




Universidad del Cauca

Documento de Arquitectura de Software

Juego N en línea

Santiago Hyun Dorado
8-3-2018

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Historial de cambios

Fecha	Versión	Descripción	Autor
03/04/2018	1.0	Definición de la introducción y la representación de la arquitectura	Santiago Hyun Dorado


	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Tabla de contenido

1. Introducción	4
1.1. Propósito	4
1.2. Alcance	4
1.3. Definiciones, acrónimos y abreviaturas	4
1.4. Referencias	5
1.5. Resumen	5
2. Objetivos y restricciones de la Arquitectura	5
3. Representación de la Arquitectura	5
3.1. Vistas arquitecturales	5
3.1.1. Vista de escenarios	6
3.1.2. Vista lógica	6
3.1.3. Vista de desarrollo	6
3.1.4. Vista de proceso	7
3.1.5. Vista física	7
3.2. Patrones de diseño arquitectónicos	7
3.3. Estilo arquitectural	8
4. Descomposición de la arquitectura	9
4.1. Vista de escenarios (casos de uso)	9
4.2. Vista lógica (diseño)	13
4.3. Vista de proceso (actividades)	17
4.4. Vista de desarrollo (componentes)	20
4.5. Vista física (despliegue)	21
5. Rationale	22
6. Bibliografía	23



	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Tabla de Figuras

<i>Figura 1 Modelo 4+1 vistas</i>	6
<i>Figura 2 Patrón MVC</i>	7
<i>Figura 3 Estilo arquitectural clásico</i>	8
<i>Figura 4 Representación de la Arquitectura</i>	9
<i>Figura 5 Diagrama de Casos de uso</i>	10
<i>Figura 6 Diagrama de clases Modelo</i>	13
<i>Figura 7 Diagrama de Clases Controlador</i>	15
<i>Figura 8 Diagrama de Clases Vista</i>	16
<i>Figura 9 Diagrama de Clases Data</i>	17
<i>Figura 10 Secuencia Iniciar Juego</i>	18
<i>Figura 11 Secuencia Realizar Jugada</i>	19
<i>Figura 12 Secuencia Guardar Juego</i>	19
<i>Figura 13 Secuencia Mostrar Resultados</i>	20
<i>Figura 14 Diagrama de Paquetes</i>	20
<i>Figura 15 Diagrama de Componentes</i>	21
<i>Figura 16 Diagrama de Despliegue</i>	22

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

1. Introducción

Esta sección provee un resumen de todo el *Documento de Arquitectura de Software (SAD)* para el sistema Juego N en Línea. Brinda información sobre el propósito, el alcance, las referencias, las definiciones, acrónimos y abreviaturas utilizadas en todo el documento.

1.1. Propósito

El propósito principal del Juego N en línea es servir como base experimental en una investigación que tiene como finalidad determinar el valor de *ARAT* en el mantenimiento de una aplicación software desarrollada en Java. Este documento tiene como finalidad expresar en diferentes vistas el conjunto de decisiones arquitecturales que se han realizado en el desarrollo del sistema, con el objetivo de mostrar de distintas maneras los componentes y las conexiones que deben establecerse para cumplir con los escenarios requeridos.

1.2. Alcance

El alcance de este documento es explicar la arquitectura del Juego N en línea y las decisiones consideradas para realizar su diseño. La información contenida en este documento permite entender la estructura del sistema desde diferentes vistas, además, permite también describir explícitamente las razones de las decisiones arquitecturalmente importantes implementadas en el proceso de desarrollo.

1.3. Definiciones, acrónimos y abreviaturas

SAD: Software Architecture Document

GUI (Graphic User Interface): Interfaz Gráfica del Usuario

POO: Programación Orientada a Objetos

ISO (International Organization for Standardization): Organización Internacional de Normalización.

ARAT (Architectural Rationale Annotations Tool): Es una herramienta software basada en anotaciones de código fuente para documentar las razones arquitecturales de un sistema desarrollado en Java.

Java: Es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.


Factory Method: Método de fábrica, patrón de diseño creacional [1].

Observer: Observador, patrón de diseño de comportamiento [1].

Stakeholder: Según N. Rozanski et al [2] un stakeholder en una arquitectura de software “es una persona, grupo o entidad con intereses o preocupaciones sobre la realización de la arquitectura”.

MVC (Model View Controller): Es uno de los patrones de diseño arquitectural más utilizado en el desarrollo web, empezó como un Framework desarrollado por Trygve Reenskaug alrededor de 1970 [3].

OMG (Object Management Group): Es un consorcio internacional de estándares tecnológicos sin fines de lucro, de membresía abierta [4].

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

UML (Unified Modeling Language): El lenguaje de modelado unificado (UML) es un lenguaje gráfico creado por el OMG para visualizar, especificar, construir y documentar los artefactos de un sistema de software [5].

1.4. Referencias

DAS: <https://www.csun.edu/engineering-computer-science>

Java: https://www.java.com/es/download/faq/whatis_java.xml

Stakeholder: <https://www.viewpoints-and-perspectives.info/home/stakeholders/>

MVC: <https://msdn.microsoft.com/en-us/library/ff649643.aspx>

ISO: <https://www.iso.org/home.html>

OMG: <https://www.omg.org/>

UML: <http://www.uml.org/>

1.5. Resumen

Este documento se compone de las siguientes secciones:

- Sección 1: Provee un introducción relacionada con la arquitectura de software del Juego N en línea.
- Sección 2: Describe las objetivos a nivel general de la Arquitectura del sistema.
- Sección 3: Describe la representación de la arquitectura en diferentes vistas.
- Sección 4: Desglosa la representación de la arquitectura en diagramas para cada vista.
- Sección 5: Describe el Rationale arquitectónico del sistema.
- Sección 6: Referencias bibliográficas usadas en la creación de este documento.


2. Objetivos y restricciones de la Arquitectura

El Juego N en línea es un activo para un experimento el cual consta de comprobar si el uso de anotaciones de código como herramienta de documentación del *Rationale Arquitectónico* ayuda a mejorar la eficiencia y la efectividad en el mantenimiento de un sistema. El principal objetivo de la arquitectura del Juego N en línea es permitir la extensibilidad del sistema en términos de tipos de juegos y tablero, con el fin de conseguir aumentar la mantenibilidad y la capacidad de modificación del sistema, de tal forma que se pueda dar cumplimiento a uno de los principios de POO denominado “Abierto/Cerrado”. Otro objetivo de esta arquitectura es separar adecuadamente las responsabilidades en la interacción con el usuario con el fin de efectuar cambios visuales sin recompilar el código.

3. Representación de la Arquitectura

3.1. Vistas arquitecturales

La representación del sistema se puede ver desde diferentes puntos de vista, en el Modelo 4+1 Vistas propuesto por Kruchten p. et al [6] se definen 5 perspectivas que permiten extender el entendimiento de la arquitectura a las diferentes personas involucradas en el desarrollo de un sistema. En la Figura 1 se puede observar la organización de las vistas y su relación.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

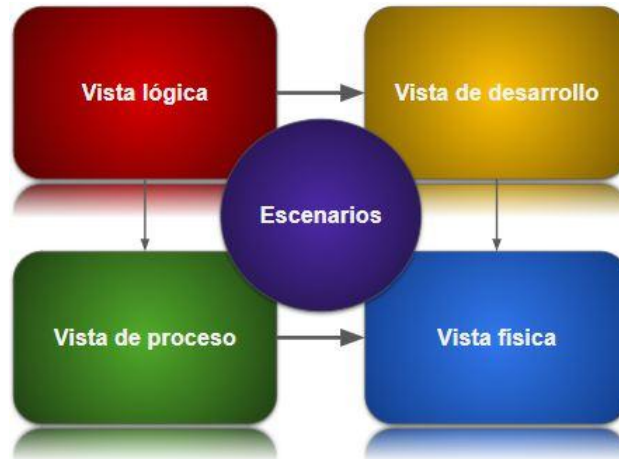


Figura 1 Modelo 4+1 vistas

3.1.1. Vista de escenarios

Conocida también como Vista de Casos de Uso, brinda información a nivel gráfico sobre los requerimientos funcionales de un sistema. Esta vista permite establecer las acciones y los roles que existen en el desarrollo de un sistema. Cada rol definido en el sistema debe tener una cantidad de casos de uso correspondientes a los requerimientos establecidos.

