

# Arquitectura

May 5, 2022

## 1 Patrón MVC

El contenido principal del código está dividido en 3 paquetes. Para ello se ha utilizado el patrón modelo-vista-controlador (MVC), el cual permite separar el modelo (la parte lógica de la aplicación) de la vista (la salida de información que interactúa con el usuario) y del controlador (la entrada de información que actúa en el modelo).

Con respecto al modelo, se usa el paquete logic; con respecto al controlador, el modo consola usa el paquete Controller y el modo GUI aprovecha de la estructura controlador-vista proporcionada por Swing en los actionPerformed; por último, tenemos la vista en el paquete view, que a su vez se divide en subpaquetes para ofrecer distintas vistas según si se juega en consola o en GUI.

De esta forma se implementa el patrón MVC de una forma eficiente, organizada y respetando la encapsulación en todo momento.

## 2 Clases

Las clases del proyecto con las siguientes:

**GameBuilder:** Builder genérico de Game. Se encarga de generar GameClassic o GameTeams con sendas llamadas a GameClassicBuilder y GameTeamsBuilder.

**GameClassicBuilder:** Builder de GameClassic.

**GameTeamsBuilder:** Builder de GameTeams.

**Client:** Clase intermediara entre el ClientController y la vista.

**Command:** Clase genérica de comando que parsea los comandos introducidos para poder llamar a las clases específicas que gestionan dichos comandos.

**ExitCommand:** Clase que, dado el comando exit, termina el juego.

**HelpCommand:** Clase que, dado el comando help, muestra los comandos disponibles a ejecutar.

**PlaceCubeCommand:** Clase que, dado el comando de poner cubo, pide a Game que se coloque un cubo en la posición indicada.

**SaveCommand:** Clase que, dado el comando save, llama a SaveLoadManager para que guarde la partida en el instante en el que se llama al comando.

**Controller:** El controlados (en el sentido del MVC), intermediario de la vista y el modelo.

**SaveLoadManager:** Clase encargada de la carga, guardado y borrado de partidas y replays.

**Board:** Clase que define la estructura y lógica del tablero del juego.

**Color:** Clase de enumerados con métodos auxiliares para la representación de los colores.

**Cube:** Clase que representa un cubo, con su posición y jugador al que pertenece.

**Game:** clase abstracta que representa una visión general del juego, que luego se particulariza en GameClassic y GameTeams. Es un modelo (en el sentido del MVC).

**GameClassic:** Clase que implementa la lógica del juego en el modo Classic (jugadores individuales).

**Gameteams:** Clase que implementa la lógica del juego en el modo Teams (jugadores por equipos).

**Player:** Clase que representa a un jugador humano o IA y sus características (color, nombre, puntuación...).

**Replayable:** Interfaz que extiende de Reportable y que incluye un método toString() con el fin de encapsular Game para generar estados, que son usados por multitud de clases.

**Reportable:** Interfaz a través de la cual, implementada en clases, permite ejecutar un rport(), serialización del objeto en formato .json a través de unos estándares definidos.

**Rival:** Interfaz que permite unificar si se juega contra un jugador o contra un equipo mediante el concepto de rival.

**Shape:** Clase de enumerados con métodos auxiliares para la representación de las formas del tablero.

**Team:** Clase que representa a un equipo con el modo de juego GameTeams.

**TurnManager:** Clase que gestiona el paso de turnos.

**GameState:** Clase que representa el estado del juego en un momento determinado.

**Replay:** Clase a través de la cual se gestionan las replays, pudiendo por ejemplo avanzar y retroceder en las mismas.

**Rolit:** Clase de punto de partida de la aplicación. En esta se muestra el me´nu principal y se decide si acceder al modo GUI o al modo consola.

**HelpException:** Clase que representa la excepción ejecutada desde el comando HelpCommand.

**Server:** Clase que gestiona los clientes desde la perspectiva del servidos y que procesa los mensajes emitidos desde los clientes.

**ServerClient:** Clase que representa cada cliente desde la perspectiva del servidor.

**ServerClientThread:** Clase que interactúa con el cliente enviando y recibiendo mensajes.

**ServerView:** Clase que abre el diálogo a través del cuál el usuario puede abrir y detener un servidor.

**WaitPlayerThread:** Thread a través del cual el servidor va recibiendo las conexiones entrantes de los distintos clientes que se van conectando.

**GreedyStrategy** Estrategia IA consistente en una búsqueda superficial con el algoritmo Minimax.

**MinimaxStrategy:** Estrategia IA consistente una búsqueda en profundidad con el algoritmo Minimax y la poda alfa-beta.

**RandomStrategy:** Estrategia IA consistente en introducir un cubo de forma aleatoria siempre que este esté en una posición válida.

