GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2016 / 2017

*TÍTULO DEL TFG*

# DESARROLLO DE UNA PLATAFORMA DE JUEGO

*nombre del documento*

# MEMORIA TFG

| DATOS DE LA ALUMNA O DEL ALUMNO | DATOS DEL DIRECTOR O DE LA DIRECTORA |
|---|---|
| NOMBRE JUAN ALBERTO | NOMBRE KOLDOBIKA |
| APELLIDOS MONASTERIO MONASTERIO | APELLIDOS GOJENOLA GALLETEBEITIA |
| DNI | DEPARTAMENTO LENGUAJES Y SISTEMAS INFORMÁTICOS |
| FDO.: | FDO.: |
| FECHA: | FECHA: |

Anexo II

## RESUMEN

El trabajo fin de grado que se presenta a continuación es una propuesta del alumno con el objetivo de formarse en el desarrollo de videojuegos.

En primer lugar, se ha desarrollado un motor gráfico en forma de librería en Java para facilitar el desarrollo de los videojuegos.

A continuación, utilizando el motor gráfico, se han implementado los juegos Ping-Pong y Space-Defense, basados en los juegos Pong y Space Invaders, dos de los más conocidos y exitosos de la historia. El Ping-Pong es un juego multijugador online en el que los jugadores pueden enfrentarse unos con otros en una partida en la que se simula el tenis de mesa. El Space-Defense es un juego de supervivencia individual al que se le han añadido varias mecánicas nuevas respecto al juego original. Ambos juegos tienen un estilo competitivo, por lo que cuentan con un apartado donde se muestra la clasificación con los mejores jugadores.

Finalmente se ha creado una plataforma de juego, desde la cual, los usuarios tendrán acceso a los juegos previo registro. Dispone de funcionalidades para gestionar la cuenta del jugador y acceder a los juegos presentes en un catálogo. Se espera que un futuro tanto el catálogo como las funcionalidades aumenten.

# ÍNDICE DE CONTENIDOS


***1. Introducción ..... 19***

1.1. Descripción y situación del trabajo ..... 19

1.2. Razones ..... 19

1.3. Planteamiento del problema ..... 20

1.4. Justificación ..... 20

***2. Planteamiento inicial ..... 22***

2.1. Objetivos ..... 22

2.2. Alcance ..... 22

2.2.1. EDT ..... 23

2.2.2. Tareas ..... 23

2.2.2.1. Gestión inicial ..... 24

2.2.2.2. Planificación ..... 24

2.2.2.3. Formación ..... 25

2.2.2.4. Análisis y diseño ..... 26

2.2.2.5. Implementación ..... 28

2.2.2.6. Pruebas ..... 29

2.2.2.7. Documentación ..... 30

2.3. Planificación temporal ..... 32

2.4. Herramientas ..... 35

2.4.1. Hardware ..... 35

2.4.2. Software ..... 35

2.4.3. Lenguajes de programación ..... 36

2.5. Gestión de riesgos ..... 38

2.6. Evaluación económica ..... 41

2.6.1. Gasto en personal ..... 41

2.6.2. Gasto hardware ..... 41

2.6.3. Gasto software ..... 42

2.6.4. Gasto en marketing ..... 42

2.6.5. Gasto total ..... 42

2.6.6. Beneficios ..... 43

***3. Antecedentes ..... 44***

3.1. Concepto de videojuego e historia ..... 44

3.2. Modelos de negocio ..... 44

3.3. Motores gráficos de videojuegos ..... 46

3.3.1. Unity ..... 46

3.3.2. GameMaker ..... 47

3.3.3. LibGDX ..... 47

3.4. Plataformas de juego ..... 47

3.4.1. Steam ..... 48

3.4.2. Battle.net ..... 48

*4. Captura de requisitos ..... 49*

4.1. Jerarquía de actores ..... 49

4.2. Casos de uso Space-Defense ..... 49

4.3. Casos de uso Ping-Pong ..... 50

4.4. Casos de uso plataforma de juego ..... 51

4.5. Modelo de dominio ..... 51

*5. Análisis y diseño ..... 53*

5.1. Motor gráfico ..... 53

5.1.1. Diagrama de clases ..... 53

5.1.2. Clases ..... 53

5.1.2.1. Módulo de entrada datos ..... 53

5.1.2.2. Módulo de estados ..... 54

5.1.2.3. Módulo principal ..... 55

5.1.2.4. Módulo de interfaz de usuario ..... 56

5.1.2.5. Módulo de sonido ..... 59

5.1.2.6. Módulo de utilidades ..... 60

5.2. Space-Defense ..... 62

5.2.1. Diagrama de clases ..... 62

5.2.2. Clases ..... 63

5.2.2.1. Módulo Estados ..... 63

5.2.2.2. Módulo Gameplay Objects ..... 65

5.2.2.3. Módulo principal ..... 72

5.2.2.4. Módulo Utilidades ..... 72

5.3. Ping-Pong ..... 73

5.3.1. Cliente ..... 73

5.3.1.1. Diagrama de clases ..... 73

5.3.1.2. Clases ..... 74

5.3.1.2.1. Módulo conexión servidor ..... 74

5.3.1.2.2. Módulo estados ..... 75

5.3.1.2.3. Módulo Gameplay Objects ..... 79

5.3.1.2.4. Módulo principal ..... 80

5.3.1.2.5. Módulo utilidades ..... 81

5.3.2. Servidor ..... 82

5.3.2.1. Diagrama de clases ..... 82

5.3.2.2. Clases ..... 82

5.3.2.2.1. Módulo gestión clientes ..... 82

5.3.2.2.2. Módulo servidor ..... 84

5.3.2.2.3. Módulo gestión principal ..... 85

5.3.2.2.4. Módulo gestión partidas ..... 86

5.3.2.2.5. Módulo gestión emparejamientos ..... 91

5.4. Plataforma de juego ..... 91

5.4.1. Diagrama de clases ..... 91

5.4.2. Clases ..... 92

5.5. Modelo relacional de bases de datos ..... 94

*6. Desarrollo ..... 95*

6.1. Motor gráfico ..... 95
6.1.1. Módulo de entrada datos ..... 95
6.1.2. Módulo de estados ..... 96
6.1.3. Módulo principal ..... 96
6.1.4. Módulo de interfaz de usuario ..... 98
6.1.5. Módulo de sonido ..... 99
6.1.6. Módulo de utilidades ..... 99
6.2. Space-Defense ..... 100
6.2.1. Uso de módulos ..... 100
6.2.2. Máquina estados ..... 100
6.2.3. Game Objects ..... 101
6.2.4. Fichero configuración ..... 102
6.3. Ping-Pong ..... 104
6.3.1. Uso de módulos ..... 104
6.3.2. Multithreading ..... 104
6.3.3. Diseño comunicación Cliente-Servidor ..... 105
6.3.3.1. Topología ..... 105
6.3.3.2. Protocolo ..... 107
6.3.3.3. Transmisión de mensajes ..... 107
6.3.4. Máquina estados ..... 108
6.3.4.1. Cliente ..... 108
6.3.4.2. Servidor ..... 110
6.3.5. Sistema Elo ..... 110
6.3.6. Sistema emparejamiento ..... 111
6.3.7. Fichero configuración ..... 113
6.3.7.1. Cliente ..... 113
6.3.7.2. Servidor ..... 114
6.4. Plataforma de juego ..... 116
6.4.1. Diseño ..... 116
6.4.2. Envío de correo ..... 116
6.4.3. Carga de juegos ..... 118
6.4.4. Ejecución juegos ..... 119
***7. Verificación y evaluación ..... 120***
7.1. Space-Defense ..... 120
7.1.1. Pruebas de rendimiento ..... 120
7.1.2. Pruebas de funcionalidad ..... 121
7.2. Ping-Pong ..... 124
7.2.1. Pruebas de rendimiento ..... 124
7.2.2. Pruebas de funcionalidad ..... 125
7.3. Plataforma de juego ..... 128
7.3.1. Pruebas de funcionalidad ..... 128
***8. Conclusiones y trabajo futuro ..... 131***
8.1. Cumplimiento de objetivos y plazos ..... 131
8.2. Conclusiones y trabajo futuro ..... 132
***9. Bibliografía ..... 133***
9.1. Libros ..... 133

9.2. Recursos electrónicos ..... 133

***10. Anexo I. – Casos de uso extendidos ..... 134***

10.1. Space-Defense ..... 134

10.2. Ping-Pong ..... 144

10.3. Plataforma de juego ..... 153

***11. Anexo II.- Diagramas de secuencia ..... 159***

11.1. Motor gráfico ..... 159

11.2. Space-Defense ..... 167

11.3. Ping-Ping ..... 183

11.3.1. Cliente ..... 183

11.3.2. Servidor ..... 195

11.4. Plataforma de juego ..... 203

# ÍNDICE DE FIGURAS

Ilustración 1: EDT ..... 23
Ilustración 2: GANTT ..... 34
Ilustración 3: COMPARATIVA MODELOS DE NEGOCIO (2) ..... 45
Ilustración 4: COMPARATIVA VENTAS DIGITALES Y FÍSICAS (3) ..... 47
Ilustración 5: JERARQUÍA DE ACTORES ..... 49
Ilustración 6: DIAGRAMA CASOS DE USO SPACE-DEFENSE ..... 49
Ilustración 7: DIAGRAMA CASOS DE USO PING-PONG ..... 50
Ilustración 8: DIAGRAMA CASOS DE USO PLATAFORMA DE JUEGO ..... 51
Ilustración 9: MODELO DE DOMINIO ..... 51
Ilustración 10: DIAGRAMA DE CLASES MOTOR GRÁFICO ..... 53
Ilustración 11: CONTROLADOR RATÓN ..... 53
Ilustración 12: CONTROLADOR TECLADO ..... 54
Ilustración 13: ISTATE ..... 54
Ilustración 14: STATE MACHINE ..... 55
Ilustración 15: GESTOR PRINCIPAL ..... 55
Ilustración 16: VENTANA PRINCIPAL ..... 56
Ilustración 17: GAME CANVAS ..... 56
Ilustración 18: UI OBJECT ..... 56
Ilustración 19: UI OBJECT INTERACTIVE ..... 57
Ilustración 20: IUI LISTENER ..... 57
Ilustración 21: TEXTO ..... 57
Ilustración 22: IMAGEN ..... 58
Ilustración 23: BOTÓN INTERFAZ ..... 58
Ilustración 24: ICONO SONIDO ..... 58
Ilustración 25: SONIDO ..... 59
Ilustración 26: GESTOR SONIDO ..... 59
Ilustración 27: GESTOR XML ..... 60
Ilustración 28: GESTOR CONEXIÓN BD ..... 60
Ilustración 29: GESTOR USUARIOS ..... 60
Ilustración 30: USUARIO ..... 61
Ilustración 31: DIAGRAMA DE CLASES SPACE-DEFENSE ..... 62

