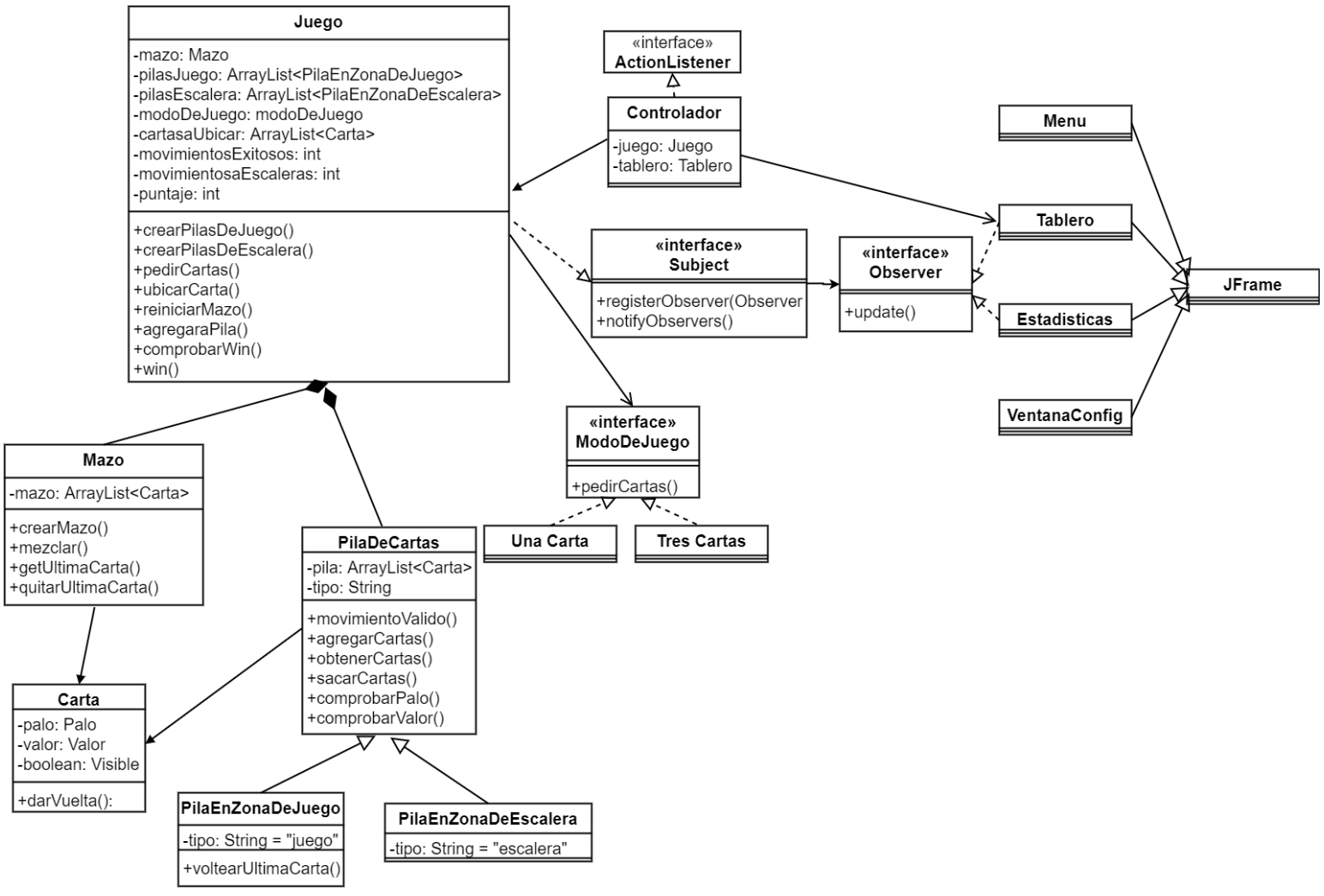
A lo largo de este documento, se mostrará en detalle el diseño del sistema, a través de diferentes tipos de diagramas UML. Además se especificarán qué patrones de diseño se utilizarán y las ventajas que traen los mismos.

2. Diseño del sistema

A continuación se mostraran los distintos diagramas y explicaciones que determinan el diseño del sistema.

