FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Catan

[TB025] Paradigmas de la Programacion
Curso 01
Segundo cuatrimestre de 2025

Alumno	Padrón	Email
Goicoechea, Mariano Ezequiel	112138	mgoicoechea@fi.uba.ar
Espada, Joaquin	112033	jespada@fi.uba.ar
Mendoza, Carlos Franco	111758	cfmendoza@fi.uba.ar
Bogetti, Emiliano	109729	ebogetti@fi.uba.ar

Índice

1. Introducción **2**

2. Supuestos **2**

3. Diagramas de paquetes **2**

4. Diagramas de clase **2**

5. Detalles de implementación **4**

6. Excepciones **4**

7. Diagramas de secuencia **5**

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Paradigmas de la Programacion que consiste en desarrollar una aplicación que simule el juego de mesa 'Catan' en Java y JavaFX utilizando los conceptos del paradigmas de la orientación a objetos e interfaces graficas vistos hasta ahora en el curso.

2. Supuestos

En cuanto a los supuestos considerados en el modelo, se establecen una serie de restricciones y condiciones que guían el comportamiento del juego. Solo se permite mover al Ladrón cuando exista al menos un jugador susceptible de ser robado, y el mismo siempre inicia la partida ubicado en el Terreno Desierto. El comercio entre jugadores requiere que ambos ofrezcan recursos válidos: no es posible comerciar si alguno no aporta nada o entrega la totalidad de sus recursos. Por otro lado, no se impone un límite a la cantidad de construcciones que se pueden realizar en un turno, ni a la compra de cartas o a los intercambios permitidos. Finalmente, la partida contempla un mínimo de 3 jugadores y un máximo de 4, ajustándose a los parámetros definidos para el desarrollo.

3. Diagramas de paquetes

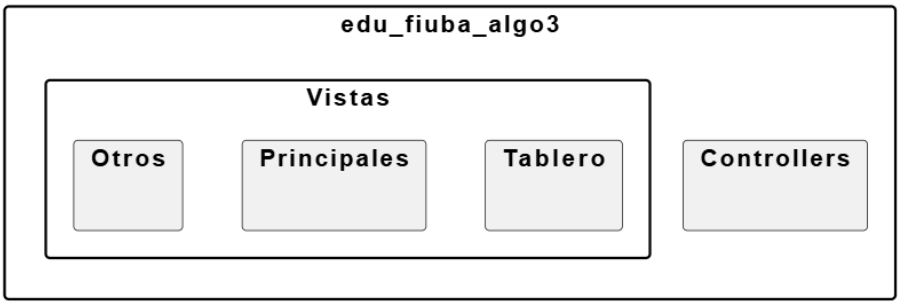

Figura 1: Diagrama de Paquete Vistas y Controllers.

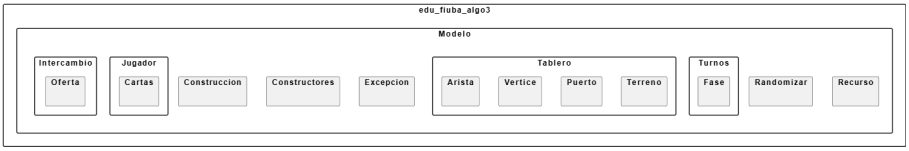

Figura 2: Diagrama de Paquete Modelo.

4. Diagramas de clase

En los siguientes diagramas se puede observar los sistemas de clases mas interesantes, las relaciones entre las mismas, ademas de sus respectivos atributos y metodos mas relevantes.

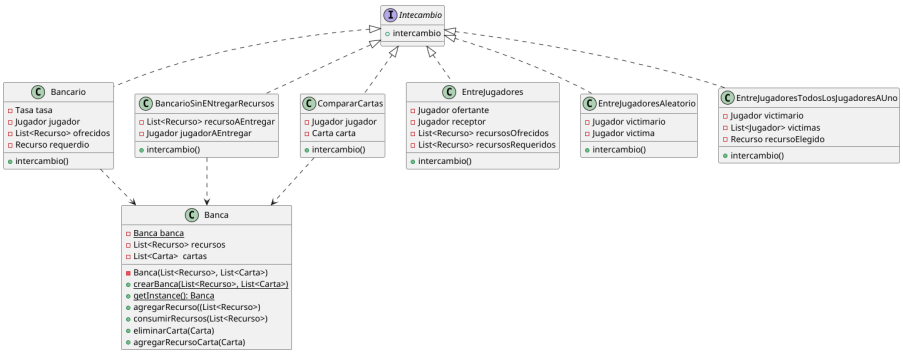

Figura 3: Diagrama de Intercambio.