Ilustración 32: ESTADO MENÚ INICIO ..... 63
Ilustración 33: ESTADO INSTRUCCIONES ..... 63
Ilustración 34: ESTADO VER RANKING ..... 63
Ilustración 35: ESTADO JUGAR ..... 64
Ilustración 36: ESTADO PAUSA ..... 64
Ilustración 37: ESTADO FIN PARTIDA ..... 65
Ilustración 38: STATE MACHINE ..... 65
Ilustración 39: GAME OBJECT ..... 66
Ilustración 40: ANIMACIÓN ..... 66
Ilustración 41: EXPLOSIÓN ..... 66
Ilustración 42: JUGADOR ..... 67
Ilustración 43: GESTOR VIDA ..... 67
Ilustración 44: GESTOR PUNTUACIÓN ..... 68
Ilustración 45: ENEMIGO ..... 68
Ilustración 46: ENEMIGO A ..... 68
Ilustración 47: ENEMIGO B ..... 69
Ilustración 48: ENEMIGO C ..... 69
Ilustración 49: BALA ..... 69
Ilustración 50: BONIFICACIÓN ..... 70
Ilustración 51: BONIFICACIÓN INSTANTÁNEA ..... 70
Ilustración 52: BONIFIACIÓN TEMPORAL ..... 70
Ilustración 53: BONIFICACIÓN VIDA ..... 71
Ilustración 54: BONIFICACIÓN ESCUDO ..... 71
Ilustración 55: BONIFICACIÓN DISPARO RÁPIDO ..... 71
Ilustración 56: BONIFICACIÓN TRIPLE DISPARO ..... 71
Ilustración 57: GESTOR PRINCIPAL ..... 72
Ilustración 58: GESTOR USUARIOS ..... 72
Ilustración 59: USUARIO ..... 73
Ilustración 60: DIAGRAMA DE CLASES CLIENTE PING-PONG ..... 73
Ilustración 61: CLIENT ..... 74
Ilustración 62: CLIENT LISTENER ..... 74
Ilustración 63: CONTROL CLIENTE ..... 74
Ilustración 64: ISTATE ..... 75

Ilustración 65: STATE MACHINE ..... 75
Ilustración 66: ESTADO MENU INICIO ..... 76
Ilustración 67: ESTADO VER RANKING ..... 76
Ilustración 68: ESTADO INSTRUCCIONES ..... 76
Ilustración 69: ESTADO BUSCANDO PARTIDA ..... 77
Ilustración 70: ESTADO INICIANDO PARTIDA ..... 77
Ilustración 71: ESTADO JUGANDO ..... 78
Ilustración 72: ESTADO PAUSA ..... 78
Ilustración 73: ESTADO FIN PARTIDA ..... 79
Ilustración 74: BOLA ..... 79
Ilustración 75: PADDLE ..... 80
Ilustración 76: GESTOR PRINCIPAL ..... 80
Ilustración 77: GESTOR USUARIOS ..... 81
Ilustración 78: USUARIO ..... 81
Ilustración 79: DIAGRAMA DE CLASES SERVIDOR PING-PONG ..... 82
Ilustración 80: DATOS CLIENTE ..... 82
Ilustración 81: PERFIL JUGADOR ..... 83
Ilustración 82: CLIENTE ..... 83
Ilustración 83: LISTA CLIENTES ..... 83
Ilustración 84: SERVER LISTENER ..... 84
Ilustración 85: SERVER ..... 84
Ilustración 86: CONTROLADOR SERVIDOR ..... 85
Ilustración 87: GESTOR PRINCIPAL ..... 85
Ilustración 88: IESTADO ..... 86
Ilustración 89: ESTADO INICIANDO PARTIDA ..... 86
Ilustración 90: ESTADO JUGANDO ..... 86
Ilustración 91: ESTADO PAUSA ..... 87
Ilustración 92: ESTADO FIN PARTIDA ..... 87
Ilustración 93: BOLA ..... 88
Ilustración 94: JUGADOR ..... 88
Ilustración 95: MARCADOR ..... 89
Ilustración 96: TABLERO ..... 89
Ilustración 97: PARTIDA ..... 90

Ilustración 98: GESTOR PARTIDAS ..... 90
Ilustración 99: GESTOR EMPAREJAMIENTOS ..... 91
Ilustración 100: DIAGRAMA DE CLASES PLATAFORMA DE JUEGO ..... 91
Ilustración 101: USUARIO ..... 92
Ilustración 102: GESTOR USUARIOS ..... 92
Ilustración 103: JUEGO ..... 93
Ilustración 104: GESTOR JUEGOS ..... 93
Ilustración 105: MODELO RELACIONAL DE BASES DE DATOS ..... 94
Ilustración 106: MÁQUINA ESTADOS SPACE-DEFENSE ..... 101
Ilustración 107: MODELO CLIENTE-SERVIDOR (1) ..... 105
Ilustración 108: MODELO P2P (1) ..... 106
Ilustración 109: MÁQUINA ESTADOS CLIENTE PING-PONG ..... 108
Ilustración 110: MÁQUINA ESTADOS SERVIDOR PING-PONG ..... 110
Ilustración 111: ESTRUCTURA CARPETA JUEGOS ..... 118
Ilustración 112: COMPARATIVA PLANIFICACIÓN ..... 131
Ilustración 113: MENÚ SPACE-DEFENSE ..... 134
Ilustración 114: MENÚ SPACE-DEFENSE ..... 135
Ilustración 115: VENTANA RANKING SPACE-DEFENSE ..... 135
Ilustración 116: MENÚ SPACE-DEFENSE ..... 137
Ilustración 117: VENTANA JUEGO SPACE-DEFENSE ..... 137
Ilustración 118: VENTANA JUEGO SPACE-DEFENSE ..... 138
Ilustración 119: VENTANA JUEGO SPACE-DEFENSE ..... 139
Ilustración 120: VENTANA PAUSA SPACE-DEFENSE ..... 139
Ilustración 121: VENTANA PAUSA SPACE-DEFENSE ..... 140
Ilustración 122: VENTANA JUEGO SPACE-DEFENSE ..... 140
Ilustración 123: VENTANA JUEGO SPACE-DEFENSE ..... 141
Ilustración 124: VENTANA JUEGO SPACE-DEFENSE ..... 142
Ilustración 125: VENTANA GAME OVER SPACE-DEFENSE ..... 143
Ilustración 126: MENÚ PING-PONG ..... 144
Ilustración 127: MENÚ PING-PONG ..... 145
Ilustración 128: VENTANA RANKING PING-PONG ..... 145
Ilustración 129: MENÚ PING-PONG ..... 146
Ilustración 130: VENTANA BUSCAR PARTIDA ..... 147

Ilustración 131: VENTANA JUEGO PING-PONG ..... 147
Ilustración 132: VENTANA JUEGO PING-PONG ..... 148
Ilustración 133: VENTANA JUEGO PING-PONG ..... 149
Ilustración 134: VENTANA PAUSA PING-PONG ..... 149
Ilustración 135: VENTANA PAUSA PING-PONG ..... 150
Ilustración 136: VENTANA REANUDACIÓN PING-PONG ..... 150
Ilustración 137: VENTANA JUEGO PING-PONG ..... 150
Ilustración 138: VENTANA JUEGO PING-PONG ..... 151
Ilustración 139: VENTANA GAME OVER PING-PONG ..... 151
Ilustración 140: VENTANA SERVIDOR ..... 152
Ilustración 141: VENTANA INICIO SESIÓN ..... 153
Ilustración 142: VENTANA MENÚ PRINCIPAL ..... 153
Ilustración 143: VENTANA REGISTRO ..... 154
Ilustración 144: VENTANA INICIAR SESIÓN ..... 155
Ilustración 145: VENTANA RECUPERAR CONTRASEÑA ..... 155
Ilustración 146: VENTANA MENÚ PRINCIPAL ..... 156
Ilustración 147: VENTANA CAMBIAR CONTRASEÑA ..... 156
Ilustración 148: VENTANA MENÚ PRINCIPAL ..... 157
Ilustración 149: VENTANA CAMBIAR CORREO ..... 157
Ilustración 150: VENTANA MENÚ PRINCIPAL ..... 158
Ilustración 151: SECUENCIA GESTOR PRINCIPAL ..... 159
Ilustración 152: SECUENCIA INICIO JUEGO ..... 160
Ilustración 153: SECUENCIA GAME CANVAS ..... 160
Ilustración 154: SECUENCIA GESTOR USUARIOS ..... 161
Ilustración 155: SECUENCIA GESTOR SONIDO ..... 162
Ilustración 156: SECUENCIA ICONO SONIDO ..... 163
Ilustración 157: SECUENCIA STATE MACHINE ..... 164
Ilustración 158: SECUENCIA IMAGEN ..... 164
Ilustración 159: SECUENCIA BOTÓN INTERFAZ ..... 165
Ilustración 160: SECUENCIA TEXTO ..... 166
Ilustración 161: SECUENCIA STATE MACHINE ..... 167
Ilustración 162: SECUENCIA GESTOR PRINCIPAL ..... 168
Ilustración 163: SECUENCIA GESTOR USUARIOS ..... 168

Ilustración 164: SECUENCIA USUARIO ..... 169
Ilustración 165: SECUENCIA ESTADO MENÚ INICIAL ..... 169
Ilustración 166: SECUENCIA ESTADO VER RANKING ..... 170
Ilustración 167: SECUENCIA ESTADO INSTRUCCIONES ..... 170
Ilustración 168: SECUENCIA ESTADO JUGAR ..... 171
Ilustración 169: SECUENCIA ESTADO PAUSA ..... 171
Ilustración 170: SECUENCIA ESTADO FIN PARTIDA ..... 172
Ilustración 171: SECUENCIA JUGADOR ..... 173
Ilustración 172: SECUENCIA SISTEMA SPAWN ..... 174
Ilustración 173: SECUENCIA EXPLOSIÓN ..... 174
Ilustración 174: SECUENCIA ENEMIGO ..... 175
Ilustración 175: SECUENCIA ENEMIGO A ..... 176
Ilustración 176: SECUENCIA ENEMIGO B ..... 177
Ilustración 177: SECUENCIA ENEMIGO C ..... 178
Ilustración 178: SECUENCIA BALA ..... 179
Ilustración 179: SECUENCIA BONIFICACIÓN ..... 180
Ilustración 180: SECUENCIA BONIFICACIÓN TEMPORAL ..... 180
Ilustración 181: SECUENCIA BONIFICACIÓN DISPARO RÁPIDO ..... 181
Ilustración 182: SECUENCIA BONIFICACIÓN ESCUDO ..... 181
Ilustración 183: SECUENCIA BONIFICACIÓN TRIPLE DISPARO ..... 182
Ilustración 184: SECUENCIA BONIFICACIÓN VIDA ..... 182
Ilustración 185: SECUENCIA STATE MACHINE ..... 183
Ilustración 186: SECUENCIA GESTOR PRINCIPAL ..... 184
Ilustración 187: SECUENCIA GESTOR USUARIOS ..... 184
Ilustración 188: SECUENCIA USUARIO ..... 185
Ilustración 189: SECUENCIA CONTROL CLIENTE ..... 185
Ilustración 190: SECUENCIA ESTADO MENÚ INICIO ..... 186
Ilustración 191: SECUENCIA ESTADO VER RANKING ..... 187
Ilustración 192: SECUENCIA ESTADO INSTRUCCIONES ..... 188
Ilustración 193: SECUENCIA ESTADO BUSCANDO PARTIDA ..... 189
Ilustración 194: SECUENCIA ESTADO INICIANDO PARTIDA ..... 190
Ilustración 195: SECUENCIA ESTADO JUGANDO ..... 191
Ilustración 196: SECUENCIA ESTADO PAUSA ..... 192