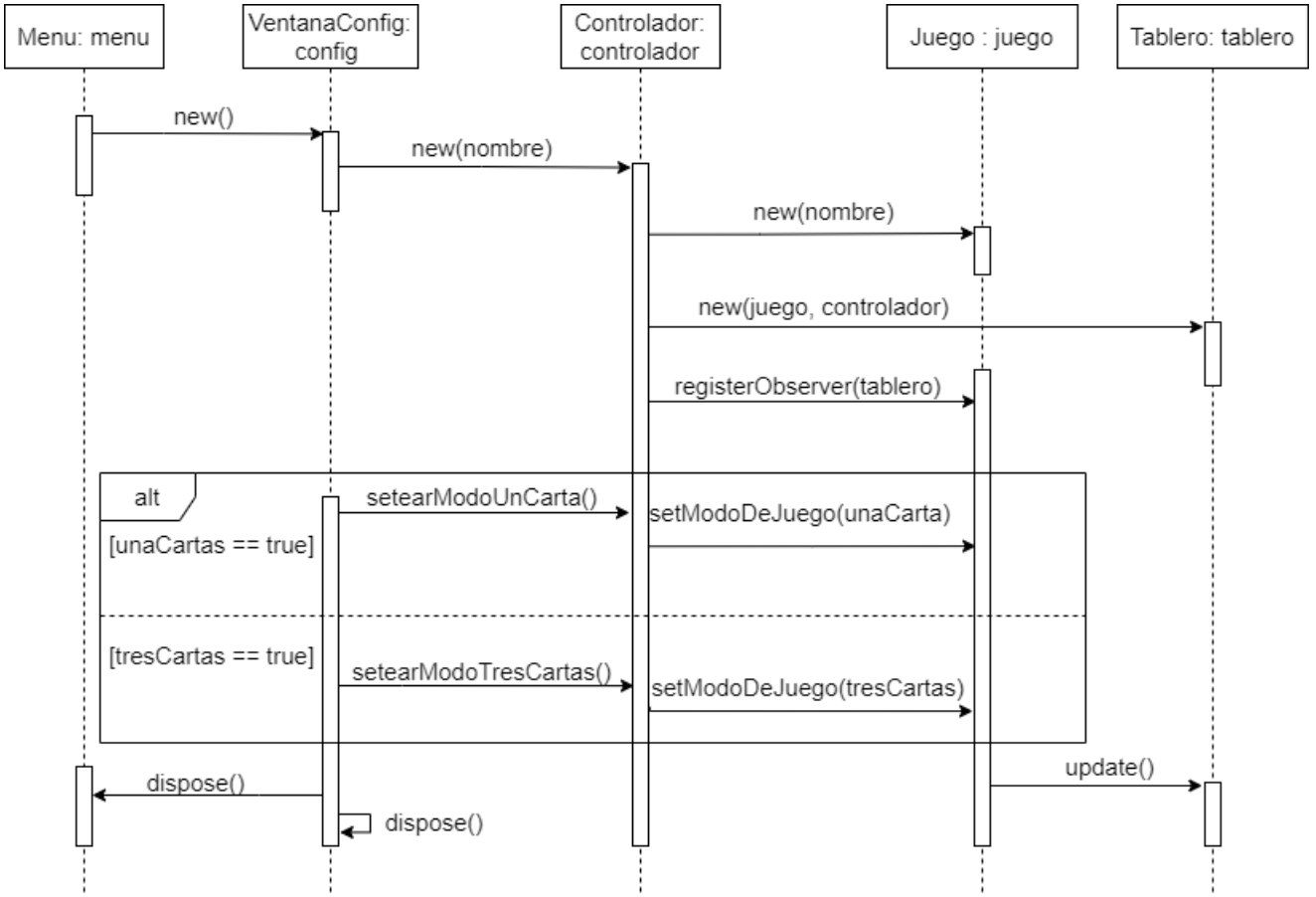
2.1. Diagrama de clases

A continuación se muestra el diagrama de clases del proyecto, mostrando los campos y métodos más importantes de las mismas. Los cuales pueden variar ligeramente durante el desarrollo del proyecto.