Audiencia: Stakeholders

Artefactos relacionados: Diagrama de Casos de Uso

3.1.2. Vista lógica

La Vista Lógica describe un sistema mediante el paradigma orientado a objetos, esta vista muestra las abstracciones diseñadas para estructurar el sistema junto con sus atributos y las operaciones que definen su comportamiento. Además muestra cómo se relacionan cada una de ellas para formar partes más grandes de un sistema complejo.

Audiencia: Diseñadores


Artefactos relacionados: Diagrama de clases

3.1.3. Vista de desarrollo

Conocida también como Vista de Componentes, representa la arquitectura en términos de componentes software generalmente constituidos por una o más clases, estos componentes se relacionan entre sí para definir el comportamiento esperado por los usuarios finales. Estos componentes suelen ser software, sin embargo, se pueden dar casos donde los algunos de los componentes sean físicos o de hardware.

Audiencia: Desarrolladores

Artefactos relacionados: Diagrama de componentes, Diagramas de paquetes

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

3.1.4. Vista de proceso

Esta vista permite observar las tareas individuales que realizan las abstracciones software para cumplir con un determinado proceso. Los requisitos del usuario se pueden comprender como tareas que un sistema debe cumplir, esta vista permite desglosar esas actividades en partes más pequeñas con el objetivo de determinar los procesos de manera gráfica y ordenada. La complejidad de los procesos determina el nivel de abstracción adecuado para realizar el diseño del modelo.

Audiencia: Integradores, Desarrolladores

Artefactos relacionados: Diagrama de actividades, Diagrama de secuencia

3.1.5. Vista física

Es importante conocer la distribución de los componentes físicos que constituyen un sistema, la Vista Física permite observar la topología y las comunicaciones que se establecen entre los nodos físicos que conforman la arquitectura hardware del sistema.

Audiencia: Líderes de despliegue

Artefactos relacionados: Diagrama de despliegue

En la sección [Descomposición de la arquitectura](#) se muestran los diagramas utilizados en el Juego N en línea para cada tipo de vista.

3.2. Patrones de diseño arquitectónicos

El patrón de diseño arquitectural implementado en el desarrollo del Juego N en línea es el patrón *Modelo Vista Controlador MVC*. Por medio de este patrón podemos realizar una separación de la de interacción del usuario con la Interfaz Gráfica en tres diferentes componentes. En la Figura 2 se puede observar la interacción entre los componentes del patrón.

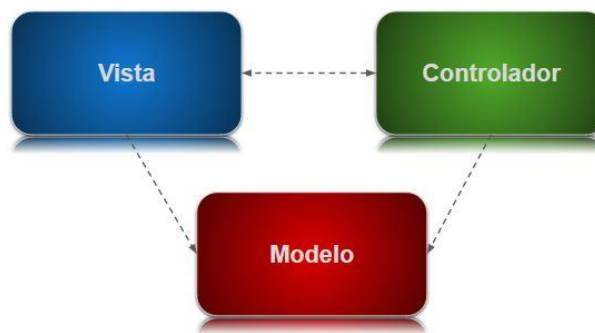



Figura 2 Patrón MVC

Modelo: Representa la información sobre el dominio del negocio, los datos y el comportamiento del sistema.

Vista: Es un componente que representa la *Interfaz Gráfica del Usuario GUI* con los datos del modelo, este componente captura las acciones y los datos y acude al controlador adecuado para procesarlos.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Controlador: Captura los datos ingresados por el usuario, manipula el modelo y actualiza la vista de forma correcta.

En la sección [Vista lógica](#) se muestra con más detalle las clases y los paquetes que componen este patrón de diseño arquitectural.

3.3. Estilo arquitectural

Los estilos arquitecturales son soluciones para resolver un problema relacionado con requerimientos NO funcionales o atributos de calidad, cada estilo arquitectural representa la experiencia de los diseñadores en un dominio de un problema específico. Las necesidades de calidad cambian de un sistema a otro por lo tanto los estilos arquitecturales son muy diversos y es muy difícil que contemplen todos los atributos de calidad requeridos [7]. Para el desarrollo del Juego N en Línea el requisito NO funcional de mayor prioridad es la mantenibilidad. Esta se traduce según la Norma ISO 25000 [8], en la facilidad para efectuar cambios, reutilizar componentes o agregar funcionalidad al sistema.

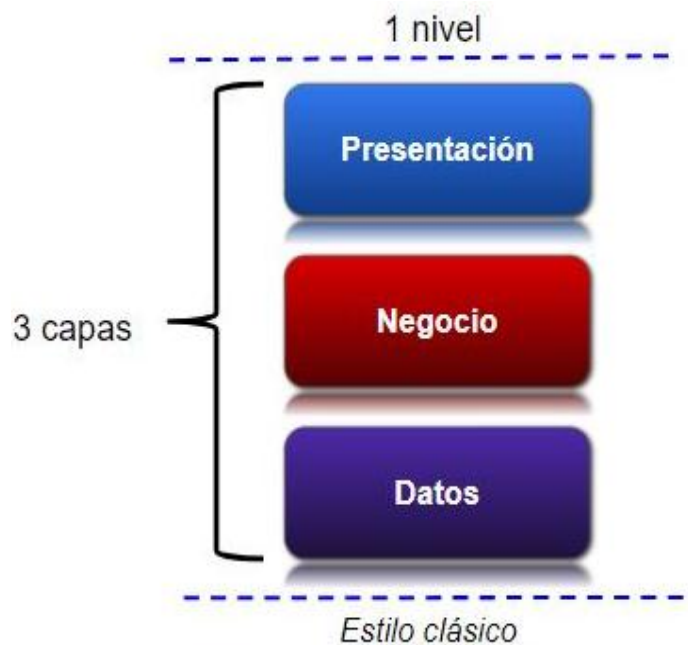



Figura 3 Estilo arquitectural clásico

En la Figura 3 se puede observar que el Juego N en línea sigue el estilo clásico de un nivel y tres capas propuesto por G. Florijn et al [7].

Capa de Presentación: Se encarga de controlar el flujo de trabajo entre el usuario y la GUI, también es responsable por mostrar la información del modelo. En el Juego N en línea esta capa contiene los componentes *Vista* y *Controlador* del *Patrón MVC*.

Capa de Negocio: Define los atributos y el comportamiento de las abstracciones de software en términos del contexto del negocio. En esta capa se puede encontrar las entidades que representan el *Modelo* del *Patrón MVC*.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Capa de datos: Esta capa es la encargada de acceder a los datos y establecer la persistencia de la información en el sistema.

En la Figura 4 se puede observar la organización de las capas y los componentes del MVC.




Figura 4 Representación de la Arquitectura

4. Descomposición de la arquitectura

La representación de la arquitectura se puede definir desde diferentes vistas adecuadas para una determinada audiencia. Estas vistas están relacionadas con uno o más diagramas UML que detallan con más información la arquitectura del sistema.