Ilustración 197: SECUENCIA ESTADO FIN PARTIDA ..... 193
Ilustración 198: SECUENCIA PADDLE..... 194
Ilustración 199: SECUENCIA INICIAR SERVIDOR ..... 195
Ilustración 200: SECUENCIA CLIENTE CONECTADO ..... 195
Ilustración 201: SECUENCIA CLIENTE DESCONECTADO..... 195
Ilustración 202: SECUENCIA MENSAJE CLIENTE ..... 196
Ilustración 203: SECUENCIA CLIENTE..... 196
Ilustración 204: SECUENCIA GESTOR EMPAREJAMIENTOS ..... 197
Ilustración 205: SECUENCIA GESTOR PARTIDAS ..... 198
Ilustración 206: SECUENCIA PARTIDA ..... 198
Ilustración 207: SECUENCIA ESTADO INICIANDO PARTIDA ..... 199
Ilustración 208: SECUENCIA ESTADO JUGANDO..... 200
Ilustración 209: SECUENCIA ESTADO PAUSA ..... 201
Ilustración 210: SECUENCIA ESTADO FIN PARTIDA ..... 202
Ilustración 211: SECUENCIA REGISTRARSE ..... 203
Ilustración 212: SECUENCIA RECUPERAR CONTRASEÑA..... 204
Ilustración 213: SECUENCIA INICIAR SESIÓN ..... 204
Ilustración 214: SECUENCIA CONSTRUCTORA VISTA PRINCIPAL..... 205
Ilustración 215: SECUENCIA CAMBIAR CORREO ..... 205
Ilustración 216: SECUENCIA CAMBIAR CONTRASEÑA..... 206
Ilustración 217: SECUENCIA INICIAR JUEGO..... 206

# ÍNDICE DE TABLAS

Tabla 1: DESCRIPCIÓN TAREA ..... 23
Tabla 2: TAREA DEFINIR OBJETIVOS ..... 24
Tabla 3: TAREA REUNIÓN CON El DIRECTOR ..... 24
Tabla 4: TAREA IDENTIFICAR TAREAS ..... 24
Tabla 5: TAREA ESTIMAR TIEMPOS ..... 25
Tabla 6: TAREA PLANIFICACIÓN TEMPORAL ..... 25
Tabla 7: TAREA BÚSQUEDA DE RECURSOS ..... 25
Tabla 8: TAREA APRENDER ARQUITECTURA DE REDES ..... 26
Tabla 9: TAREA APRENDER FUNCIONAMIENTO MOTORES GRÁFICOS ..... 26
Tabla 10: TAREA IDENTIFICAR FUNCIONALIDADES ..... 26
Tabla 11: TAREA DISEÑO INTERFAZ ..... 27
Tabla 12: TAREA CASOS DE USO ..... 27
Tabla 13: TAREA MODELO DE DOMINIO ..... 27
Tabla 14: TAREA DIAGRAMAS DE CLASE ..... 28
Tabla 15: TAREA DIAGRAMAS DE SECUENCIA ..... 28
Tabla 16: TAREA IMPLEMENTACIÓN MOTOR GRÁFICO ..... 28
Tabla 17: TAREA IMPLEMENTACIÓN PING-PONG ..... 29
Tabla 18: TAREA IMPLEMENTACIÓN SPACE-DEFENSE ..... 29
Tabla 19: TAREA IMPLEMENTACIÓN PLATAFORMA DE JUEGO ..... 29
Tabla 20: TAREA PRUEBAS PING-PONG ..... 29
Tabla 21: TAREA PRUEBAS SPACE-DEFENSE ..... 30
Tabla 22: TAREA PRUEBAS PLATAFORMA DE JUEGO ..... 30
Tabla 23: TAREA REALIZACIÓN DE LA MEMORIA ..... 30
Tabla 24: TAREA REUNIÓN CON EL DIRECTOR ..... 31
Tabla 25: TAREA PREPARACIÓN DE LA DEFENSA ..... 31
Tabla 26: PLANIFICACIÓN TEMPORAL ..... 33
Tabla 27: RIESGO PÉRDIDA DE INFORMACIÓN ..... 38
Tabla 28: RIESGO PROBLEMAS CON EL HARDWARE ..... 38
Tabla 29: RIESGO SURGIMIENTO DE ERRORES EN LA IMPLEMENTACIÓN ..... 39
Tabla 30: RIESGO MALA ESTIMACIÓN TEMPORAL ..... 39

Tabla 31: RIESGO PÉRDIDA DE TIEMPO ..... 40
Tabla 32: GASTO EN PERSONAL ..... 41
Tabla 33: GASTO HARDWARE ..... 41
Tabla 34: GASTO SOFTWARE ..... 42
Tabla 35: GASTO TOTAL ..... 42
Tabla 36: CLIENTE-SERVIDOR VS P2P ..... 106
Tabla 37: PRUEBAS ..... 120
Tabla 38: PRUEBAS RENDIMIENTO SPACE-DEFENSE ..... 121
Tabla 39: PRUEBAS FUNCIONALIDAD NAVEGACION SPACE-DEFENSE ..... 122
Tabla 40: PRUEBAS FUNCIONALIDAD JUGABILIDAD SPACE-DEFENSE ..... 124
Tabla 41: PRUEBAS RENDIMIENGO PING-PONG ..... 125
Tabla 42: PRUEBAS FUNCIONALIDAD NAVEGACIÓN PING-PONG ..... 126
Tabla 43: PRUEBAS FUNCIONALIDAD JUGABILIDAD PING-PONG ..... 128
Tabla 44: PRUEBAS FUNCIONALIDAD PLATAFORMA DE JUEGO ..... 130
Tabla 45: CASO DE USO GESTIONAR SONIDO SPACE-DEFENSE ..... 134
Tabla 46: CASO DE USO VER RANKING SPACE-DEFENSE ..... 135
Tabla 47: CASO DE USO JUGAR SPACE-DEFENSE ..... 137
Tabla 48: CASO DE USO MOVERSE SPACE-DEFENSE ..... 138
Tabla 49: CASO DE USO PAUSAR SPACE-DEFENSE ..... 139
Tabla 50: CASO DE USO REANUDAR SPACE-DEFENSE ..... 140
Tabla 51: CASO DE USO DISPARAR SPACE-DEFENSE ..... 141
Tabla 52: CASO DE USO FINALIZAR PARTIDA SPACE-DEFENSE ..... 143
Tabla 53: CASO DE USO GESTIONAR SONIDO PING-PONG ..... 144
Tabla 54: CASO DE USO VER RANKING PING-PONG ..... 145
Tabla 55: CASO DE USO JUGAR ..... 147
Tabla 56: CASO DE USO MOVERSE ..... 148
Tabla 57: CASO DE USO PAUSAR ..... 149
Tabla 58: CASO DE USO REANUDAR ..... 150
Tabla 59: CASO DE USO FINALIZAR PARTIDA ..... 151
Tabla 60: CASO DE USO INICIAR SERVIDOR ..... 152
Tabla 61: CASO DE USO INICIAR SESIÓN ..... 153
Tabla 62: CASO DE USO REGISTRARSE ..... 154

Tabla 63: CASO DE USO RECUPERAR CONTRASEÑA ..... 155
Tabla 64: CASO DE USO CAMBIAR CONTRASEÑA ..... 156
Tabla 65: CASO DE USO CAMBIAR CORREO ..... 157
Tabla 66: CASO DE USO INICIAR JUEGO ..... 158

# 1. INTRODUCCIÓN

## 1.1. Descripción y situación del trabajo

Con este trabajo se pretende desarrollar una plataforma de juego exclusiva para ordenadores con características similares a las que existen actualmente en el mercado como puede ser Steam. Para que los usuarios de la plataforma puedan disfrutar desde un comienzo se crearán dos videojuegos.

El primer juego será el Space-Defense en el cual el jugador estará al control de una nave y tendrá que ir eliminando diferentes tipos de enemigos que aparecerán en oleadas de forma continuada hasta que el jugador agote las vidas disponibles. Este juego será similar al Space Invaders.

El segundo juego será multijugador online y se llamará Ping-Pong. La jugabilidad será muy similar al del juego Pong. Este juego consiste en un enfrentamiento entre dos jugadores en el que se van pasando una bola de un campo a otro hasta que uno consigue superar al contrincante y conseguir el punto. El jugador que llega a una determinada puntuación gana la partida.

Para facilitar el desarrollo de los juegos y con el objetivo de desarrollar más juegos para la plataforma en un futuro se va a crear un motor gráfico. Este motor aunará en una serie de módulos todas las funcionalidades que deberá tener todo juego a desarrollar en este proyecto.

Para desarrollar todo lo explicado se aplicarán los conocimientos obtenidos en las diferentes asignaturas de la carrera y se buscará formación de aquello que se desconozca. Como lenguaje principal de programación se usará Java, uno de los lenguajes más utilizados para el desarrollo de juegos en 2D.

## 1.2. Razones

Hoy en día los videojuegos están presentes en la sociedad como una de las principales formas de ocio. La mayoría de las personas disponen de un dispositivo desde el que jugar como puede ser un ordenador, consola, iPad o móvil y los utilizan de manera habitual para jugar.

Esto ha provocado que los beneficios obtenidos por la industria del videojuego hayan crecido enormemente en los últimos años, obteniendo incluso más ingresos que la industria musical y del cine juntas.

Analizando el mercado de los videojuegos se ha detectado un cambio en la forma de acceso a los videojuegos por parte de los jugadores. En los últimos años las ventas digitales de videojuegos han superado con creces a las ventas físicas en las tiendas de toda la vida. Esto ha provocado el surgimiento de las plataformas de juego como forma de venta de videojuegos. Estas plataformas

ofrecen grandes catálogos de videojuegos al alcance de un solo clic para sus usuarios.

Por estos motivos, se ha considerado oportuno formarse en el desarrollo de videojuegos ya que la industria del videojuego parece tener un futuro por lo menos prometedor.

## 1.3. Planteamiento del problema

El objetivo principal de este proyecto es adquirir los conocimientos necesarios para el desarrollo de videojuegos en dos dimensiones (2D). Para ello se pretende realizar una plataforma de juego en la que se incluirán dos juegos que se programarán desde cero sin la ayuda de herramientas externas como pueden ser los motores gráficos que hay en el mercado.

Para desarrollar un videojuego no solo hace falta experiencia en programación, sino que es necesario conocer otros aspectos involucrados en el diseño del juego como es la jugabilidad, el aspecto gráfico y la sonorización. Como este proyecto está centrado en la programación de la lógica del juego (jugabilidad), los juegos a desarrollar estarán basados en los juegos del Pong y del Space Invaders. De esta forma se pueden encontrar recursos gratuitos para crear el apartado gráfico y de sonido. Además, la mecánica de juego es conocida en ambos casos.

Dado que los juegos se van a desarrollar desde cero, se va a desarrollar un motor gráfico propio para facilitar el desarrollo y reducir los tiempos de implementación. Las funcionalidades de este motor gráfico serán únicamente las indispensables para desarrollar ambos juegos.

## 1.4. Justificación

Con este proyecto se pretende formarse en la producción de videojuegos y todo lo que le rodeo. Por ello, el proyecto abarcará diferentes aspectos como pueden ser los motores gráficos, juegos multijugador online y las plataformas de juego.

La creación de un motor gráfico facilitará la creación de videojuegos y reducirá en gran medida los tiempos de producción.

Con el motor gráfico se pretende crear una comunidad de desarrolladores de videojuegos para incorporar a la plataforma de juego. De esta forma, mediante la participación de más gente, los módulos y las funcionalidades ofrecidas en el motor gráfico se esperan ampliar en un futuro.