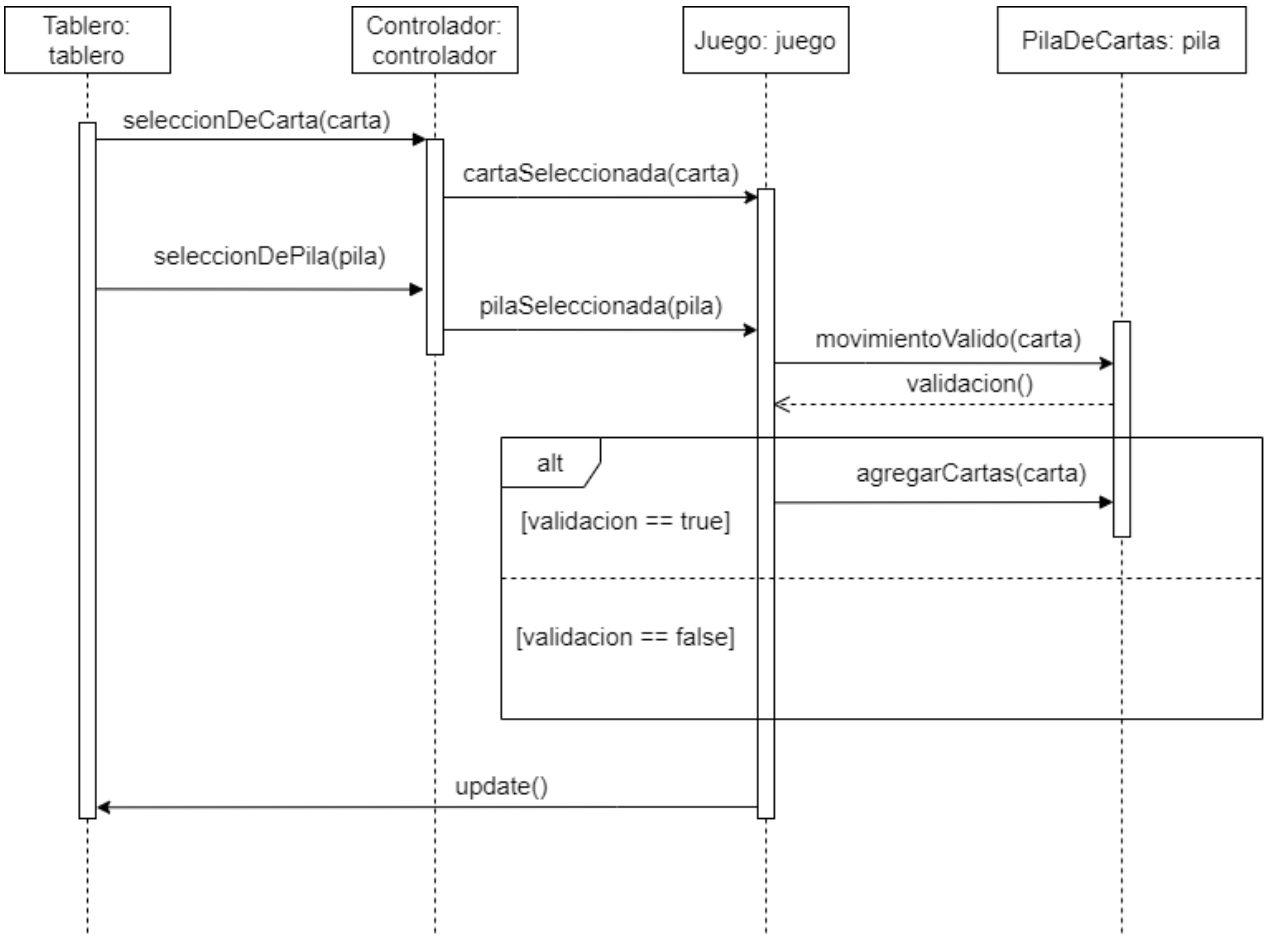
2.2. Diagramas de secuencia

En el primer diagrama, podemos observar las secuencia de creación de los principales módulos del programa, el Controlador (controller), el Juego (model) y el Tablero (view), como así también la presencia de los patrones de diseño Observer y Strategy.



La aplicación comienza con la creación de un Menú y luego de una VentanaConfig, el usuario interactúa con esta ventana, ingresando su nombre y estableciendo el modo de juego deseado. Definido esto, se crea un Controlador, el cual crea un nuevo Juego, determinando el nombre del jugador y un nuevo Tablero. Observamos que este tablero es registrado por parte del Juego para ser notificado al tener cambios en la información, propio del patrón de diseño Observer. Luego, a través del uso del patrón Strategy, se setea el modo de juego deseado. Finalmente el Tablero es notificado para actualizar su información de las cartas que debe mostrar al usuario.