4.1. Vista de escenarios (casos de uso)

Las funcionalidades a nivel general se describen en la Figura 5. El jugador en el contexto del juego real es la persona que utiliza un tablero con n filas por m columnas y una cantidad x de fichas con el objetivo de realizar una secuencia de fichas continua del mismo color. Esta persona es la encargada de ubicar la ficha en una columna del tablero y soltarla, la evaluación del ganador se hace de manera visual, rectificando que haya una cantidad x de fichas contiguas del mismo color.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

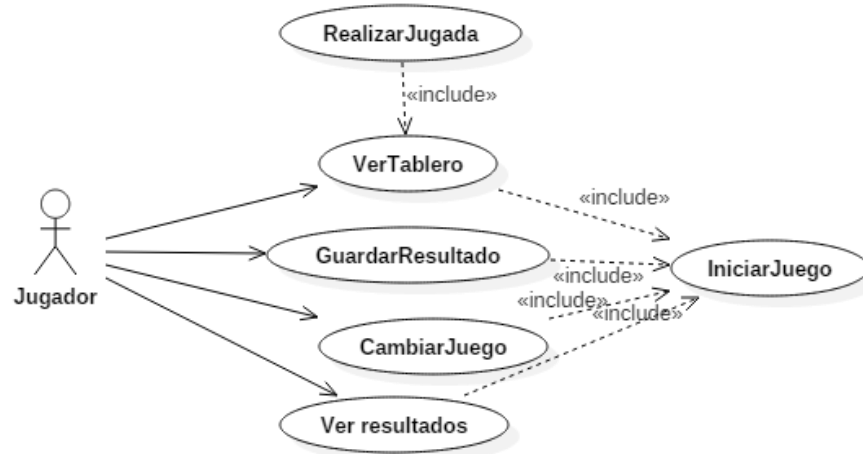



Figura 5 Diagrama de Casos de uso

ID	CU01
Nombre	Iniciar Juego
Actores	Jugador
Sinopsis	Iniciar los valores del juego y del tablero
Pre-condición	Se deben cargar los tipos del juego al componente gráfico por medio de reflexión
Curso típico de eventos	<ol style="list-style-type: none"> 1. Seleccionar el número de casillas contiguas para ganar 2. Seleccionar el tipo de juego que quiere jugar 3. Presionar el botón jugar para crear el juego y el tablero gráfico
Cursos alternativos	Si no presiona el botón jugar no se deben mostrar los datos del juego ni del tablero.
Extensiones	Ninguno
Prioridad	Alta
% Completado	100


ID	CU02
Nombre	Cambiar Juego
Actores	Jugador
Sinopsis	Cambiar de tipo de juego, volver a seleccionar el número de casillas y el tipo de juego al que se quiere cambiar
Pre-condición	Se debe haber iniciado el juego (CU01)
Curso típico de eventos	<ol style="list-style-type: none"> 1. El sistema crea la barra de opciones con el número de casillas para ganar y los tipos de juego que se pueden jugar. 2. El jugador selecciona el número de casillas contiguas para ganar 3. El jugador selecciona el tipo de juego al que quiere cambiar 4. El jugador presiona el botón reiniciar o el botón jugar para crear el juego y el tablero gráfico con los nuevos valores seleccionados
Cursos alternativos	Si no presiona el botón reiniciar el juego continuará con los mismos valores
Extensiones	Ninguno

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

Prioridad	Baja
% Completado	100
ID	CU02
Nombre	Cambiar Juego
Actores	Jugador
Sinopsis	Cambiar de tipo de juego, volver a seleccionar el número de casillas y el tipo de juego al que se quiere cambiar
Pre-condición	Se debe haber iniciado el juego (CU01)
Curso típico de eventos	<ol style="list-style-type: none"> 1. El jugador selecciona el número de casillas contiguas para ganar 2. El jugador selecciona el tipo de juego al que quiere cambiar 3. El jugador presiona el botón reiniciar o el botón jugar para crear el juego y el tablero gráfico con los nuevos valores seleccionados
Cursos alternativos	Si no presiona el botón reiniciar el juego continuará con los mismos valores
Extensiones	Ninguno
Prioridad	Baja
% Completado	100

ID	CU03
Nombre	Ver tablero
Actores	Jugador
Sinopsis	Vista gráfica con los datos del juego y del tablero. Esta vista del tablero se actualiza de forma automática cada vez que el jugador inicia un juego o realiza una jugada.
Pre-condición	Se debe haber iniciado el juego (CU01)
Curso típico de eventos	<ol style="list-style-type: none"> 1. El sistema crea el juego con la información que suministra el usuario en el caso de uso Iniciar Juego. 2. El sistema crea el tablero con la información del juego seleccionado y el número de casillas contiguas para ganar. 3. El sistema muestra la información del tablero en una matriz de botones. 4. El sistema muestra la información del juego en una barra informativa.
Cursos alternativos	El usuario no suministra la información para crear el Juego y presiona Jugar: El sistema debe tomar los valores que están por defecto en la barra de opciones
Extensiones	Ninguno
Prioridad	Baja
% Completado	100


ID	CU04
Nombre	Realizar Jugada
Actores	Jugador
Sinopsis	Se selecciona la ubicación en la cual el jugador decide realizar su jugada
Pre-condición	Debe poderse observar la matriz del tablero (CU03)
Curso típico de	<ol style="list-style-type: none"> 1. El jugador selecciona un botón de la matriz del tablero para indicar cuál

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

eventos	es la posición en la que quiere jugar. 2. El sistema determina si la ubicación es permitida para realizar la jugada. 3. El sistema ubica la ficha y actualiza el tablero con la nueva información. 4. El sistema muestra la información del juego en una barra informativa.
Cursos alternativos	El sistema determina que la ubicación no es permitida: El sistema debe ignorar las posiciones en las cuales no sea válida la jugada.
Extensiones	Ninguno
Prioridad	Baja
% Completado	100

ID	CU05
Nombre	Guardar Resultado
Actores	Jugador
Sinopsis	Se desea guardar el resultado del juego junto con la matriz del tablero en un archivo externo
Pre-condición	Debe existir un ganador
Curso típico de eventos	1. El sistema pregunta al jugador si desea guardar 2. El sistema guarda la información a través de un controlador, el cual se encarga de obtener el número del juego de un archivo que tiene un contador el cual representa el identificador de cada juego guardado en count-game.txt. 3. El controlador le notifica al tablero gráfico que ya se guardaron los resultados.
Cursos alternativos	El jugador no quiere guardar: El sistema debe reiniciarse con los valores por defecto de la barra de opciones. Error al guardar: El sistema le notifica al jugador que no se puede guardar el juego
Extensiones	Ninguno
Prioridad	Baja
% Completado	100

ID	CU06
Nombre	Ver Resultados
Actores	Jugador
Sinopsis	Lista con todos los resultados de los juegos guardados anteriormente
Pre-condición	Se debe haber iniciado el juego (CU01)
Curso típico de eventos	1. El jugador presiona el botón Resultados 2. El sistema carga la información de todos los resultados guardados previamente 3. El sistema crea el componente gráfico para mostrar los resultados 4. El jugador observa los resultados
Cursos alternativos	Error al cargar resultados: El sistema le notifica al jugador que hay un error al cargar los resultados

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

Extensiones	Ninguno
Prioridad	Baja
% Completado	100

4.2. Vista lógica (diseño)

En esta vista se muestra la organización de las clases dentro del patrón de diseño arquitectural *MVC*. En la Figura 6 se muestra el paquete correspondiente al componente del *Modelo*.