La plataforma de juego ofrece una forma fácil de acceso a videojuegos para todo aquel que quiera jugar a un juego sin tener que desplazarse a una tienda física a comprarlo. Esta plataforma ofrecerá desde un comienzo los juegos Ping-

Pong y Space-Defense para todos sus usuarios. En un futuro se deberá incorporar una tienda a través de la cual se descarguen los juegos que la comunidad de desarrolladores vaya creando.

Mediante la creación de esta plataforma de juego se pretende crear una comunidad de jugadores que disfruten de los juegos ofrecidos.

# 2. PLANTEAMIENTO INICIAL

## 2.1. Objetivos

A continuación, se muestran los objetivos del proyecto:

- Realización del juego multijugador online Ping-Pong utilizando un motor gráfico propio. Este juego será similar al Pong.
- Realización del juego Space-Defense utilizando un motor gráfico propio. Este juego será similar al Space Invaders.
- Realización de un motor gráfico de juegos 2D que cuente con los módulos necesarios para desarrollar los juegos propuestos.
- Realización de una plataforma de juego desde donde los usuarios puedan acceder a los juegos.
- Aprender a desarrollar distintos tipos de juegos en 2D.

## 2.2. Alcance

En este apartado se explicarán las diferentes fases con las que contará el proyecto. Cada una de estas fases contará a su vez con un listado de tareas a realizar, las cuales se mostrarán mediante un EDT (estructura de descomposición del trabajo).

Las fases del proyecto son las siguientes:

- **Gestión inicial:** fase inicial del proyecto en el que se decidirá que desarrollar.
- **Planificación:** durante esta fase se definirán las tareas con las que contará el proyecto y se realizará una planificación temporal.
- **Formación:** durante esta fase se adquirirán los conocimientos necesarios para desarrollar el proyecto.
- **Análisis y diseño:** durante esta fase se diseñará cada aplicación a realizar.
- **Implementación:** durante esta fase se desarrollará el codigo de cada una de las aplicaciones.
- **Pruebas:** durante esta fase se verificará que cada aplicación funciona correctamente y se realizarán las correcciones oportunas para que así sea.
- **Documentación**: fase final en la que se realizará la memoria y se preparará la defensa del proyecto.

### 2.2.1. EDT

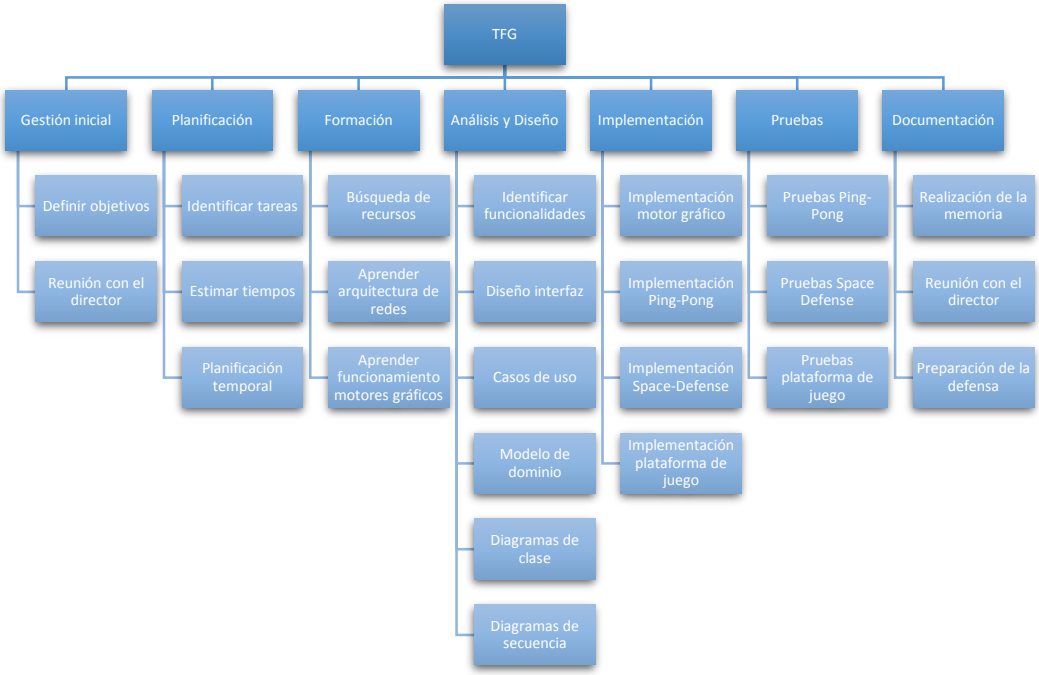



Ilustración 1: EDT

### 2.2.2. Tareas

A continuación, se explicará con más detalle cada una de las tareas. Para describir cada tarea se ha creado la siguiente tabla:

| TAREA | |
|---|---|
| **Descripción** | |
| **Duración** | |
| **Recursos** | |
| **Precedentes** | |
| **Salidas** | |

Tabla 1: DESCRIPCIÓN TAREA

- **Descripción:** breve descripción explicando en que consiste la tarea.
- **Duración:** tiempo estimado en horas para realizar la tarea.
- **Recursos:** listado con las herramientas y el material a utilizar en la tarea.
- **Precedentes:** listado con las tareas que deben ser completadas antes de comenzar la tarea descrita.
- **Salidas:** explicación de lo que se obtiene al finalizar la tarea.

#### 2.2.2.1. Gestión inicial

| Definir objetivos | |
|---|---|
| **Descripción** | Fijar los objetivos que se pretenden lograr con el proyecto. |
| **Duración** | 2 horas. |
| **Recursos** | Microsoft Word. |
| **Precedentes** | - |
| **Salidas** | Objetivos del proyecto. |

Tabla 2: TAREA DEFINIR OBJETIVOS

| Reunión con el director | |
|---|---|
| **Descripción** | Realización de una reunión con el director para exponerle el proyecto a realizar. |
| **Duración** | 1 hora. |
| **Recursos** | Documento con los objetivos y la descripción del proyecto que se pretende realizar. |
| **Precedentes** | Definir objetivos. |
| **Salidas** | - |

Tabla 3: TAREA REUNIÓN CON El DIRECTOR

#### 2.2.2.2. Planificación

| Identificar tareas | |
|---|---|
| **Descripción** | Identificar las fases del proyecto y las tareas de cada fase. |
| **Duración** | 3 horas. |
| **Recursos** | Microsoft Word. |
| **Precedentes** | Gestión inicial. |
| **Salidas** | Estructura de descomposición del trabajo (EDT). |

Tabla 4: TAREA IDENTIFICAR TAREAS

| Estimar tiempos | |
|---|---|
| **Descripción** | Estimar el tiempo que se tardará en realizar cada una de las tareas. |
| **Duración** | 1 hora. |
| **Recursos** | Microsoft Word. |
| **Precedentes** | Identificar tareas. |
| **Salidas** | Estimación del tiempo para cada una de las tareas. |

Tabla 5: TAREA ESTIMAR TIEMPOS

| Planificación temporal | |
|---|---|
| **Descripción** | Estimar la fecha de inicio y fin para cada una de las tareas. |
| **Duración** | 1 hora. |
| **Recursos** | Microsoft Word y GanttProject. |
| **Precedentes** | Estimar tiempos. |
| **Salidas** | Tabla con el listado tareas y su fecha de inicio y fin.<br>Diagrama de Gantt. |

Tabla 6: TAREA PLANIFICACIÓN TEMPORAL

#### 2.2.2.3. Formación

| Búsqueda de recursos | |
|---|---|
| **Descripción** | Búsqueda de material en internet. |
| **Duración** | 2 horas. |
| **Recursos** | Equipo con conexión a internet. |
| **Precedentes** | Planificación. |
| **Salidas** | Recursos electrónicos y libros. |

Tabla 7: TAREA BÚSQUEDA DE RECURSOS

| Aprender arquitectura de redes | |
|---|---|
| **Descripción** | Adquirir conocimientos para el desarrollo de juegos multijugador online. |
| **Duración** | 12 horas. |
| **Recursos** | Material encontrado en la búsqueda de recursos. |
| **Precedentes** | Búsqueda de recursos. |
| **Salidas** | Conocimiento adquirido. |

Tabla 8: TAREA APRENDER ARQUITECTURA DE REDES

| Aprender funcionamiento motores gráficos | |
|---|---|
| **Descripción** | Adquirir los conocimientos necesarios para crear un motor gráfico propio con las funcionalidades básicas. |
| **Duración** | 30 horas. |
| **Recursos** | Material encontrado en la búsqueda de recursos. |
| **Precedentes** | Búsqueda de recursos. |
| **Salidas** | Conocimiento adquirido. |

Tabla 9: TAREA APRENDER FUNCIONAMIENTO MOTORES GRÁFICOS

#### 2.2.2.4. Análisis y diseño

| Identificar funcionalidades | |
|---|---|
| **Descripción** | Identificar las funcionalidades que tendrán cada una de las aplicaciones que se van a desarrollar. |
| **Duración** | 12 horas. |
| **Recursos** | Microsoft Word. |
| **Precedentes** | Formación. |
| **Salidas** | Listado con las funcionalidades de cada aplicación. |

Tabla 10: TAREA IDENTIFICAR FUNCIONALIDADES

| Diseño interfaz | |
|---|---|
| **Descripción** | Realizar prototipos digitales para cada aplicación a desarrollar. |
| **Duración** | 4 horas. |
| **Recursos** | Adobe Photoshop. |
| **Precedentes** | Identificar funcionalidades. |
| **Salidas** | Prototipos digitales con las interfaces de usuario de cada aplicación. |

Tabla 11: TAREA DISEÑO INTERFAZ

| Casos de uso | |
|---|---|
| **Descripción** | Identificar los casos de uso de cada aplicación a desarrollar. |
| **Duración** | 5 horas. |
| **Recursos** | Visual Paradigm y Microsoft Word. |
| **Precedentes** | Diseño interfaz. |
| **Salidas** | Diagramas y tablas con los casos de uso de cada aplicación. |

Tabla 12: TAREA CASOS DE USO

| Modelo de dominio | |
|---|---|
| **Descripción** | Crear el modelo de dominio que englobe a todas las aplicaciones que se van a desarrollar. |
| **Duración** | 1 hora. |
| **Recursos** | Visual Paradigm. |
| **Precedentes** | Identificar funcionalidades. |
| **Salidas** | Modelo de dominio. |

Tabla 13: TAREA MODELO DE DOMINIO

| Diagramas de clase | |
|---|---|
| **Descripción** | Crear los diagramas de clase para cada aplicación que se va a desarrollar. |
| **Duración** | 8 horas. |
| **Recursos** | Visual Paradigm. |
| **Precedentes** | Casos de uso. |
| **Salidas** | Diagramas de clase de cada aplicación. |

Tabla 14: TAREA DIAGRAMAS DE CLASE

| Diagramas de secuencia | |
|---|---|
| **Descripción** | Crear los diagramas de secuencia para cada aplicación que se va a desarrollar. |
| **Duración** | 30 horas. |
| **Recursos** | Visual Paradigm. |
| **Precedentes** | Diagramas de clase. |
| **Salidas** | Diagramas de secuencia de cada aplicación. |

Tabla 15: TAREA DIAGRAMAS DE SECUENCIA

#### 2.2.2.5. Implementación

| Implementación motor gráfico | |
|---|---|
| **Descripción** | Implementar cada módulo del motor gráfico. |
| **Duración** | 12 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Análisis y diseño. |
| **Salidas** | Código del motor gráfico. |