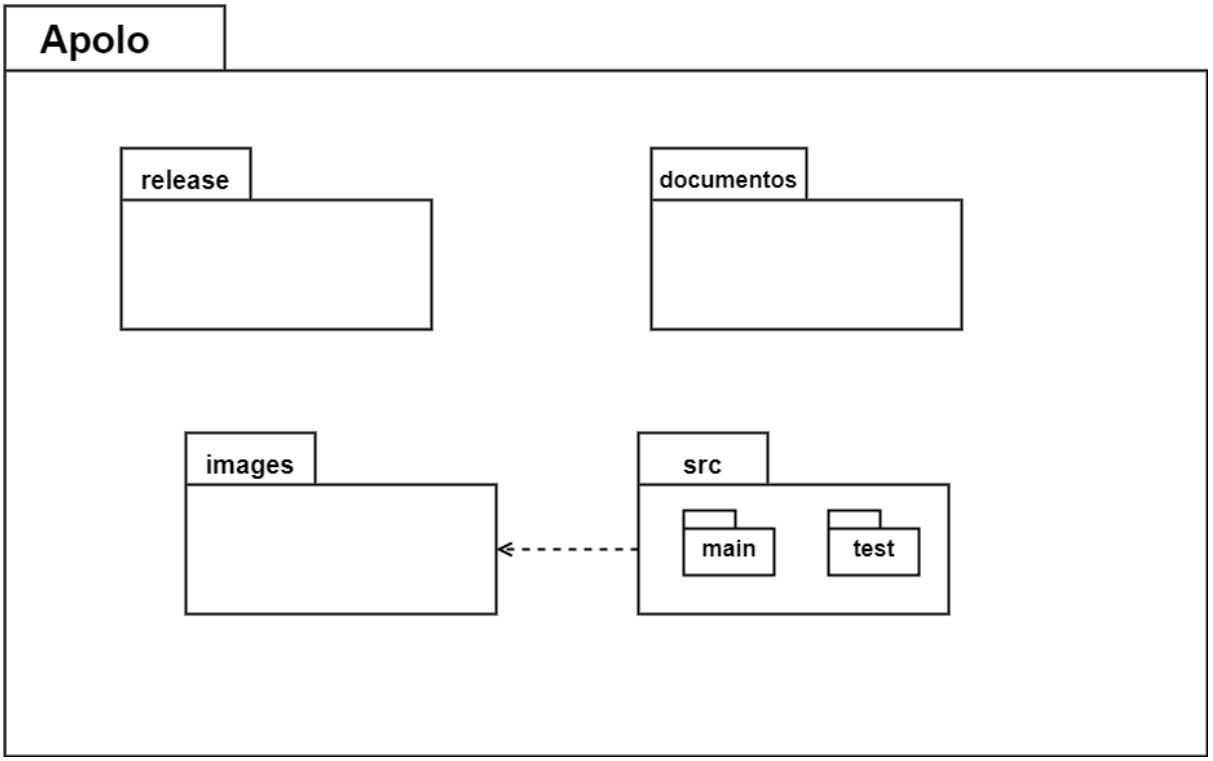
En el siguiente diagrama de secuencia, podemos observar a grandes rasgos el funcionamiento del programa en una de las interacciones más importantes del mismo, el realizado de movimientos y la determinación de si estos son válidos o no.



Luego de que el usuario realiza sobre la ventana de la interfaz gráfica Tablero, la acción de selección de la carta que desea mover y luego la pila a donde desea moverla. Estas acciones son recibidas por el Controlador y comunicadas al Juego, el cual pregunta a la pila seleccionada si el movimiento deseado es válido, de ser correcto, se agrega efectivamente la carta a la pila, si no lo es, no realiza el movimiento. Finalizado esto, notifica al tablero, haciendo uso del patrón de diseño Observer, que actualice su información a mostrar al usuario.