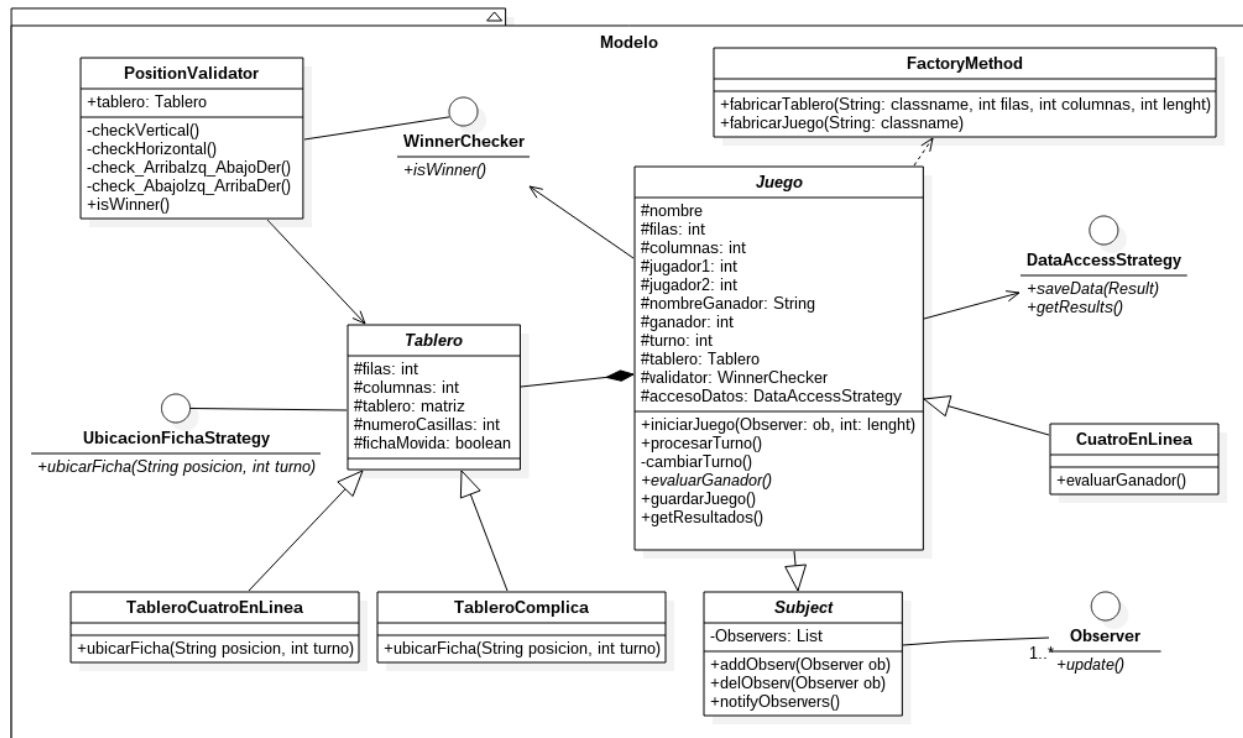



Figura 6 Diagrama de clases Modelo

Clase-Juego: Es una clase abstracta que representa las reglas de negocio del Juego N en línea, esta clase es encargada de iniciar el juego, procesar el turno y evaluar el ganador. Los atributos protegidos de esta clase permiten compartir información común entre los diferentes tipos de juego. El método evaluar ganador es abstracto ya que la forma de evaluar depende de las reglas para definir un ganador de cada tipo de juego, es por eso que debe ser implementado por las sub clases derivadas. Esta clase está compuesta por un tablero y tiene una relación directa con la clase PositionValidator que se encarga de hacer la verificación del tablero y determinar si hay un ganador.

Clase-Tablero: Es una clase abstracta que representa los tipos de tableros, sus atributos protegidos permiten que en las clases derivadas de esta solo se desarrolle la implementación del método abstracto ubicarFicha.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Clase-PositionValidator: Es una clase auxiliar que se encarga de verificar la existencia de un ganador en un tablero. Para la verificación se realiza un recorrido por toda la matriz identificando si existe un ganador. Se realiza una verificación horizontal, vertical, diagonal hacia arriba y diagonal hacia abajo para determinar si existe un ganador.

Interface-WinnerChecker: Es la interfaz que contiene el método abstracto para verificar si un jugador es ganador.

Clase-MetodoDeFabrica: Es una clase que representa el patrón de diseño *Factory Method*. Esta clase contiene un método llamado fabricarTablero, el cual recibe el nombre de un tablero, las filas, las columnas y el número de casillas para ganar, instancia el tablero con esta información y la retorna al solicitante. También tiene otro método para crear un Juego a través del nombre de la clase que se quiere instanciar. Esta clase utiliza la reflexión propia de Java a través de Class.forName().

Clase-Subject: Es una clase abstracta que representa al Observado del patrón de diseño *Observador*. Esta clase contiene una lista de los observadores que están pendientes de los cambios del Observado. Este se encarga de agregar, eliminar y notificar a todos los observadores que tiene en la lista.


Clase-CuatroEnLinea: Es una clase derivada de Juego, esta clase redefine los atributos filas y columnas en 6 y 7 respectivamente. Esta clase realiza la implementación del método abstracto evaluarGanador. La regla del juego para determinar un ganador es: si un jugador marca las n fichas del mismo color de manera consecutiva.

Clase-Complica: Es una clase derivada de Juego, esta clase redefine los atributos filas y columnas en 7 y 4 respectivamente. Esta clase realiza la implementación del método abstracto evaluarGanador. Las reglas del juego para determinar un ganador son: si los dos jugadores marcan n casillas en línea en la misma jugada se sigue con el juego, si un jugador marca las n fichas de manera consecutiva gana el juego.

Clase-TableroCuatroEnLinea: Es una clase derivada de Tablero, esta clase implementa el método abstracto ubicarFicha heredado de Tablero. La ubicación de las fichas en este tablero es igual a la forma de ubicar en el juego físico, se debe seleccionar una columna y posicionar la ficha en la última posición disponible de la columna seleccionada.

Clase-TableroComplica: Es una clase derivada de Tablero, esta clase implementa el método abstracto ubicarFicha heredado de Tablero. La ubicación de las fichas en este tablero se define a continuación, se debe seleccionar una columna y posicionar la ficha en la última posición disponible de la columna seleccionada, si no hay más posiciones en una columna todas las fichas se deben mover hacia abajo, eliminando la ficha inferior y añadiendo la nueva ficha en la cima de la columna.

Interface-Observer: Es una interfaz con un método abstracto que realiza las acciones pertinentes después de haber sido notificado. Este método abstracto puede ser implementado por diferentes clases, sin embargo, este no se realiza si el Observador no ha sido agregado a una lista de observadores de la clase Subject (Observado).

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Interface-UbicaciónFichaStrategy: Es la interfaz en la cual se define el método abstracto para ubicar la ficha en el tablero. El comportamiento de este método varía dependiendo de las reglas del juego y es desarrollado de diferentes maneras por las clases que realizan su implementación.

En la Figura 7 se observan las clases que componen el *Controlador* del patrón MVC.