Tabla 16: TAREA IMPLEMENTACIÓN MOTOR GRÁFICO

| Implementación Ping-Pong | |
|---|---|
| **Descripción** | Implementar el juego Ping-Pong. |
| **Duración** | 60 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Implementación motor gráfico. |
| **Salidas** | Código y ejecutable del juego Ping-Pong. |

Tabla 17: TAREA IMPLEMENTACIÓN PING-PONG

| Implementación Space-Defense | |
|---|---|
| **Descripción** | Implementar el juego Space-Defense. |
| **Duración** | 30 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Implementación motor gráfico. |
| **Salidas** | Código y ejecutable del juego Space-Defense. |

Tabla 18: TAREA IMPLEMENTACIÓN SPACE-DEFENSE

| Implementación plataforma de juego | |
|---|---|
| **Descripción** | Implementar la plataforma de juego. |
| **Duración** | 15 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Análisis y diseño. |
| **Salidas** | Código y ejecutable de la aplicación. |

Tabla 19: TAREA IMPLEMENTACIÓN PLATAFORMA DE JUEGO

#### 2.2.2.6. Pruebas

| Pruebas Ping-Pong | |
|---|---|
| **Descripción** | Comprobar que el juego Ping-Pong funciona correctamente y corregir los fallos en caso de que hubiera. |
| **Duración** | 8 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Implementación Ping-Ping. |
| **Salidas** | Resultados de las pruebas y código corregido. |

Tabla 20: TAREA PRUEBAS PING-PONG

| Pruebas Space-Defense | |
|---|---|
| **Descripción** | Comprobar que el juego Space-Defense funciona correctamente y corregir los fallos en caso de que hubiera. |
| **Duración** | 4 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Implementación Space-Defense. |
| **Salidas** | Resultados de las pruebas y código corregido. |

Tabla 21: TAREA PRUEBAS SPACE-DEFENSE

| Pruebas plataforma de juego | |
|---|---|
| **Descripción** | Comprobar que la aplicación funciona correctamente y corregir los fallos en caso de que hubiera. |
| **Duración** | 3 horas. |
| **Recursos** | Eclipse. |
| **Precedentes** | Implementación plataforma de juego. |
| **Salidas** | Resultados de las pruebas y código corregido. |

Tabla 22: TAREA PRUEBAS PLATAFORMA DE JUEGO

#### 2.2.2.7. Documentación

| Realización de la memoria | |
|---|---|
| **Descripción** | Realizar la memoria del proyecto. |
| **Duración** | 70 horas. |
| **Recursos** | Microsoft Word. |
| **Precedentes** | Gestión inicial. |
| **Salidas** | Memoria del proyecto. |

Tabla 23: TAREA REALIZACIÓN DE LA MEMORIA

| Reunión con el director | |
|---|---|
| **Descripción** | Realización de una reunión con el director para entregarle la memoria del proyecto. |
| **Duración** | 1 hora. |
| **Recursos** | Memoria del proyecto. |
| **Precedentes** | Realización de la memoria. |
| **Salidas** | - |

Tabla 24: TAREA REUNIÓN CON EL DIRECTOR

| Preparación de la defensa | |
|---|---|
| **Descripción** | Preparación y realización de una presentación para su exposición durante la defensa del proyecto. |
| **Duración** | 10 horas. |
| **Recursos** | Microsoft Power Point. |
| **Precedentes** | Realización de la memoria. |
| **Salidas** | Presentación. |

Tabla 25: TAREA PREPARACIÓN DE LA DEFENSA

## 2.3. Planificación temporal

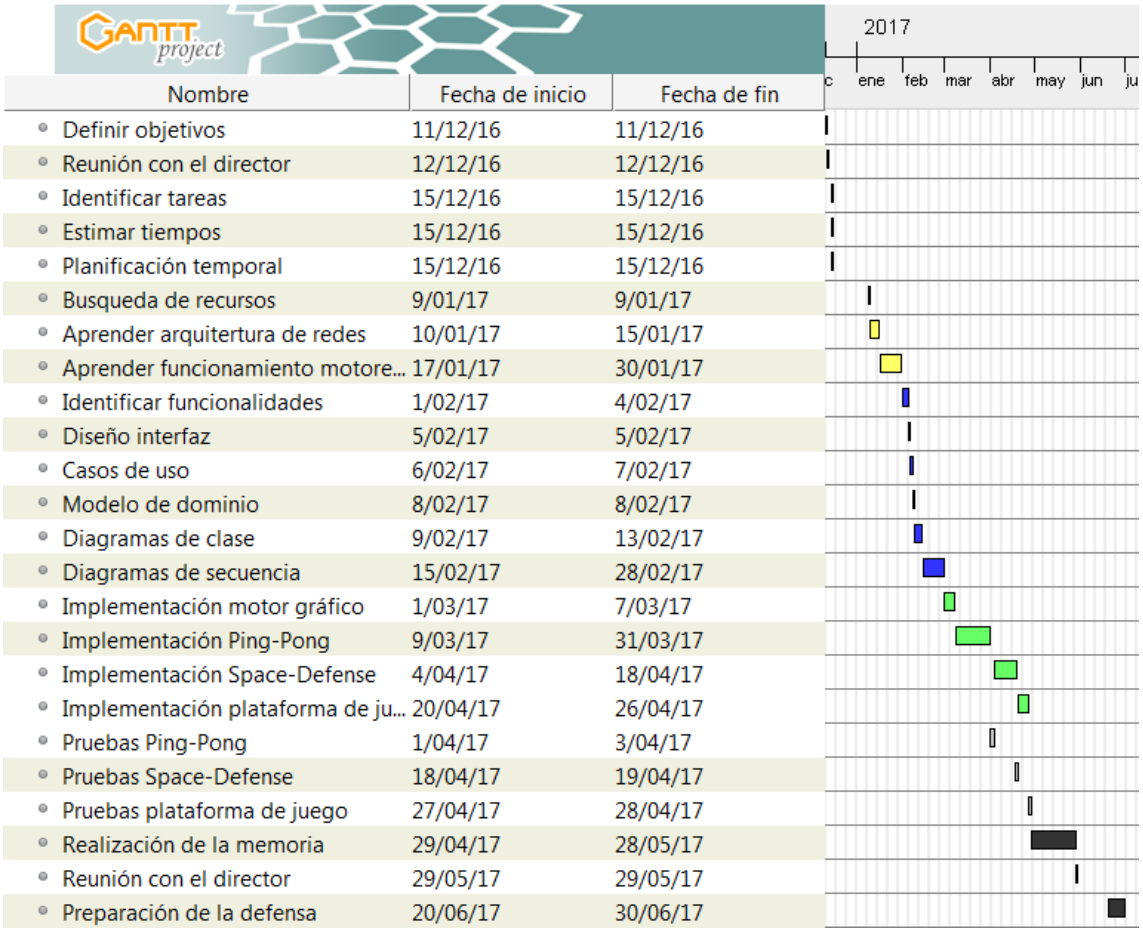
A continuación, se expondrá la tabla con la planificación temporal realizada junto con el diagrama de Gantt.

| Índice | Tarea | Fecha inicio | Fecha fin | Días | Horas |
|---|---|---|---|---|---|
| **1** | **Gestión inicial** | **11/12/2016** | **12/12/2016** | **2** | **3** |
| 1.1 | Definir objetivos | 11/12/2016 | 11/12/2016 | 1 | 2 |
| 1.2 | Reunión con el director | 12/12/2016 | 12/12/2016 | 1 | 1 |
| **2** | **Planificación** | **15/12/2016** | **15/12/2016** | **1** | **5** |
| 2.1 | Identificar tareas | 15/12/2016 | 15/12/2016 | 1 | 3 |
| 2.2 | Estimar tiempos | 15/12/2016 | 15/12/2016 | 1 | 1 |
| 2.3 | Planificación temporal | 15/12/2016 | 15/12/2016 | 1 | 1 |
| **3** | **Formación** | **09/12/2017** | **30/01/2017** | **22** | **44** |
| 3.1 | Búsqueda de recursos | 09/01/2017 | 09/01/2017 | 1 | 2 |
| 3.2 | Aprender arquitectura de redes | 10/01/2017 | 15/01/2017 | 6 | 12 |
| 3.3 | Aprender funcionamiento motores gráficos | 17/01/2017 | 30/01/2017 | 14 | 30 |
| **4** | **Análisis y diseño** | **01/02/2017** | **28/02/2017** | **28** | **60** |
| 4.1 | Identificar funcionalidades | 01/02/2017 | 04/02/2017 | 4 | 12 |
| 4.2 | Diseño interfaz | 05/02/2017 | 05/02/2017 | 1 | 4 |
| 4.3 | Casos de uso | 06/02/2017 | 07/02/2017 | 2 | 5 |
| 4.4 | Modelo de dominio | 08/02/2017 | 08/02/2017 | 1 | 1 |
| 4.5 | Diagramas de clase | 09/02/2017 | 13/02/2017 | 5 | 8 |
| 4.6 | Diagramas de secuencia | 15/02/2017 | 28/02/2017 | 14 | 30 |
| **5** | **Implementación** | **01/03/2017** | **22/04/2017** | **53** | **117** |
| 5.1 | Implementación motor gráfico | 01/03/2017 | 07/03/2017 | 7 | 12 |

| Índice | Tarea | Fecha inicio | Fecha fin | Días | Horas |
|---|---|---|---|---|---|
| 5.2 | Implementación Ping-Pong | 09/03/2017 | 31/03/2017 | 23 | 60 |
| 5.3 | Implementación Space-Defense | 04/04/2017 | 18/04/2017 | 15 | 30 |
| 5.4 | Implementación plataforma de juego | 20/04/2017 | 26/04/2017 | 7 | 15 |
| **6** | **Pruebas** | **01/04/2017** | **25/04/2017** | **49** | **15** |
| 6.1 | Pruebas Ping-Pong | 01/04/2017 | 03/04/2017 | 3 | 8 |
| 6.2 | Pruebas Space-Defense | 18/04/2017 | 19/04/2017 | 2 | 4 |
| 6.3 | Pruebas plataforma de juego | 27/04/2017 | 28/04/2017 | 2 | 3 |
| **7** | **Documentación** | **30/04/2017** | **30/06/2017** | **66** | **81** |
| 7.1 | Realización de la memoria | 29/04/2017 | 28/05/2017 | 30 | 70 |
| 7.2 | Reunión con el director | 29/05/2017 | 29/05/2017 | 1 | 1 |
| 7.3 | Preparación de la defensa | 20/06/2017 | 30/06/2017 | 11 | 10 |
| **TOTAL HORAS** | | | | | 325 |

Tabla 26: PLANIFICACIÓN TEMPORAL

## GANTT




| Nombre | Fecha de inicio | Fecha de fin |
|---|---|---|
| Definir objetivos | 11/12/16 | 11/12/16 |
| Reunión con el director | 12/12/16 | 12/12/16 |
| Identificar tareas | 15/12/16 | 15/12/16 |
| Estimar tiempos | 15/12/16 | 15/12/16 |
| Planificación temporal | 15/12/16 | 15/12/16 |
| Busqueda de recursos | 9/01/17 | 9/01/17 |
| Aprender arquitertura de redes | 10/01/17 | 15/01/17 |
| Aprender funcionamiento motore... | 17/01/17 | 30/01/17 |
| Identificar funcionalidades | 1/02/17 | 4/02/17 |
| Diseño interfaz | 5/02/17 | 5/02/17 |
| Casos de uso | 6/02/17 | 7/02/17 |
| Modelo de dominio | 8/02/17 | 8/02/17 |
| Diagramas de clase | 9/02/17 | 13/02/17 |
| Diagramas de secuencia | 15/02/17 | 28/02/17 |
| Implementación motor gráfico | 1/03/17 | 7/03/17 |
| Implementación Ping-Pong | 9/03/17 | 31/03/17 |
| Implementación Space-Defense | 4/04/17 | 18/04/17 |
| Implementación plataforma de ju... | 20/04/17 | 26/04/17 |
| Pruebas Ping-Pong | 1/04/17 | 3/04/17 |
| Pruebas Space-Defense | 18/04/17 | 19/04/17 |
| Pruebas plataforma de juego | 27/04/17 | 28/04/17 |
| Realización de la memoria | 29/04/17 | 28/05/17 |
| Reunión con el director | 29/05/17 | 29/05/17 |
| Preparación de la defensa | 20/06/17 | 30/06/17 |

Ilustración 2: GANTT

## 2.4. Herramientas

En este apartado se muestran las herramientas hardware y software que se van a utilizar, así como los diferentes lenguajes de programación.