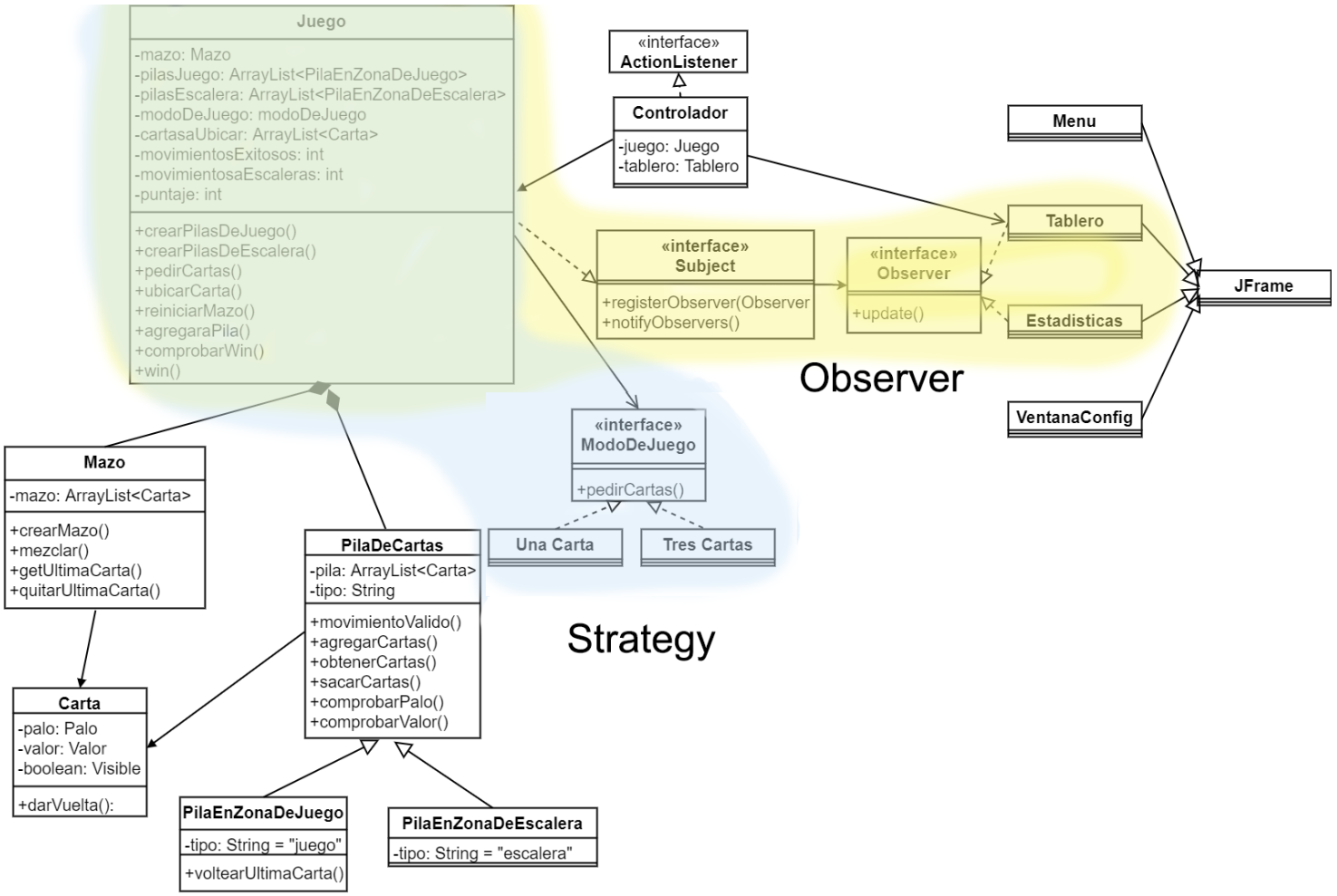
2.3. Diagrama de Paquetes

En el siguiente diagrama se pueden observar los diferentes paquetes que integran el proyecto, y las dependencias entre ellos:



2.4. Patrones de diseño

A continuación se especificará el uso de los patrones de diseño Observer y Strategy, nombrados en el documento de Arquitectura. Para esto se utilizara el diagrama de clases mostrado anteriormente y se resalta aquellas clases que hagan referencia al uso de cada patrón.



Patrón Strategy: dentro de la zona resaltada por el patrón Strategy, observamos la interfaz ModoDeJuego y las clases Juego, UnaCarta y TresCartas. Dentro de la clase Juego, uno de sus campos es un objeto de la clase ModoDeJuego, la cual se utilizará para determinar si al pedir cartas al mazo, nos entrega una carta o tres cartas. Para esto, existen las clases UnaCarta y TresCartas, quién son las que se encargaran de pedir una o tres según corresponda, ya que al momento de pedir una carta, la clase Juego ejecutará el método de su campo ModoDeJuego, el cual dependiendo de su clase específica, realizará el pedido correspondiente. Al esto estar definido por objetos que pueden crearse durante la ejecución, el modo de juego puede modificarse de forma dinámica durante la partida, a partir del seteo de este campo.

Patrón Observer: dentro de la zona resaltada por el patrón Observer, observamos las interfaces Subject, Observer y las clases Juego, Tablero y Estadísticas. Las interfaces son aquellas propias del patrón, y las cuales implementan las clases de nuestro proyecto. Juego implementa Subject, ya que es quien lleva la información de la partida y Tablero y Estadísticas implementan Observer, ya que son ventanas de la interfaz gráfica, quienes utilizan la información para mostrarla al usuario. Tablero utiliza la información de que cartas se encuentran en cada pila y el puntaje de la partida. Estadísticas utiliza la información de puntaje, cantidad de movimientos realizados exitosamente y cantidad de movimientos a zona de escaleras, para mostrar con mayor detalle cómo se realizó el cálculo del puntaje.

La aplicación de este patrón nos permite que este envío de información de parte de Juego a Tablero y Estadísticas, solo se haga cuando la información que estos necesitan tenga cambios. De esta forma, cuando Juego sufra cambios en su información, este “avisa” a las ventanas para que se actualicen.