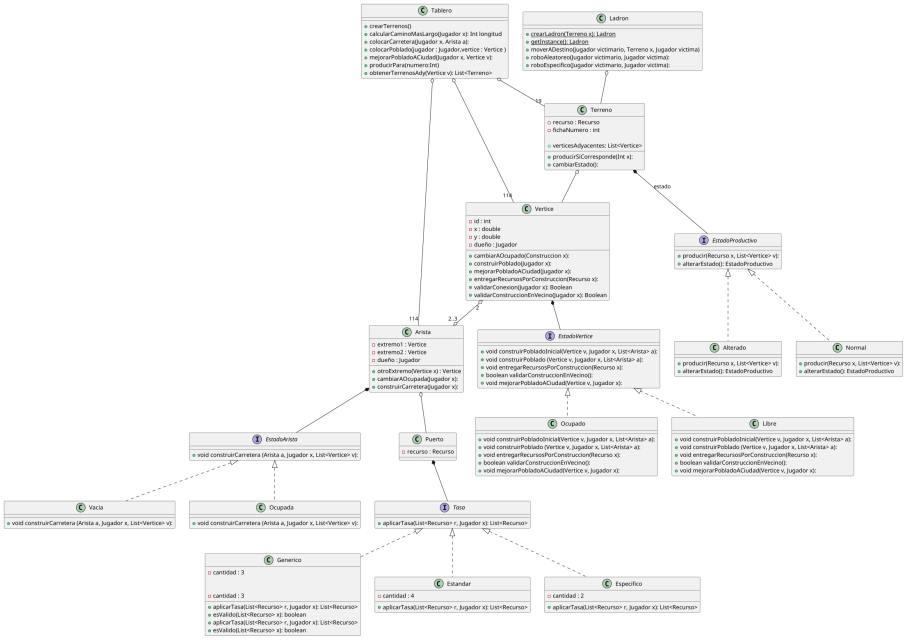

Figura 4: Diagrama de Tablero.

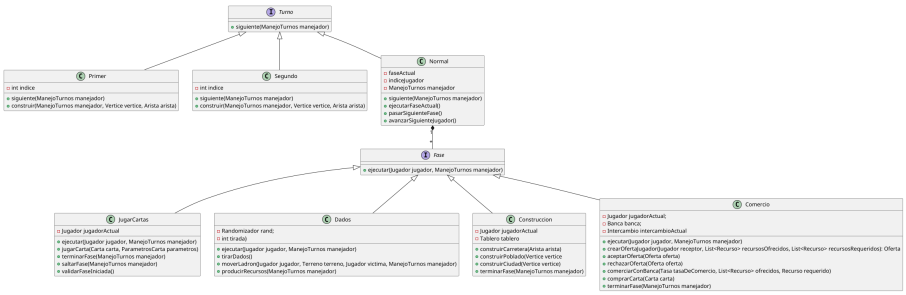

Figura 5: Diagrama de Turno.

5. Detalles de implementación

La implementación del proyecto se estructura aplicando diversos patrones de diseño y principios de organización del modelo. Las clases Juego, Banca y Ladron utilizan el patrón Singleton, garantizando una única instancia centralizada para cada una. Las clases Arista y Vertice implementan el patrón State, gestionando internamente su propio cambio de estado, mientras que Terreno aplica el patrón Strategy, cuyo comportamiento puede modificarse desde una clase externa. La clase Jugador delega el manejo de sus recursos, cartas y puntos para mantener una responsabilidad clara. El tablero se representa mediante un grafo, y la clase Juego es la encargada de coordinar el flujo de turnos y fases. Para la inicialización se emplean clases constructoras dedicadas a crear la Banca, el Juego, los Jugadores, el Tablero y los Terrenos. Tanto los recursos de los terrenos como las fichas se generan aleatoriamente en cada partida, mientras que los puertos mantienen posiciones fijas variando únicamente en las tasas asignadas. Finalmente, la interfaz gráfica se desarrolló siguiendo el patrón MVC, separando la lógica, la vista y el control de manera clara y modular.

6. Excepciones

AristaOcupadaNoSePuedeConstruir .

CartaDeshabilitada .

ElJuegoNoHaSidoCreadoAun .

ElLadronNoHaSidoCreadoAun .

LaBancaNoHaSidoCreadaAun .

MovimientoInvalido .

NoAlcanzanLosRecursos .

NoSePudoComprarUnaCartaSeAgotaronLasCartasDeLaBanca .

NoSePudoRealizarElIntercambioLaBancaNoTieneSuficientesRecursos .

NoSePuedeConstruirElJugadorNoEsDueñoDeLaAristaAdyacente .

NoSePuedeConstruirPorFaltaDeConexion .

NoSePuedeMejorarACiudad .

NoTieneRecursosSuficientesParaDescartar .

RecursoInvalido .