### 2.4.1. Hardware

Las herramientas hardware que se van a utilizar son las siguientes:

- Ordenador de sobremesa con Windows 7.1.
- Ordenador portátil con Windows 8.2.

### 2.4.2. Software

A continuación, se muestran las herramientas software que se van a utilizar.

**Eclipse**

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Aunque soporta varios lenguajes se usa principalmente para el desarrollo de aplicaciones en lenguaje Java.

Las razones por las que se ha escogido este entorno de desarrollo son las siguientes:

- Se tiene experiencia previa trabajando en él.
- Posee una comunidad que crea constantes plugins que podrían ser de utilidad.

**MySQL Workbench**

MySQL Workbench es una herramienta visual para el desarrollo de bases de datos. También dispone de consola donde ejecutar código en SQL.

Se va a utilizar para crear las bases de datos y acceder localmente a ellas sin necesidad de subirlas a un servidor.

**Visual Paradigm**

Visual Paradigm es una herramienta para el diseño de aplicaciones mediante UML (lenguaje unificado de modelado).

Los motivos por los que se ha escogido este programa se exponen a continuación:

- Soporta ingeniería inversa para crear tanto diagramas de clase como secuencia a partir de código en Java.

- Soporta la creación de plantillas automáticas en Java a partir de diagramas de clases.
- Facilidad de uso.
- Proporciona una gran variedad de diferentes tipos de diagramas.

En el proyecto se va a utilizar para el desarrollo de los diagramas de clases, secuencia, casos de uso, máquinas de estado y modelo de dominio.

**Adove Photoshop**

Adove Photoshop es un potente software de creación y edición de fotografía.

Se va a utilizar para la creación y edición de las imágenes e iconos que necesiten las aplicaciones que se van a desarrollar.

**Microsoft Office**

Microsoft Office ofrece un paquete con múltiples herramientas para utilizar en el ámbito de la ofimática.

Para este proyecto se van a utilizar dos de los programas que se incluyen en el paquete:

- Microsoft Word para el desarrollo de toda la documentación incluida la memoria.
- Microsoft Power Point para el desarrollo de la presentación de la defensa.

**GanttProject**

GanttProject es un software gratuito utilizado para la gestión y organización de grandes proyectos.

En este proyecto se va a utilizar para la creación del diagrama de Gantt con la planificación temporal.

### 2.4.3. Lenguajes de programación

A continuación, se muestran los lenguajes de programación que se van a utilizar.

**Java**

Java es un lenguaje de programación orientado a objetos utilizado ampliamente para el desarrollo de todo tipo de aplicaciones.

Se ha escogido Java debido a los conocimientos y el manejo previo que se tiene en este lenguaje de programación.

**XML**

XML es un lenguaje basado en marcas usado para crear información estructurada. Un documento XML se compone de múltiples etiquetas dentro de las cuales se almacena la información u otras etiquetas formando una estructura en forma de árbol.

Este lenguaje se va a utilizar con dos fines diferentes durante el trabajo:

- Crear los ficheros de configuración para las aplicaciones.
- Crear los mensajes para el intercambio de información entre el cliente y servidor en la aplicación online.

## 2.5. Gestión de riesgos

En este apartado se van a describir y analizar cada uno de los riesgos que se han tenido en cuenta a la hora de desarrollar este proyecto.

Para cada riesgo se expondrá una pequeña descripción del mismo, las medidas de prevención, el plan de contingencia, la probabilidad de que el riesgo ocurra y el impacto que puede tener en el desarrollo del proyecto. Tanto la probabilidad como el impacto se van a medir en cinco rangos diferenciados: muy baja, baja, media, alta y muy alta.

| **RIESGO** | **PERDIDA DE INFORMACIÓN** |
|---|---|
| **DESCRIPCIÓN** | Se pierde la documentación o el código del disco duro del ordenador. |
| **MEDIDAS DE PREVENCIÓN** | Realizar copias de seguridad periódicas tanto del código como de la documentación y almacenarlas en Drive y en un dispositivo USB. |
| **PLAN DE CONTINGENCIA** | Recuperar la información perdida de Drive o el USB. |
| **PROBABILIDAD** | Baja. |
| **IMPACTO** | Muy alto. |

Tabla 27: RIESGO PÉRDIDA DE INFORMACIÓN

| **RIESGO** | **PROBLEMAS CON EL HARDWARE** |
|---|---|
| **DESCRIPCIÓN** | El ordenador utilizado para desarrollar el trabajo se estropea. |
| **MEDIDAS DE PREVENCIÓN** | Tener un ordenador portátil en disposición de ser usado. |
| **PLAN DE CONTINGENCIA** | Continuar el proyecto desde el portátil hasta el arreglo del ordenador principal. |
| **PROBABILIDAD** | Baja. |
| **IMPACTO** | Alto. |

Tabla 28: RIESGO PROBLEMAS CON EL HARDWARE

| RIESGO | SURGIMIENTO DE ERRORES EN LA IMPLEMENTACIÓN |
|---|---|
| **DESCRIPCIÓN** | A la hora de implementar una determinada parte del software se producen errores de los cuales se desconoce la posible solución. |
| **MEDIDAS DE PREVENCIÓN** | Disponer de libros y foros de solución software. |
| **PLAN DE CONTINGENCIA** | Acudir a los foros o libros para buscar la solución. |
| **PROBABILIDAD** | Alta. |
| **IMPACTO** | Varía dependiendo del error. |

Tabla 29: RIESGO SURGIMIENTO DE ERRORES EN LA IMPLEMENTACIÓN

| RIESGO | MALA ESTIMACIÓN TEMPORAL |
|---|---|
| **DESCRIPCIÓN** | Se sobrepasa el tiempo estimado para realizar una determinada tarea. |
| **MEDIDAS DE PREVENCIÓN** | Poner un margen de error a la hora de realizar la estimación temporal. |
| **PLAN DE CONTINGENCIA** | Invertir tiempo extra para finalizar la tarea cuanto antes sea posible para que no retrase el inicio de la siguiente tarea. |
| **PROBABILIDAD** | Alta. |
| **IMPACTO** | Medio. |

Tabla 30: RIESGO MALA ESTIMACIÓN TEMPORAL

| RIESGO | PÉRDIDA DE TIEMPO |
|---|---|
| **DESCRIPCIÓN** | Debido a motivos de enfermedad, familiares o de trabajo se produce un retraso respecto a la planificación temporal. |
| **MEDIDAS DE PREVENCIÓN** | - |
| **PLAN DE CONTINGENCIA** | Recuperar el tiempo perdido haciendo horas extra. |
| **PROBABILIDAD** | Media. |
| **IMPACTO** | Varía en función del tiempo que se pierda. |

Tabla 31: RIESGO PÉRDIDA DE TIEMPO

## 2.6. Evaluación económica

En este apartado se va a hacer un desglose de los gastos y beneficios del proyecto.

En primer lugar, se explicará la parte de gastos en la se diferencian cuatro áreas:

- Gasto en personal.
- Gasto hardware.
- Gasto software.
- Gasto en marketing.

Finalmente, aunque con lo desarrollado en este proyecto no se pretende obtener ningún beneficio, se explicará la forma de amortizar los gastos en un futuro.

### 2.6.1. Gasto en personal

En esta parte aparecerá la remuneración que se debe esperar como programador. Se ha estimado que un programador amateur como es el caso, cobra alrededor de 15€ la hora.

| | **Coste hora** | **Horas trabajadas** | **Total** |
|---|---|---|---|
| **Programador** | 15€ | 325 horas | 4875€ |
| | | **Gasto total** | 4875€ |

Tabla 32: GASTO EN PERSONAL

### 2.6.2. Gasto hardware

Para realizar el proyecto se ha utilizado un ordenador de sobremesa y un ordenador portátil de forma auxiliar durante algunas etapas del proyecto.

| | **Precio** | **Vida media** | **Tiempo de uso** | **Amortización** |
|---|---|---|---|---|
| **Ordenador sobremesa** | 1400€ | 6 años | 5 meses | 106€ |
| **Ordenador portátil** | 800€ | 4 años | 1 mes | 17€ |
| | | | **Gasto total** | 123€ |

Tabla 33: GASTO HARDWARE

### 2.6.3. Gasto software

En este apartado únicamente se contempla el gasto de Microsoft Office y Adove Photoshop ya que el resto del software utilizado es gratuito.

Hay que tener en cuenta que la licencia de Adove Photoshop es de suscripción mensual. Se ha estimado que se utilizará únicamente durante un mes ya que es tiempo suficiente para realizar el trabajo que requiere de su uso.

| | **Precio** | **Vida media** | **Tiempo de uso** | **Amortización/Coste** |
|---|---|---|---|---|
| **Microsoft Office** | 254€ | 4 años | 5 meses | 26€ |
| **Adobe Photoshop** | 24€/mes | - | 1 mes | 24€ |
| | | | **Gasto total** | 50€ |

Tabla 34: GASTO SOFTWARE

### 2.6.4. Gasto en marketing

Si se espera que la plataforma de juego sea conocida y tenga éxito será necesario lanzar una campaña de marketing. Para ello se publicitará la plataforma en diferentes páginas web relacionadas con el mundo de los videojuegos. Esta campaña estará activa durante el primer año después del lanzamiento de la plataforma y tendrá un coste total de 1200€.

### 2.6.5. Gasto total

A continuación, se expone el gasto total esperado de este proyecto.

| **Área** | **Gasto** |
|---|---|
| Personal | 4875€ |
| Hardware | 123€ |
| Software | 50€ |
| Marketing | 1200€ |
| **Total** | 6248€ |

Tabla 35: GASTO TOTAL

El gasto total del proyecto asciende a 6248€.

### 2.6.6. Beneficios

Con todo lo desarrollado en este proyecto no se pretende obtener ningún beneficio. Sin embargo, si la plataforma tiene éxito y se continúan desarrollando videojuegos, se pretende que el 20% de los ingresos obtenidos por la venta de cada juego se obtenga como beneficio. El 80% de los ingresos obtenidos por la venta de cada juego irán para el desarrollador. El modelo de negocio de los juegos deberá ser de pago único.

Dado que se espera que los juegos estén desarrollados mediante el motor gráfico desarrollado y sean por lo tanto bastante simples, se ha estimado que el precio medio del juego sea de 2€. Por lo tanto, por cada juego vendido se obtendría 0,4€ de beneficio. Para amortizar el coste del proyecto haría falta la venta de 15.620 juegos.