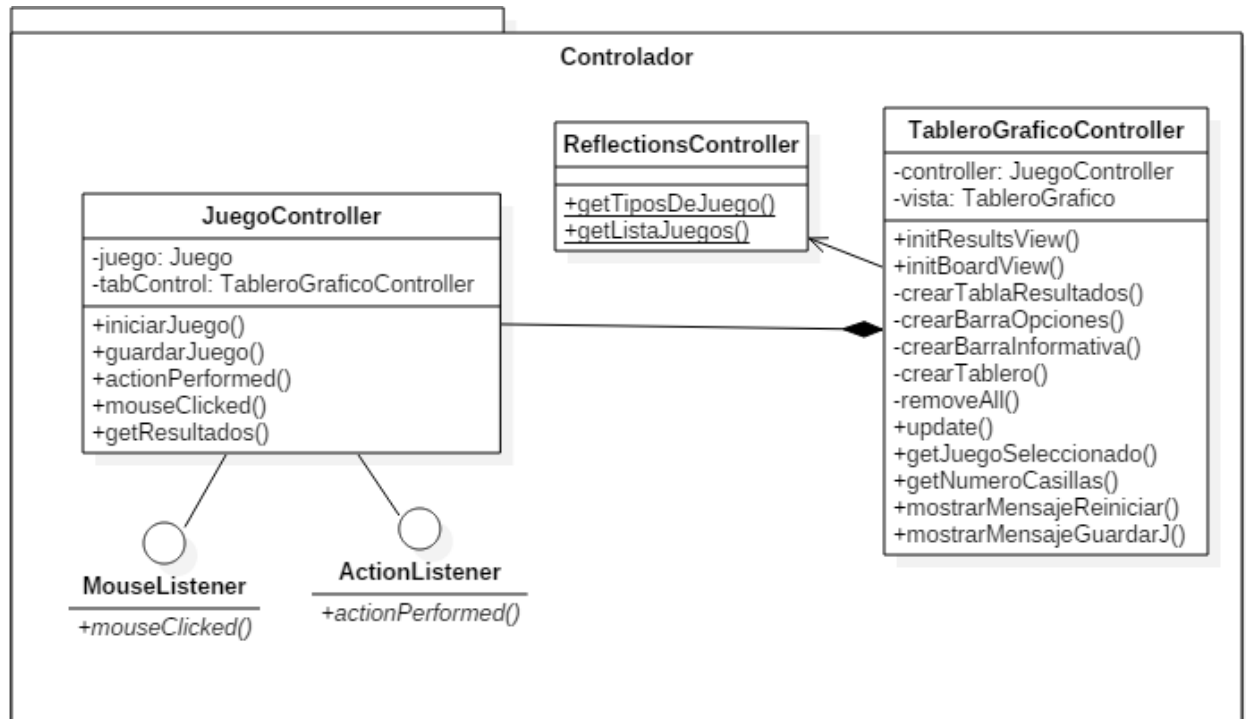



Figura 7 Diagrama de Clases Controlador

Clase-JuegoController: Esta clase implementa la interface **ActionListener** con el objetivo de manejar los eventos de iniciar juego, reiniciar, ver resultados y seleccionar posición. También se encarga de hacer el llamado al Juego para guardar los resultados de una partida. Esta clase se relaciona con el **TableroGraficoController** para que actualice los componentes de la vista.

Clase-TableroGraficoController: Esta clase se encarga de instanciar y actualizar los componentes gráficos del **TableroGrafico** y de la **ListaResultadosGraficos** de la vista. Se encarga de definir el controlador que va a manejar los eventos desplegados desde los botones de la vista. Esta clase implementa el método abstracto `update` del Observador, con el fin de actualizar la vista **TableroGrafico** cada vez que se inicia un juego o se realiza una jugada.

Clase-ReflectionsController: Esta clase tiene métodos estáticos para obtener los tipos de juego derivados de la clase **Juego** y la lista con los nombres de los juegos. Estos métodos se utilizan en la creación de los componentes gráficos que permiten la selección del juego por parte del usuario en el **TableroGrafico**. Esta clase hace llamados a la librería `reflections-0.9.9-RC1.jar` para obtener los subtipos a través de una determinada clase.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

Interface-ActionListener: Es una clase nativa de Java que permite establecer un canal de comunicación entre los componentes gráficos y las clases que se encargan de manejar los eventos generados por el usuario. Define un método abstracto `actionPerformed` para ser desarrollado por las clases que implementen la interface.

Interface-MouseListener: Es una clase nativa de Java que permite establecer un canal de comunicación con los botones del componente gráfico. Esta clase define cuatro métodos abstractos `mouseClicked`, `mouseEntered`, `mouseExited`, `mousePressed` y `mouseReleased` los cuales permiten manejar los diferentes eventos generados por el usuario en el `TableroGrafico`.

En la Figura 8 se muestran las clases que componen la *Vista* del patrón *MVC*.

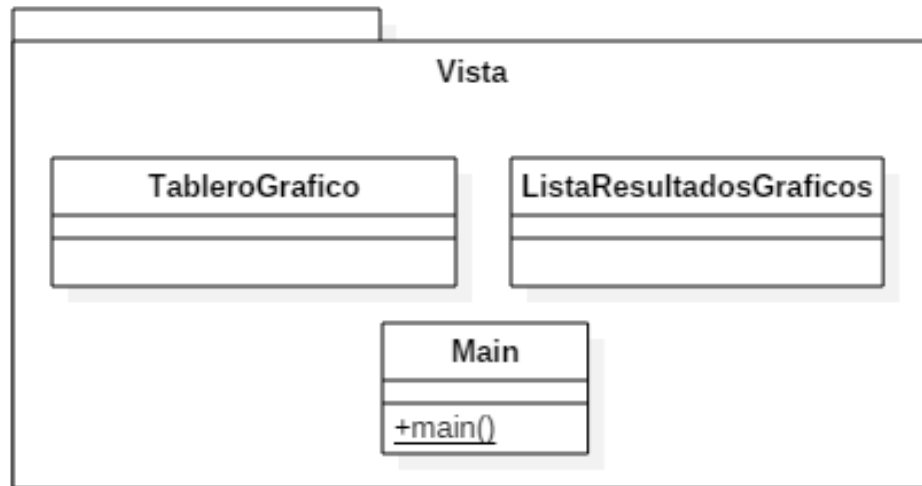



Figura 8 Diagrama de Clases Vista

Clase-TableroGrafico: Tiene atributos gráficos que representan la información del juego y del tablero. El tablero se representa mediante una matriz de botones, la selección del juego se hace mediante un Combobox y un Spinner para el número de casillas. La información del juego corresponde a una barra informativa la cual describe el juego seleccionado, el número de casillas y el turno de la jugada actual. Las acciones del juego se realizan desde 3 botones (Jugar, Reiniciar, Resultados)

Clase-ListaResultadosGraficos: Tiene atributos gráficos que representan los resultados guardados con anterioridad. La información de los resultados se despliega en una tabla con 6 columnas (número del juego, nombre del ganador, número de filas, número de columnas, nombre del juego y el número de casillas para ganar). También tiene un botón para volver al `TableroGrafico` nombrado Regresar.

Observe la Figura 10 para ver las clases que pertenecen al paquete de datos del Juego N en línea.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

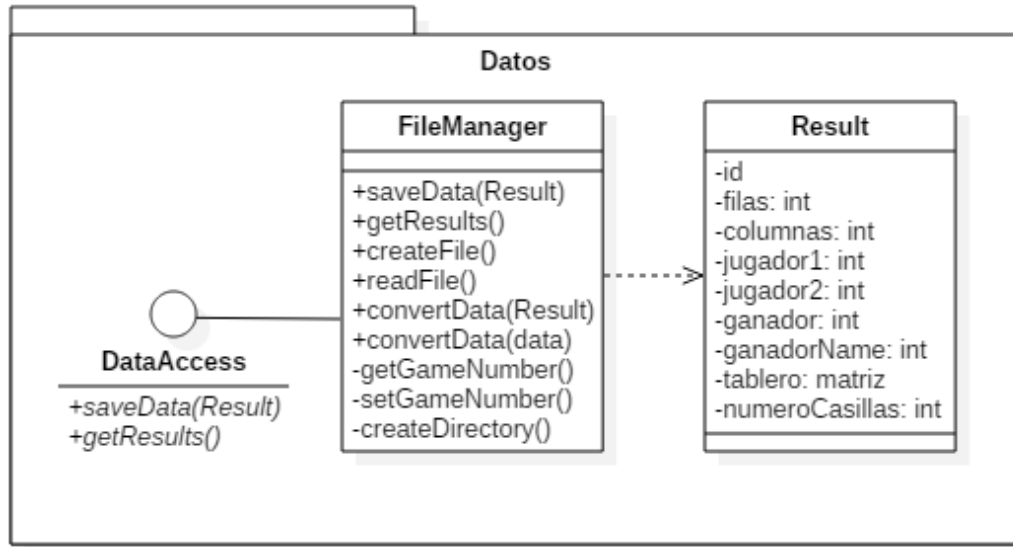


Figura 9 Diagrama de Clases Data


Clase-FileManager: Esta clase implementa los métodos de la interfaz DataAccess. Esta clase tiene como responsabilidad realizar toda la interacción con los archivos de los resultados del juego, permite escribir un archivo con un resultado, leer todos los resultados guardados, convertir una lista de datos en objetos Result y viceversa.