3. Casos de pruebas unitarias

A continuación se listan los casos de pruebas unitarias generadas hasta el momento de desarrollo en que se generó este documento. El objetivo de estas es verificar el correcto funcionamiento de cada uno de los módulos que componen nuestro proyecto. Y que además, se puedan ejecutar automáticamente cada vez que se genera un nuevo commit en el repositorio, con el fin de controlar qué cambios en el código o correcciones de bugs, no corrompan otras funcionalidades. A medida que se avance con el desarrollo, se generarán más casos de pruebas para testear el nuevo código.

Test case ID: TU 001

Descripción: se probará la funcionalidad de crear un mazo.

Función a testear: creación de mazo.

Clase a testear: Mazo

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto Mazo
2. Recorrer el nuevo objeto Mazo con una de las cartas
3. Testear si esta carta se repite en el nuevo mazo y almacenar en repetida

Resultado esperado: la cantidad de cartas recorridas es igual a 51, lo que indica que nunca ha sido repetida.

Test case ID: TU 002

Descripción: se probará el funcionamiento del mezclado del mazo

Función a testear: mezclar el mazo

Clase a testear: Mazo

Preparaciones previas: N/A

Ejecución del test (pasos):

1. 2.Crear un nuevo objeto Mazo
2. Crear otro objeto Mazo
3. Mezclar uno de los mazos
4. Recorrer el mazo de cartas no mezcladas
5. Preguntar si la carta del mazo no mezclado es igual y se encuentra en la misma posición que en mazo mezclado (coincidencia)
6. Testear si la cantidad de coincidencias es menor a 5

Resultado esperado: la cantidad de coincidencias es menor a 5, lo que indica que el mazo se considera correctamente mezclado.

Test case ID: TU 003

Descripción: se probará el correcto funcionamiento de la pila en zona de juego, agregando una carta K, cuando la pila se encuentra vacía.

Función a testear: agregar una carta a pila en zona de juego

Clase a testear: PilaEnZonaDeJuego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto pila del tipo PilaEnZonaDeJuego
2. Crear una nueva carta K
3. Verificar si colocar la carta K en la PilaEnZonaDeJuego vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento válido.

Test case ID: TU 004

Descripción: se probará el correcto funcionamiento de la pila en zona de juego, agregando una carta que no es una K, cuando la pila se encuentra vacía.

Función a testear: agregar una carta a pila en zona de juego

Clase a testear: PilaEnZonaDeJuego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto pila del tipo PilaEnZonaDeJuego
2. Crear una nueva carta que no sea K.
3. Verificar si colocar la carta As en la PilaEnZonaDeJuego vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento NO válido.

Test case ID: TU 005

Descripción: se probará el correcto funcionamiento de la pila en zona de juego, agregando una carta Q de Picas, cuando en la pila encuentra una K de Corazones.

Función a testear: agregar una carta a pila en zona de juego

Clase a testear: PilaEnZonaDeJuego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto pila del tipo PilaEnZonaDeJuego
2. Crear una nueva carta K
3. Agregar a la PilaEnZonaDeJuego la carta K
4. Crear una nueva carta Q
5. Verificar si colocar la carta Q en la PilaEnZonaDeJuego vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento válido.

Test case ID: TU 006

Descripción: se probará el correcto funcionamiento de la pila en zona de juego, agregando una carta Q de Diamantes, cuando en la pila encuentra una K de Corazones.

Función a testear: agregar una carta a pila en zona de juego

Clase a testear: PilaEnZonaDeJuego

Preparaciones previas: N/A

Ejecución del test (pasos):

Crear un nuevo objeto pila del tipo PilaEnZonaDeJuego
Crear una nueva carta K
Agregar a la PilaEnZonaDeJuego la carta K
Crear una nueva carta Q
Verificar si colocar la carta Q en la PilaEnZonaDeJuego vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento NO válido.

Test case ID: TU 007

Descripción: se probará el correcto funcionamiento de la pila en zona de escaleras, agregando una carta As, cuando la pila se encuentra vacía.

Función a testear: agregar una carta a pila en zona de escaleras

Clase a testear: PilaEnZonaDeEscaleras

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto pila del tipo PilaEnZonaDeEscalera
2. Crear una nueva carta AS,
3. Verificar si colocar la carta As en la PilaEnZonaDeEscalera vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento válido.

Test case ID: TU 008

Descripción: se probará el correcto funcionamiento de la pila en zona de escaleras, agregando una carta que no es un As, cuando la pila se encuentra vacía.

Función a testear: agregar una carta a pila en zona de escaleras

Clase a testear: PilaEnZonaDeEscaleras

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto pila del tipo PilaEnZonaDeEscalera
2. Crear una nueva carta que no sea .
3. Verificar si colocar la carta As en la PilaEnZonaDeEscalera vacía es un movimiento válido.