# 3. ANTECEDENTES

En este apartado se pretende hacer un análisis de los videojuegos a lo largo de la historia, los distintos modelos de negocio, las plataformas de juego y los motores gráficos de videojuegos.

## 3.1. Concepto de videojuego e historia

En primer lugar, es necesario entender el concepto de videojuego. Un videojuego se puede definir como un juego electrónico que se visualiza a través de una pantalla y el jugador interactúa por medio de un controlador.

Desde la aparición de los videojuegos por la década de los 40 del siglo pasado, estos se han asentado como una de las principales formas de entretenimiento en la sociedad.

Conforme avanza la tecnología y los años han ido surgiendo nuevos tipos de videojuego y se han ido creando nuevos estudios de desarrollo para poder satisfacer la creciente demanda. Esto ha hecho de la industria del videojuego una de las más rentables, superando en ingresos a la industria de la música y el cine.

Los juegos que se van a desarrollar en este proyecto están basados en dos de los videojuegos más exitosos de la historia y que más impacto han tenido en la industria del videojuego. Estos juegos son el Pong y el Space Invaders.

El Pong es un videojuego publicado por Atari en 1972 para la primera generación de videoconsolas. El juego simula una tabla de tenis de mesa en la cual un jugador controla una paleta que usa para golpear la bola y se enfrenta a otro jugador o a la computadora controlada por inteligencia artificial. Debido a su éxito ha sido versionado múltiples veces incorporando nuevas funcionalidades o extrapolándolo a otros deportes como fútbol o tenis.

El Space Invaders es un videojuego publicado por Taito Corporation en 1978. En este juego el jugador controla una nave con la que puede disparar para ir matando a los diferentes marcianos que aparecen en pantalla. Es uno de los juegos más exitosos de la historia tanto por los beneficios económicos obtenidos como por la trascendencia que tuvo, ya que fue el precursor de los videojuegos modernos que conocemos hoy en día y estableció un nuevo género conocido como “shoot 'em up”. Debido a su éxito se han creado numerosos juegos siguiente la misma temática.

## 3.2. Modelos de negocio

Actualmente existen varios modelos de negocio (2) con los que se puede obtener beneficios a través de un videojuego.

- **De pago:** en este modelo se vende el juego a un determinado precio y se tiene acceso a todo su contenido desde un inicio.
- **De pago con compras de contenido:** en este modelo se vende el juego a un determinado precio y se tiene acceso a un contenido básico de inicio. Este contenido se irá ampliando en un futuro y será accesible previo pago.
- **Free To Play (F2P):** los juegos de este tipo son totalmente gratuito para los usuarios y se tiene acceso a la mayoría de su contenido. En la mayoría de los casos suelen disponer de un sistema de pago para desbloquear ciertas características del juego de forma más rápida a como se conseguirían jugando o acceder a contenido exclusivo.
- **Gratis con anuncios:** los juegos que se venden con este modelo son de acceso gratuito con todo su contenido gratis. Los beneficios se obtienen con los anuncios que se muestran dentro del juego.
- **Suscripción:** los juegos de este tipo requieren de pagos cada cierto periodo de tiempo para poder continuar jugando. Generalmente los pagos suelen ser de forma mensual.

Basándose en el mercado de juegos para dispositivos móviles, la mayoría de los juegos que se ponen a la venta siguen uno de los tres primeros modelos de negocio.


Ilustración 3: COMPARATIVA MODELOS DE NEGOCIO (2)

Como se ve en el gráfico el modelo más común es el F2P. Esto es debido a que tiene ventajas tanto para los jugadores como para el desarrollador. Los jugadores pueden jugar de forma gratuita y a aquellos que quieran pueden gastar el dinero que deseen. En lo relativo al desarrollador, los beneficios que se puede llegar a ganar en este modelo de negocio pueden llegar a ser similares e incluso

superar los beneficios de los otros tipos de modelos aun siendo el juego totalmente gratuito. Si bien esto último es cierto, el modelo de negocio más seguro para obtener beneficios es el modelo de pago.

## 3.3. Motores gráficos de videojuegos

Para entender mejor lo que es un motor gráfico se va a realizar una pequeña descripción con sus características principales y los beneficios que tiene utilizar uno para el desarrollo de videojuegos.

Un motor gráfico está formado por una serie de módulos que facilitan el desarrollo de un videojuego. Estos módulos incluyen las funcionalidades que deberían estar presentes en la mayoría de los videojuegos. Algunos de estos módulos son:

- Sonido
- Sistema colisiones
- Controladores de entrada
- Animación
- Inteligencia artificial
- Gráficos 2D y 3D

En algunos casos vienen incluidos dentro de un software con una interfaz de usuario lo que permite el diseño del juego de manera visual.

Actualmente disponer de un motor gráfico para desarrollar un videojuego es casi imprescindible por los múltiples beneficios que aporta. Algunos de estos son:

- Se empieza el desarrollo siempre desde una base proporcionada por los módulos por lo que no es necesario programar todo.
- Reducción del tiempo de desarrollo.
- En los que tienen interfaz permite crear los escenarios de forma visual sin escribir ni una línea de código.

La mayoría de los motores gráficos que existen en el mercado están programados en C++ ya que es el lenguaje que mejor rendimiento ofrece. Sin embargo, también existen algunos en Java y se utilizan principalmente para el desarrollo de juegos sencillos y en 2D.

A continuación, se van a analizar los motores gráficos que se han tenido en cuenta para realizar el motor gráfico de este proyecto.

### 3.3.1. Unity

Unity es uno de los motores gráficos más potentes y utilizados para el desarrollo de juegos tanto en 2D como 3D para múltiples plataformas. Dispone de innumerables características que facilitan el desarrollo de los juegos además

de una interfaz desde la que se puede crear los escenarios de forma visual. Además, permite programar en diferentes lenguajes como C#, Boo o UnityScript que es similar a JavaScript.

### 3.3.2. GameMaker

GameMaker es un motor gráfico dirigido a desarrolladores de videojuegos que no tengan conocimientos de programación. A través de la interfaz de usuario de la que dispone permite la creación de videojuegos sin escribir una solo línea de código. Esto limita en gran medida el diseño del juego a las características proporcionadas por el motor. Para crear las partes más complejas permite la escritura de código utilizando el lenguaje de programación de scripts Game Maker Language.

### 3.3.3. LibGDX

LibGDX es un framework que dispone de una librería de clases agrupadas en módulos que permiten el desarrollo de videojuegos tanto en 2D como 3D para múltiples plataformas. No dispone de interfaz de usuario por lo que se requiere de un entorno de desarrollo para programar los juegos haciendo uso de las clases proporcionadas en la librería. Esto probablemente sea lo más parecido al motor gráfico que se quiere desarrollar en este proyecto.

## 3.4. Plataformas de juego

En los últimos años se ha producido un cambio en el mercado de los videojuegos. Los jugadores han pasado de comprar los juegos en las tiendas locales a hacerlo de forma digital.

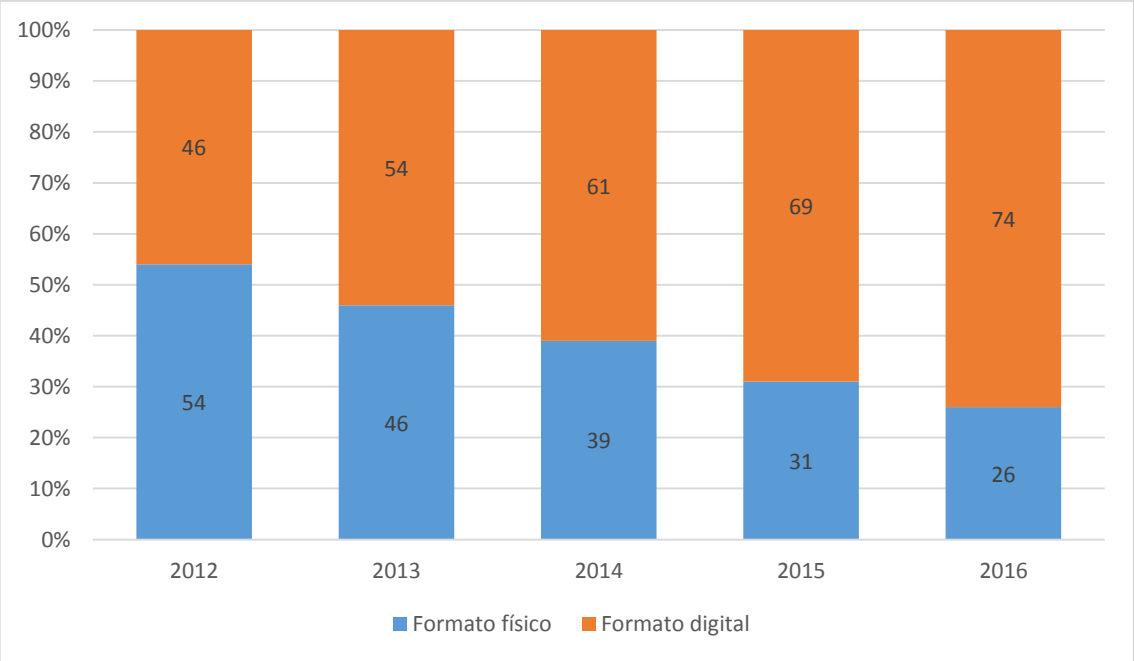



Ilustración 4: COMPARATIVA VENTAS DIGITALES Y FÍSICAS (3)

Una de las principales razones de este cambio es la diferencia de precio entre ambos formatos. Un juego comprado en formato digital cuesta bastante menos que un juego en físico. Esto es debido a que el juego digital no tiene costes asociados además de los de producción, mientras que el juego físico tiene los costes de almacenamiento, embalaje y transporte hasta la tienda donde se vaya a vender.

Otro motivo es la comodidad. Los juegos en formato digital están siempre accesibles y de forma casi instantánea siempre que se tenga buena velocidad de descarga. Por otro lado, para comprar un juego en formato físico hace falta desplazarse a la tienda y que el juego esté en stock.

Todo esto ha propiciado el crecimiento de las plataformas de juego. A continuación, se mostrarán aquellas plataformas que han sido analizadas para crear la de este proyecto.

### 3.4.1. Steam

Steam es la plataforma de juego para ordenadores de la empresa Valve Corporation lanzada en 2003. Es la plataforma de juego más conocida y grande del mundo con aproximadamente 10 millones de usuarios concurrentes.

Dispone de una red social con sistema de mensajería instantánea además de otras muchas funcionalidades.

Su modelo de negocio es bastante simple. La plataforma permite poner a la venta dentro de su tienda cualquier juego de cualquier desarrollador siempre que pase un control de calidad. Por cada copia digital vendida la plataforma se queda con un porcentaje del precio. A cambio, el juego adquiere la publicidad y el certificado de calidad por parte de la plataforma.

### 3.4.2. Battle.net

Battle.net es la plataforma de juego para ordenador de la empresa Blizzard Entertainment lanzada en 1997. Aparte del sistema de compra y acceso a los juegos disponibles, dispone de un apartado social donde los usuarios pueden comunicarse entre sí mediante un sistema de mensajería instantánea.

Los juegos disponibles son propiedad exclusiva de la empresa propietaria, no pudiendo empresas desarrolladoras externas publicar sus juegos en la plataforma. Estos juegos siguen distintos tipos de modelo de negocio como F2P, pago único o suscripción. También dispone de una moneda virtual que se puede comprar con dinero real y usar para acceder a distintos tipos de contenido dentro de los juegos.