Clase-Result: Representa el resultado de un juego, esta clase tiene un identificador por juego y contiene la unión de los atributos del juego y del tablero.

Interface-DataAccessStrategy: Es la interfaz con la cual un juego puede delegar el proceso de guardar el resultado de un juego u obtener los resultados de los juegos guardados anteriormente.

4.3. Vista de proceso (actividades)

En la Figura 10 se puede observar el flujo de mensajes y los participantes que están involucrados en iniciar un juego. El TableroGrafico activa un evento que es manejado por el JuegoController, este obtiene la información del juego seleccionado y el número de casillas del TableroGrafico a través del TableroGraficoController, después instancia el tipo de juego seleccionado por medio del método fabricar Juego del FactoryMethod y hace el llamado a Juego para que inicie el juego, este agrega la vista TableroGrafico como observador, hace un llamado al método fabricar tablero, el cual crea un tablero por medio de reflexión y lo retorna de acuerdo al tipo de juego seleccionado; seguido de eso Juego instancia un AccessData y un PositionValidador al cual le pasa el Tablero que le retorna el FactoryMethod. Finalmente JuegoController le indica al TableroGraficoController que actualice los componentes de TableroGrafico con los datos del juego y el tablero seleccionado.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

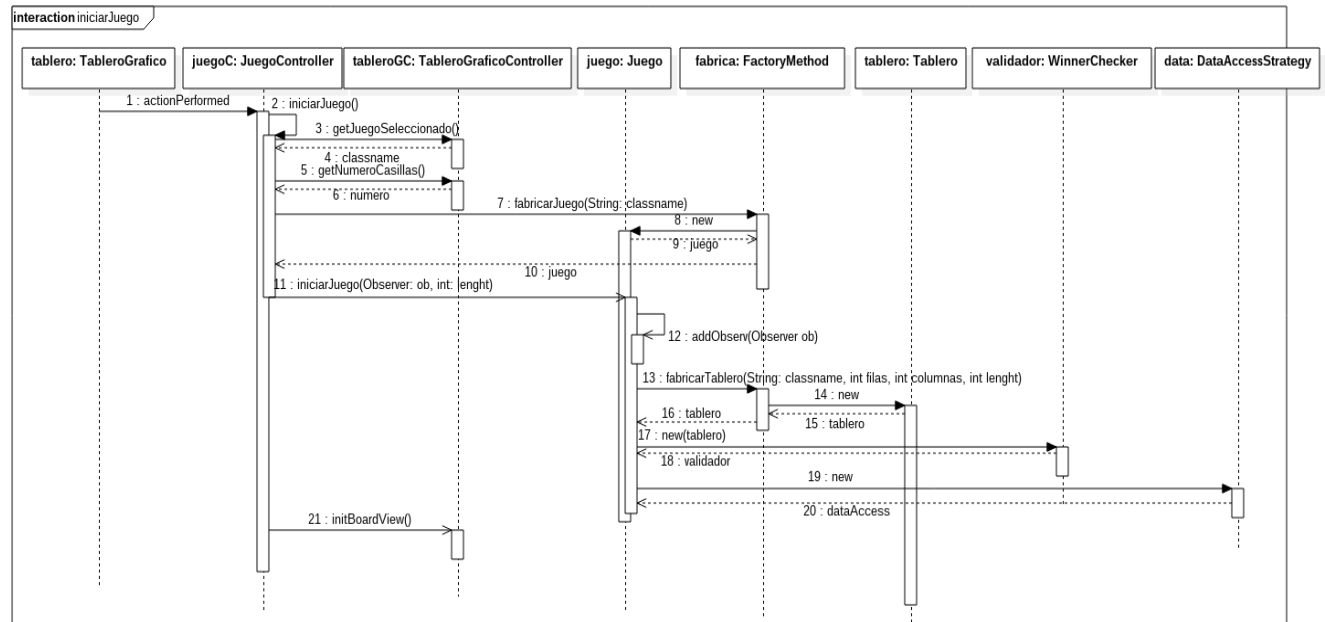



Figura 10 Secuencia Iniciar Juego

La Figura 11 muestra cómo interactúan los participantes del evento RealizarJugada. Esta secuencia comienza cuando la vista TableroGrafico captura el evento de la selección del botón en el cual el jugador quiere realizar la jugada. El JuegoController es el encargado de manejar este evento; obtiene el tablero de Juego, el turno actual y hace un llamado al Tablero para ubicar la ficha en la posición seleccionada y con el número del turno del jugador. Después el JuegoController notifica a su lista de observadores que se ha modificado el Juego y el TableroGraficoController procede a actualizar la vista TableroGrafico. Finalmente el JuegoController verifica si existe un ganador y de ser correcto envía un mensaje para informar al usuario y preguntar si desea guardar el juego.

La secuencia para guardar el resultado de un juego se muestra en la Figura 12. Después de que el TableroGraficoController envíe un mensaje al usuario a través del TableroGrafico y obtenga el nombre del jugador, el TableroGraficoController solicita al JuegoController que guarde el juego, este pasa el mensaje al Juego, el cual crea un Result, instancia los datos del resultado y solicita al AccessData que lo guarde.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software		Sistema: Juego N en línea