ReglaDeDistanciaNoValida .

VerticeOcupadoNoPuedeConstruir .

7. Diagramas de secuencia

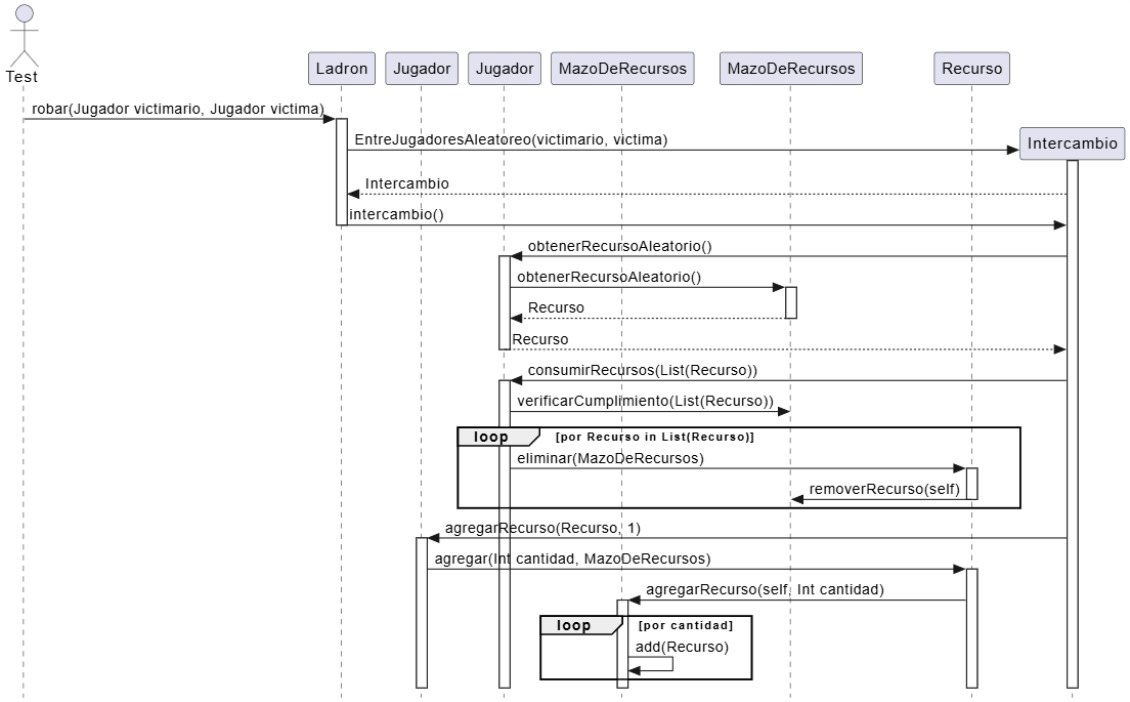

Figura 6: robar(Jugador victimario, Jugador victima)-

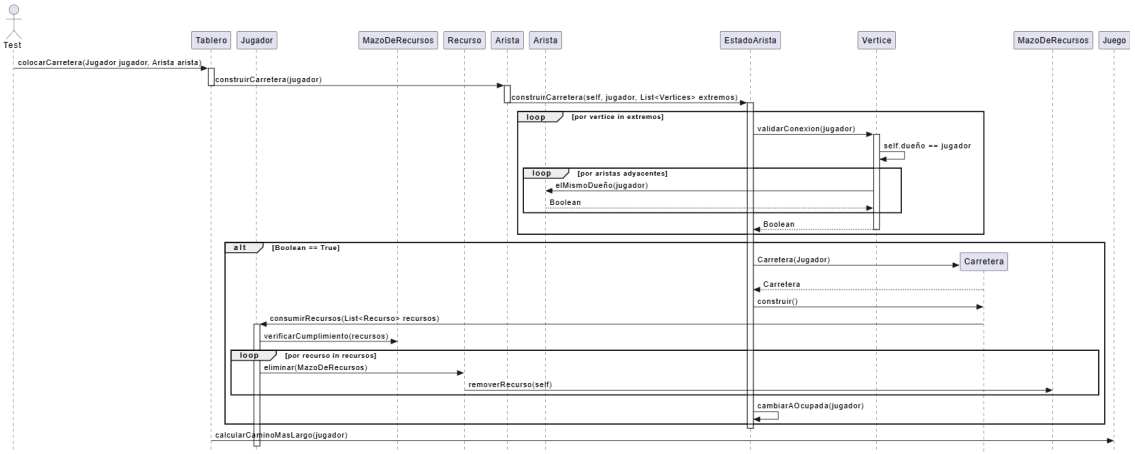

Figura 7: colocarCarretera(Jugador jugador, Arista arista)-