Resultado esperado: obtenemos un movimiento NO válido.

Test case ID: TU 009

Descripción: se probará el correcto funcionamiento de la pila en zona de escaleras, agregando una carta Dos de Corazones, cuando en la pila encuentra un As de Corazones.

Función a testear: agregar una carta a pila en zona de escaleras

Clase a testear: PilaEnZonaDeEscaleras

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto pila del tipo PilaEnZonaDeEscalera
2. Crear una nueva carta As de Corazones
3. Agregar a la PilaEnZonaDeEscalera la carta As
4. Crear una nueva carta Dos de Corazones
5. Verificar si colocar la carta Dos de Corazones en la PilaEnZonaDeEscaleras

Resultado esperado: obtenemos un movimiento válido.

Test case ID: TU 010

Descripción: se probará el correcto funcionamiento de la pila en zona de escaleras, agregando una carta Dos de Diamantes, cuando en la pila encuentra un As de Corazones.

Función a testear: agregar una carta a pila en zona de escaleras

Clase a testear: PilaEnZonaDeEscaleras

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto pila del tipo PilaEnZonaDeEscalera
2. Crear una nueva carta As de Corazones
3. Agregar a la PilaEnZonaDeEscalera la carta As
4. Crear una nueva carta Dos de Diamantes
5. Verificar si colocar la carta Dos de Diamantes en la PilaEnZonaDeEscaleras

Resultado esperado: obtenemos un movimiento NO válido.

Test case ID: TU 011

Descripción: se probará el correcto funcionamiento de pilas en zona de juego, a partir de un nuevo juego creado

Función a testear: Crear Pila en zona de Juego

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto juego del tipo Juego
2. Verificar si la cantidad de pilas en la zona en la zona de juego es igual a 7
3. Verificar si la cantidad de cartas en cada pila es la correspondiente

Resultado esperado: La cantidad de cartas en las 7 pilas es correcta.

Test case ID: TU 012

Descripción: se probará el correcto funcionamiento de pilas en zona de escaleras, a partir de un nuevo juego creado

Función a testear: Crear Pila en zona de Escaleras

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto juego del tipo Juego
2. Verificar si la cantidad de pilas en la zona de escaleras es igual a 4
3. Verificar si la cantidad de cartas en cada pila es igual a 0

Resultado esperado: La cantidad de cartas en las 4 escaleras es correcta.

Test case ID: TU 013

Descripción: se probará el correcto funcionamiento de la acción de pedir una sola carta.

Función a testear: solicitud de una carta.

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):
1. Crear un nuevo objeto juego del tipo Juego
2. Pedir una carta
3. Verificar si la cantidad de cartas restantes en el mazo es de 23
4. Verificar si la cantidad de cartas en Cartas a Ubicar es 1

Resultado esperado: La cantidad de cartas en el mazo es de 23 y en Cartas a ubicar solo tenemos 1.

Test case ID: TU 014

Descripción: se probará el correcto funcionamiento del reinicio del mazo, el cual posee una totalidad de 24 cartas restantes una vez que las pilas en zona de juego fueron creadas.

Función a testear: Reiniciar Mazo

Clase a testear: Mazo

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto juego del tipo Juego
2. Crear un nuevo Mazo
3. Copiar el mazo de Juego a este mazo creado
4. Pedir las 24 cartas del mazo de Juego y reiniciarlo.
5. Comparar las coincidencias del nuevo mazo con el mazo de juego.
6. Verificar si la cantidad de coincidencias entre el mazo creado en el paso 2 y el mazo de juego es igual a 24
7. Verificar si la cantidad de cartas en la zona de cartas a ubicar es igual a 0

Resultado esperado: La cantidad de coincidencias es igual a 24, lo que significa que el mazo se reinicio en el orden correcto, y que no hay cartas en la zona de cartas a ubicar.

Test case ID: TU 015

Descripción: se agrega una carta As a una de las pilas de Escaleras.

Función a testear: Agregar carta/s a una Pila.

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto juego del tipo Juego
2. Crear una nueva Carta, cuyo valor es As del palo Corazones
3. Agregar la carta a la primer pila de Escaleras
4. Verificar si la carta en pila de Escaleras es el As de Corazones

Resultado esperado: La carta agregada en la pila de Escaleras es un As de Corazones

---

Test case ID: TU 016

Descripción: se agrega una carta K a cada una de las pilas de escaleras, simulando la victoria.