**SimplifiedBoard:** Encapsulación del tablero a través del cual la IA decide sus movimientos.

**Strategy:** Clase genérica estrategia IA de la cual heredan las estrategias específicas.

**Pair:** Clase que representa un par, utilidad básica para varias partes del código.

**StringUtils:** Clase que recopila distintas utilidades para trabajar con cadenas, básico para varias partes del código.

**RolitObserver:** Interfaz que declara las notificaciones a través de las cuales se trabaja en el patrón observador implementado para la estructura modelo-vista-controlador para particularidades del juego.

**ConsoleWindow:** Interfaz de ventana genérica para el modo consola

**DeleteGameWindow:** Ventana para la consola a través de la cual se puede borrar un juego guardado.

**LoadGameWindow:** Ventana para la consola a través de la cual se puede cargar un juego guardado.

**LoadReplayWindow:** Ventana para la consola a través de la cual se puede cargar un replay.

**MainBashWindow:** Ventana para la consola a través de la cual se muestra el menú principal.

**NewGameClassicWindow:** Ventana para la consola a través de la cual se crea un nuevo juego con modo GameClassic.

**NewGameTeamsWindow:** Ventana para la consola a través de la cual se crea un nuevo juego con modo GameTeams.

**NewGameWindow:** Ventana para la consola a través de la cual se crea un nuevo juego.

**PlayWindow:** Ventana para la consola a través de la cual se muestra el hilo de ejecución de una partida.

**SaveReplayWindow:** Ventana para la consola a través de la cual se puede guardar la partida en el estado actual.

**BoardGUI:** Clase que gestiona la representación del board en modo GUI

**BoardRenderer:** Renderer Swing que diseña los ComboBox que muestran las distintas formas de tablero que el usuario puede escoger.

**CeldaGUI:** Clase que representa una celda del tablero en modo GUI, responsabilizándose de su diseño y de recoger los clic del usuario.

**ChooseTeamFromServerDialog:** Diálogo GUI que, desde el cliente, se escoge en qué equipo desea unirse el usuario a la hora de conectarse a una partida en red con modo GameTeams.

**ColorRenderer:** Renderer Swing que diseña los ComboBox que muestran los distintos colores el usuario puede escoger.

**ControlPanel:** Panel Swing que muestra, según el caso, guardar partidas y el avance y retroceso en los replays. Se coloca en la parte superior de la ventana de la aplicación.

**DeleteGameDialog:** Diálogo GUI a través del cual el usuario puede eliminar partidas guardadas.

**JoinServerDialog:** Diálogo GUI a través del cual el cliente puede conectarse a un servidor introduciendo los datos de este.

**LoadFileDialog:** Diálogo GUI a través del cual el usuario puede cargar partidas guardadas.

**MainWindow:** Ventana principal del modo GUI sobre la cuál se colocan todos los componentes y nacen todos los diálogos.

**Observable:** Interfaz que declara los métodos de añadir y quitar observador para la implementación del patrón Observador.

**RankingTableModel:** Tabla Swing (GUI) que muestra la tabla de las puntuaciones en tiempo real de los usuarios de una partida.

**ReplayObserver:** Interfaz que declara las notificaciones a través de las cuales se trabaja en el patrón observador implementado para la estructura modelo-vista-controlador para particularidades del Replay.

**SaveReplayDialog:** Diálogo GUI a través del cual el usuario puede guardar la repetición de una partida finalizada.

**StatusBar:** Componente GUI situado en la parte inferior de la aplicación que muestra las notificaciones lanzadas por el modelo.

**TurnAndRankingBar:** Panel Swing (GUI) que junta tanto el ranking como el turno del usuario actual en la partida.

**CreateGameDialog:** Diálogo GUI a través del cual el usuario puede crear una partida.

**CreateGameWithPlayersDialog:** Clase que se encarga de los jugadores dentro del CreateGameDialog (GUI).

**CreatePlayersPanel:** Panel Swing (GUI) encargado de gestionar los jugadores introducidos en el CreateGameDialog.

**CreateTeamsPanel:** Panel Swing (GUI) encargado de gestionar los equipos introducidos en el CreateGameDialog.

**GameConfigurationPanel:** Panel Swing (GUI) que recoge la configuración de juego especificada en el CreateGameDialog.

**PlayerDataPanel:** Panel Swing (GUI) que recoge los datos introducidos de jugadores especificados en el CreateGameDialog.

**TeamDataPanel:** Panel Swing (GUI) que recoge los datos introducidos de equipos especificados en el CreateGameDialog.

**RolitBorder:** Extensión de Border (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitButton:** Extensión de JButton (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitCheckBox:** Extensión de JCheckBox (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitComboBox:** Extensión de JComboBox (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitIconButton:** Extensión de JButton con icono (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitPanel:** Extensión de JPanel (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitRadioButton:** Extensión de JRadioButton (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitTextArea:** Extensión de JTextArea (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitTextField:** Extensión de JTextField (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.

**RolitToolBar:** Extensión de JToolBar (Swing, GUI) adaptada para el diseño particular decidido para la aplicación.