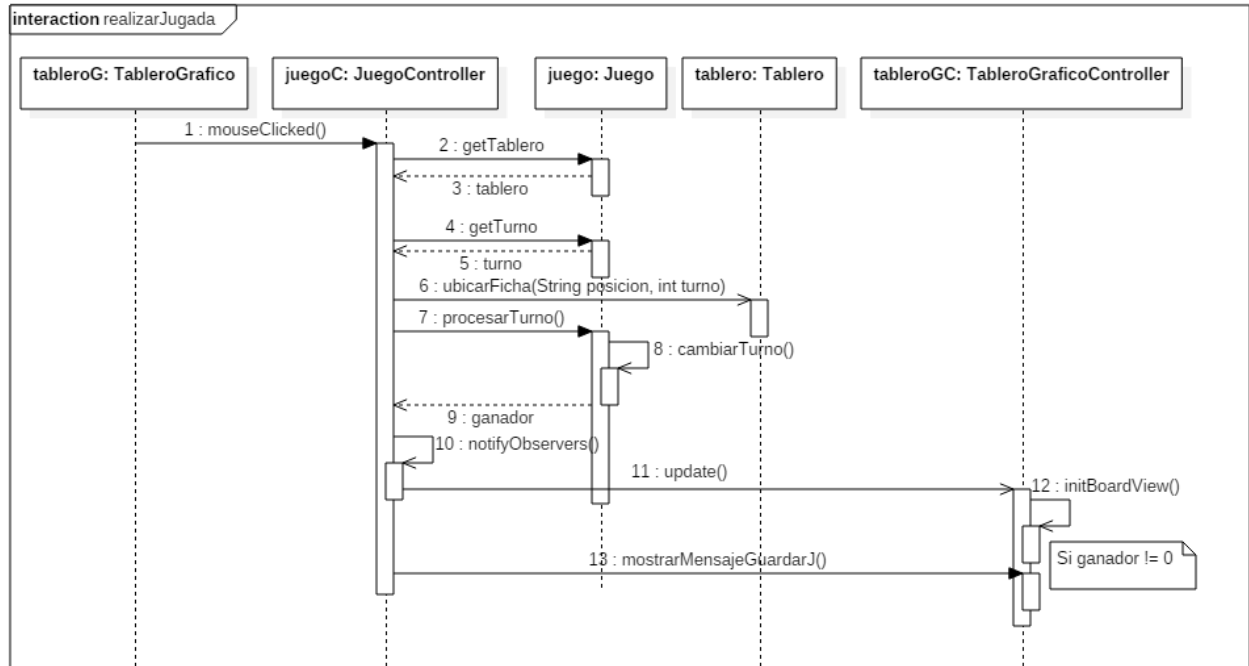


Figura 11 Secuencia Realizar Jugada

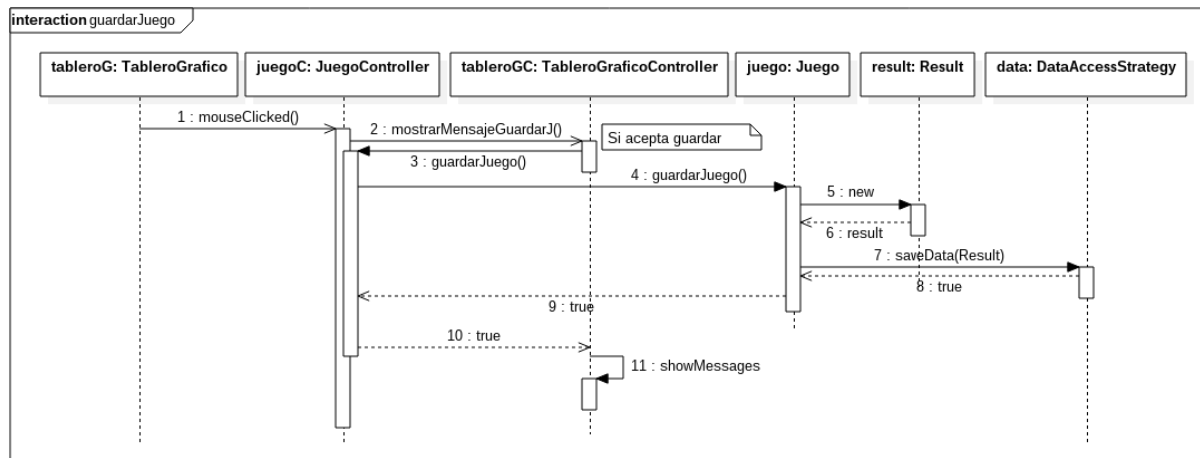



Figura 12 Secuencia Guardar Juego

En la Figura 13 se muestra cómo es el flujo de actividades que se realizan en la tarea de guardar resultado del juego. Este proceso inicia cuando el TableroGraficoController captura el evento de mostrar resultados generados por el botón Resultados de la vista TableroGrafico, este inicializa la vista ListaResultadosGraficos construyendo la tabla con los resultados de los juegos anteriores obtenidos mediante el DataAccess a través del JuegoController.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

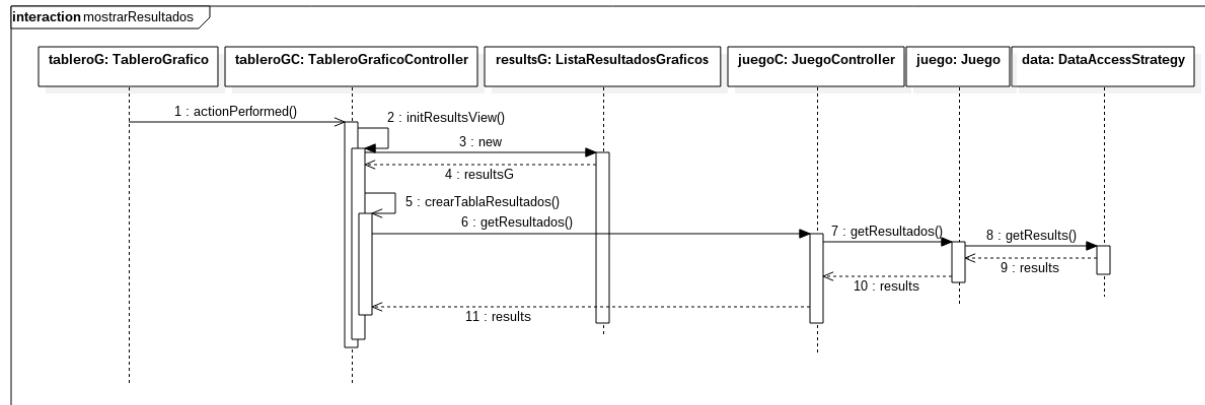


Figura 13 Secuencia Mostrar Resultados

4.4. Vista de desarrollo (componentes)

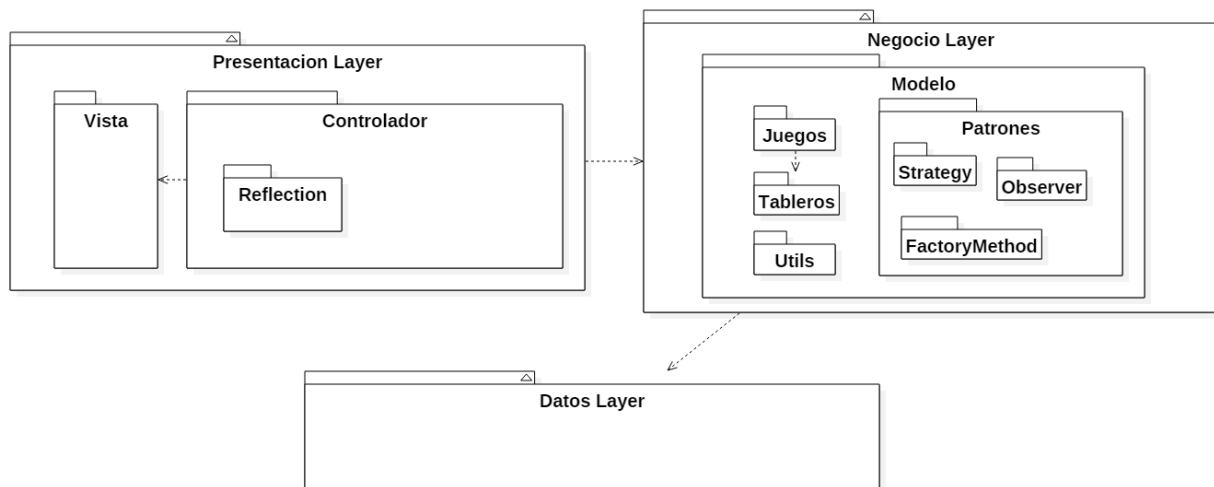



Figura 14 Diagrama de Paquetes

Los componentes del Juego N en línea se pueden observar desde diferentes perspectivas, en la Figura 14 podemos observar la organización de los paquetes entre las capas *Datos*, *Negocio* y *Presentación* con los componentes del patrón de diseño arquitectural *Modelo Vista Controlador*.