Función a testear: Comprobar Victoria

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto juego del tipo Juego
2. Agregar una carta K a cada una de las pilas de Escaleras
3. Se comprueba si finaliza la partida con victoria una vez que las pilas de Escaleras poseen una K como última carta

Resultado esperado: La partida finaliza con victoria

Test case ID: TU 017

Descripción: se verifica el correcto cálculo de la puntuación de la partida, en base a la bonificación por tiempo y a la cantidad de movimientos a zona de escalera realizados.

Función a testear: Cálculo de puntuación

Clase a testear: Juego

Preparaciones previas: N/A

Ejecución del test (pasos):

1. Crear un nuevo objeto juego del tipo Juego
2. Agregar una carta As a una de las pilas de escalera
3. Se comprueba el valor de la puntuación

Resultado esperado: La puntuación es igual a 1010, dado a la bonificación de tiempo menor a 5 minutos y a un movimiento a zona de escaleras realizado.

Ejecución de las pruebas:

La generación de las pruebas unitarias se realizó a través del framework Junit. Además se utilizó la herramienta Maven para facilitar el empaquetado, construcción y ejecución del software a la hora de correr las pruebas de manera automática. De esta manera, el proyecto cuenta con la estructura y configuraciones necesarias para que Maven funcione correctamente.
Las pruebas generadas se encuentran en la clase AppTest.

Para la ejecución de las pruebas de manera local, es necesario:

- Contar con el proyecto en un repositorio local.
- Tener instalada la herramienta Maven

Cumplidos estos requisitos, es necesario abrir una consola Git Bash, dentro del repositorio y ejecutar el comando “mvn test”. Se procederá a la compilación y ejecución de las pruebas de manera automática, notificando el resultado de las mismas.

Además también se utilizó la herramienta CircleCI, la cual nos permite realizar integración continua en nuestro proyecto. La función de esta herramienta es permitirnos correr las pruebas automáticas de manera remota. Se vinculó esta herramienta al repositorio remoto de GitHub, para que cada vez que se realiza un “push” a alguna de las branches del proyecto, las pruebas se ejecuten automáticamente. En caso de fallar alguna prueba, la herramienta notifica a los participantes del proyecto. Esto facilita la correcta integración de código por parte de varios programadores, evitando que nuevas modificaciones, hagan fallar otras funcionalidades.
Las pruebas ejecutadas hasta el momento y el resultado de las mismas, se puede observar a través del acceso a la herramienta CircleCI del proyecto que se encuentra en el Plan de Gestión de las Configuraciones.

4. Matriz de Trazabilidad

Requerimientos		Casos de uso	Casos de prueba	Clases	Casos de pruebas unitarias
Sistema	**Usuario**				
RF 1.1	RU 1 - RU 2 - RU 3	CU 1 - CU 2 - CU 3 - CU 4	TCF 001 - TCF 007 - TCF 016 - TCA 001 - TCA 002 - TCA 003 - TCA 004 - TCA 005	Juego	TU 016
RF 1.2	RU 1 - RU 2	CU 2 - CU 3	TCF 001	Mazo	TU 001 - TU 002
RF 1.3	RU 1 - RU 2 - RU 3	CU 2 - CU 3	TCF 007 - TCF 008 - TCF 009 - TCF 010 - TCF 015 -	Tablero	TU 011 - TU 012 - TU 014
RF 1.4.1	RU 1 - RU 2	CU 2	TCF 009 - TCF 010 - TCF 011- TCF 015	Juego	TU 013
RF 1.4.2	RU 1 - RU 2 - RU 3	CU 2	TCF 009 - TCA 002 - TCA 001	PilaEnZonaDeJuego	TU 003 - TU 004 - TU 005 - TU 006
RF 1.4.3	RU 1 - RU 2 - RU 3	CU 2	TCF 010 - TCA 003 - TCA 004	PilaEnZonaDeEscalera	TU 007 - TU 008 - TU 009 - TU 010
RF 2	RU 4	CU 4	TCF 013 - TCF 014	Juego	TU 017
RF 3	RU 2 - RU 4 - RU 5 - RU 6 - RU 7 - RU 8	CU 1 - CU 4 - CU 5 - CU 6 - CU 7 - CU 8	TCF 001 - TCF 002 - TCF 004 - TCF 005 - TCF 006 - TCF 012 - TCF 018 - TCF 019	Menu - Tablero - Estadísticas - Ventana Config	-
RF 4	RU 3	CU 2 - CU 3	TCF 009 - TCF 010 - TCF 011 - TCF 015	Tablero	-
RNF 1	-	CU 1	TNF 001	-	-

RNF 2	-	CU 1 - CU 2 - CU 3 - CU 4 - CU 5 - CU 6 - CU 7 - CU 8	TNF 002	-	-
RNF 3	-	CU 1	TNF 003	-	-
RNF 4	-	-	TNF 004	-	-