Los patrones de diseño utilizados en la arquitectura le dan estructura y comportamiento a la lógica del negocio, estos se encuentran en el paquete patrones para tener una idea de cuáles patrones se están utilizando en el sistema. En el paquete Utils se encuentra la clase PositionValidator que se encarga de verificar si existe un ganador. En los paquetes Juegos se encuentra la clase abstracta Juego y sus derivadas. En el paquete Tableros se encuentra la clase Tablero y sus clases derivadas. En el paquete Reflection se encuentra el ReflectionsController encargado de obtener los tipos de Juego en tiempo de

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

ejecución. En la capa de Datos se encuentra la interace de acceso a datos AccessData, una implementación de la interfaz llamada FileManager y la clase que representa los resultados del juego Result.

En la Figura 15 se puede observar la distribución de los componentes de manera que una o más clases conforman un componente. El componente Games representa la lógica de los juegos, este tiene una dependencia directa con el componente Boards el cual representa a los tableros y con el componente Patterns el cual representa los patrones de diseño estructurales y de comportamiento. El componente Boards tiene una relación directa con Patterns debido a que hace llamados a un patrón de diseño creacional (FabricaDeTableros). El componente Controllors representa los controladores encargados de modificar el modelo y actualizar las vistas, estos tienen una relación directa con Patterns debido a que necesita el comportamiento de un Observador para notificar los cambios, también tiene una relación con el componente AccesoDatos el cual representa las clases necesarias para crear persistencia en el sistema.

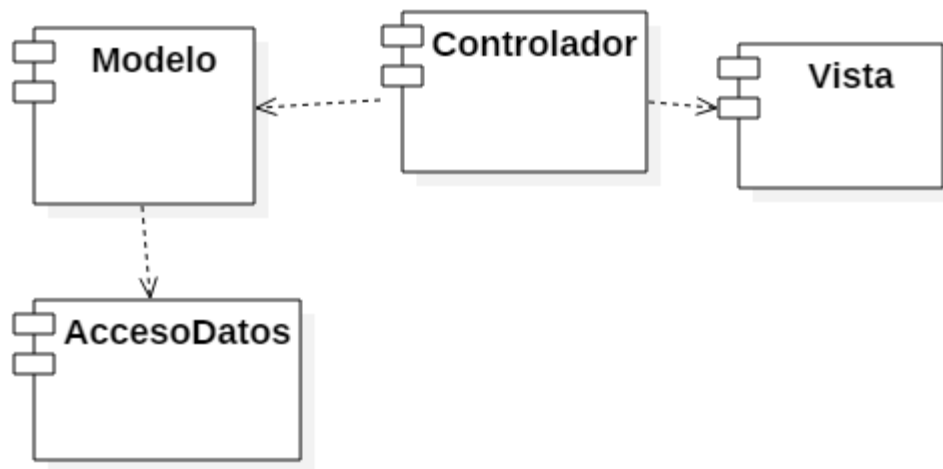



Figura 15 Diagrama de Componentes

Finalmente tenemos el componente View el cual está ligado al componente Controlador, ya que este ejecuta llamados al controlador a través de eventos generados por componentes gráficos de la vista (TableroGrafico).

4.5. Vista física (despliegue)

El sistema Juego N en línea es una aplicación monolítica y su despliegue requiere de un solo nodo, como se muestra en la Figura 16.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

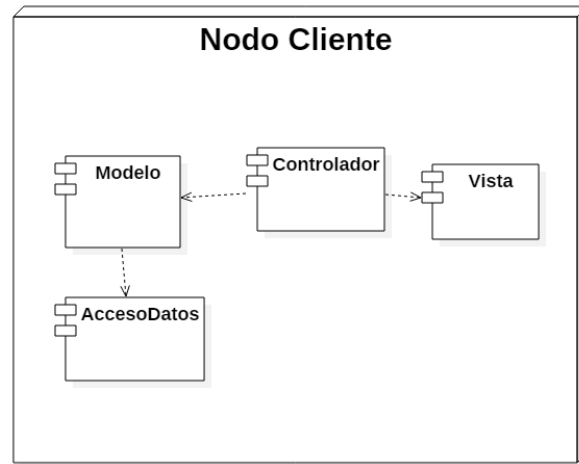


Figura 16 Diagrama de Despliegue


5. Rationale

En cuanto al SAD: la representación de la arquitectura se expresa en el modelo 4+1 vistas ya que este permite observar la arquitectura desde diferentes perspectivas y para diferentes audiencias, además para cada vista de este modelo existen diagramas bien definidos que facilitan la descripción de cada una de ellas.

Se utiliza el estilo arquitectural de tres capas para mantener separadas las responsabilidades en cuanto a persistencia, lógica del negocio y presentación de vistas; dado que la complejidad del sistema no es alta, se requiere de un estilo arquitectural simple en el que se puedan efectuar cambios sin desestabilizar el sistema y con el que se puedan detectar en menor tiempo los errores. El patrón de diseño arquitectural MVC separa la interacción del usuario con la interfaz gráfica permitiendo que se actualicen las vistas y los datos del juego de manera dinámica a través de un controlador. Este patrón favorece la Mantenibilidad mediante la separación de conceptos, con el objetivo de reutilizar código y organizar de manera adecuada las responsabilidades en la interacción con el usuario.

En el modelo:

El paquete *juegos* contiene una abstracción que representa las diferentes variantes del juego y las clases que heredan de esta abstracción, estas clases subtipos de Juego se crean por reflexión a través del FactoryMethodh en el método del JuegoController 'iniciarJuego' especificando la dirección de este paquete en una variable. El paquete *tableros* contiene una abstracción que representa las diferentes implementaciones de un tablero y las clases que heredan de esta abstracción, estas clases subtipos de Tablero también se crean por reflexión a través del FactoryMethodh en el método del Juego 'iniciarJuego' especificando la dirección de este paquete en una variable. Esto permite que el sistema se pueda extender en funcionalidades únicamente haciendo herencia de las abstracciones e implementando sus métodos abstractos, contribuyendo con la capacidad para ser modificado.

	Versión:	1.0
	Fecha:	03/04/2018
	Autor:	Santiago Hyun Dorado
Universidad del Cauca Documento de Arquitectura de Software	Sistema:	Juego N en línea

En el controlador:

El paquete Reflection permite obtener los tipos de las clases que heredan de Juego utilizando la librería desarrollada por [Google](https://www.google.com/). Esto permite que TableroGráficoController cree la barra gráfica de opciones con un ComboBox con el nombre de todas las clases que heredan de Juego, causando que en el proceso de agregación de tipos de juegos no se tenga que reescribir el código fuente actual y la vista se modifique a sí misma cuando se creen las nuevas implementaciones del Juego y el Tablero, contribuyendo en la capacidad para ser modificado.

El JuegoController se encarga de recibir las peticiones del usuario y manipular los datos del modelo y los componentes de las vistas. Esta clase permite que se realice la separación de la interacción del uso de la interfaz gráfica por parte del usuario, permitiendo que el flujo de las peticiones sea más limpio y se pueda hacer refactorización del código en menos tiempo.

6. Bibliografía

- [1] E. Robson and E. Freeman, *Head First Design Patterns Poster*. 2005.
- [2] N. Rozanski and E. Woods, *Software Systems Architecture*. 2005.
- [3] M. Fowler, *Patterns of Enterprise Application Architecture*, vol. 48, no. 2. 2002.
- [4] OMG, "Object Management Group." [Online]. Available: <https://www.omg.org/>.
- [5] Object Management Group, "Unified Modeling Language UML." [Online]. Available: <http://www.uml.org/>.
- [6] P. Krunten, "Architectural blueprints—the 4+ 1" view model of software architecture," *IEEE Softw.*, vol. 12, no. November, pp. 42–50, 1995.
- [7] G. Florijn, "Architectural styles and patterns," 2015.
- [8] Portal ISO, "ISO/IEC 25010." [Online]. Available: <http://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0>.