# 4. CAPTURA DE REQUISITOS

En este apartado se explicarán las funcionalidades que deben tener cada una de las aplicaciones que se van desarrollar. Para ello se hará uso del diagrama de casos de uso.

## 4.1. Jerarquía de actores

Para el conjunto de todas las aplicaciones a desarrollar se han identificado tres tipos de actores diferentes e independientes.

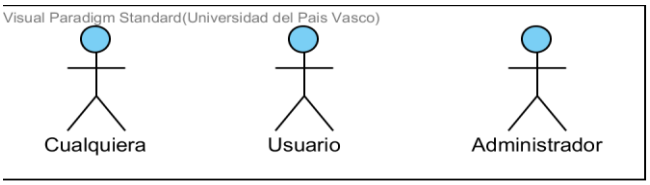



Ilustración 5: JERARQUÍA DE ACTORES

- **Cualquiera:** persona que utiliza la plataforma de juego sin haber iniciado sesión.
- **Usuario:** persona que ha iniciado sesión en la plataforma de juego y tiene acceso a los juegos.
- **Administrador:** persona que se encargará de gestionar el servidor del juego Ping-Pong.

## 4.2. Casos de uso Space-Defense

A continuación, se muestra el diagrama de casos de uso del juego Space-Defense y una breve descripción de cada caso de uso.

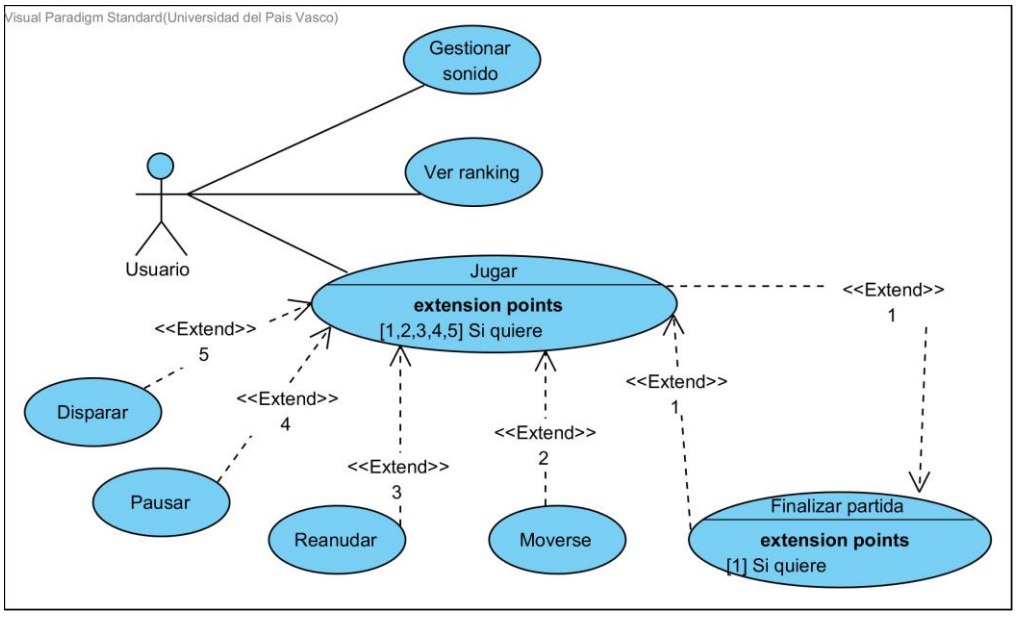



Ilustración 6: DIAGRAMA CASOS DE USO SPACE-DEFENSE

- **Gestionar sonido:** permite al usuario activar o desactivar el sonido.
- **Ver ranking:** permite al usuario ver el ranking con las mejores puntuaciones obtenidas entre todos los usuarios y su mejor puntuación.
- **Jugar:** permite al usuario jugar una partida.
- **Pausar:** permite al usuario pausar la partida.
- **Reanudar:** permite al usuario reanudar una partida en pausa.
- **Moverse:** permite al usuario mover la nave horizontalmente a través del tablero.
- **Disparar:** permite al usuario disparar balas durante la partida.
- **Finalizar partida:** permite al usuario finalizar la partida antes de que acabe.

### 4.3. Casos de uso Ping-Pong

A continuación, se muestra el diagrama de casos de uso del juego Ping-Pong y una breve descripción de cada caso de uso. Se incluyen los casos de uso de la aplicación cliente y servidor.

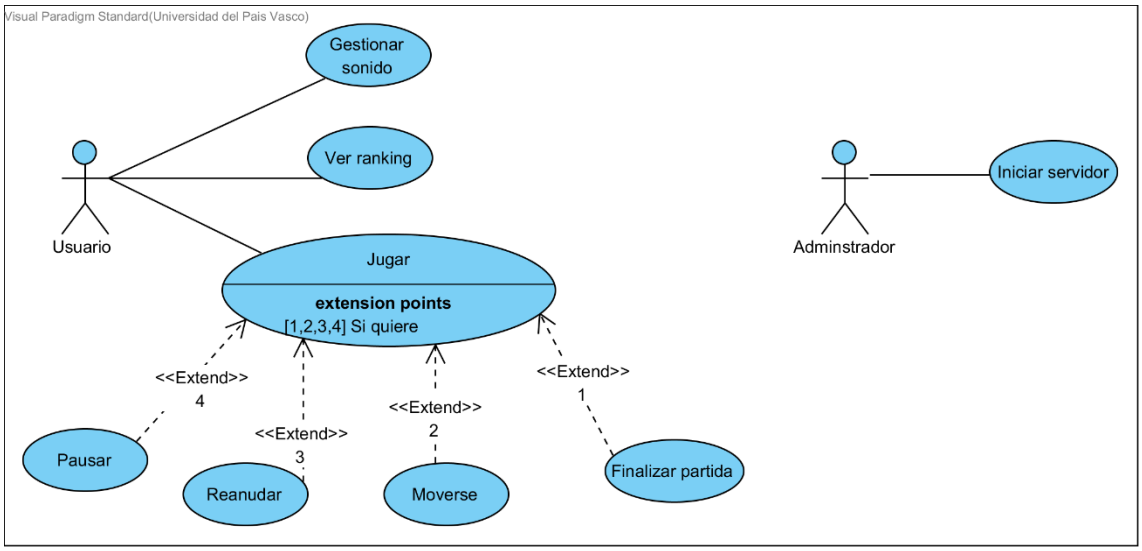



Ilustración 7: DIAGRAMA CASOS DE USO PING-PONG

- **Gestionar sonido:** permite al usuario activar o desactivar el sonido.
- **Ver ranking:** permite al usuario ver el ranking con el Elo (nivel de habilidad) de los mejores jugadores y el suyo propio.
- **Jugar:** permite al usuario jugar una partida.
- **Pausar:** permite al usuario pausar la partida.
- **Reanudar:** permite al usuario reanudar una partida en pausa.
- **Moverse:** permite al usuario moverse con su paleta a través del tablero de juego verticalmente.
- **Finalizar partida:** permite al usuario finalizar la partida antes de que acabe. En este caso se le dará por perdida.
- **Iniciar servidor:** permite al administrador iniciar el servidor para que los usuarios puedan jugar.

## 4.4. Casos de uso plataforma de juego

A continuación, se muestra el diagrama de casos de uso de la plataforma de juego y una breve descripción de cada caso de uso.

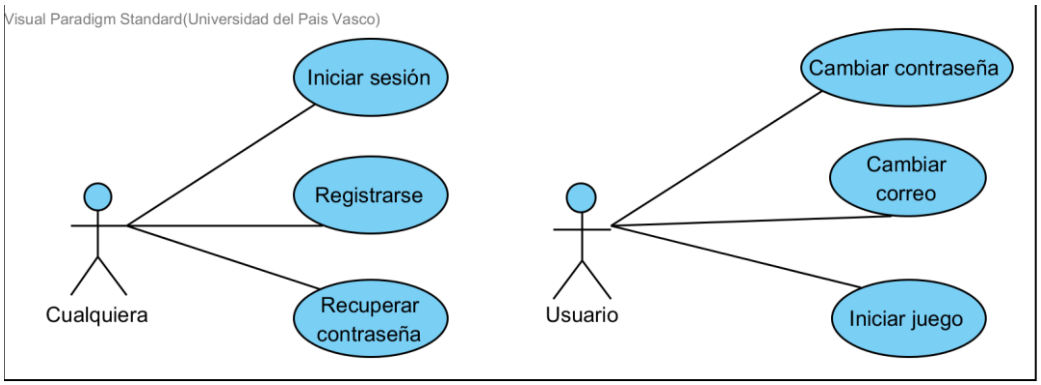



Ilustración 8: DIAGRAMA CASOS DE USO PLATAFORMA DE JUEGO

- **Registrarse:** permite a cualquier persona crear una cuenta para poder acceder a la plataforma de juego. Se deberá proporcionar un correo electrónico, nombre usuario y contraseña.
- **Recuperar contraseña:** permite recuperar la contraseña mediante el envío de un correo electrónico.
- **Iniciar sesión:** permite al usuario acceder a la plataforma de juego mediante el nombre usuario y contraseña.
- **Cambiar contraseña:** permite al usuario cambiar la contraseña de su cuenta.
- **Cambiar correo:** permite al usuario cambiar el correo electrónico de su cuenta.
- **Iniciar juego:** permite al usuario jugar a uno de los juegos que están disponibles en el catálogo.

## 4.5. Modelo de dominio

En este apartado se muestra el modelo de dominio diseñado.

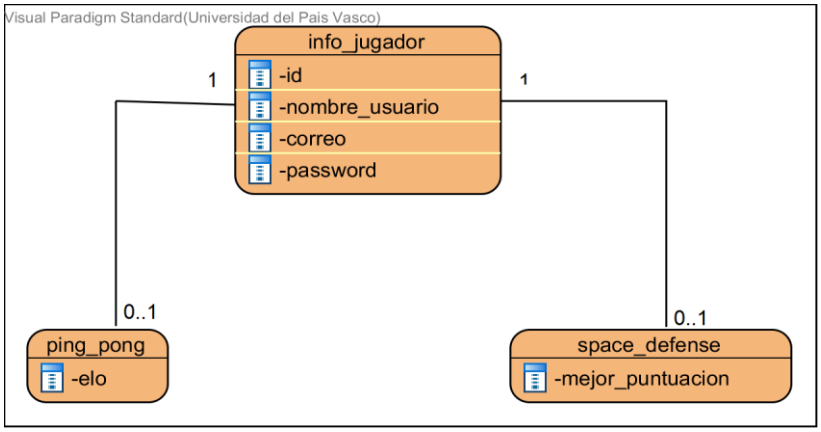



Ilustración 9: MODELO DE DOMINIO

En la entidad info_jugador se almacenará la información de los usuarios que se registren en la plataforma de juego.

En las entidades ping_pong y space_defense se almacenará el Elo y la mejor puntuación de cada usuario respectivamente. Esta información se podría haber incorporado directamente en la entidad info_jugador, pero se ha decidido utilizar entidades diferentes con el objetivo de tener entidades independientes para cada uno de los juegos.

# 5. ANÁLISIS Y DISEÑO

## 5.1. Motor gráfico

En este apartado se explicarán las clases que componen el motor gráfico desarrollado.

Para facilitar la comprensión del diagrama de clases y su funcionamiento se han agrupado las clases en múltiples módulos.

### 5.1.1. Diagrama de clases

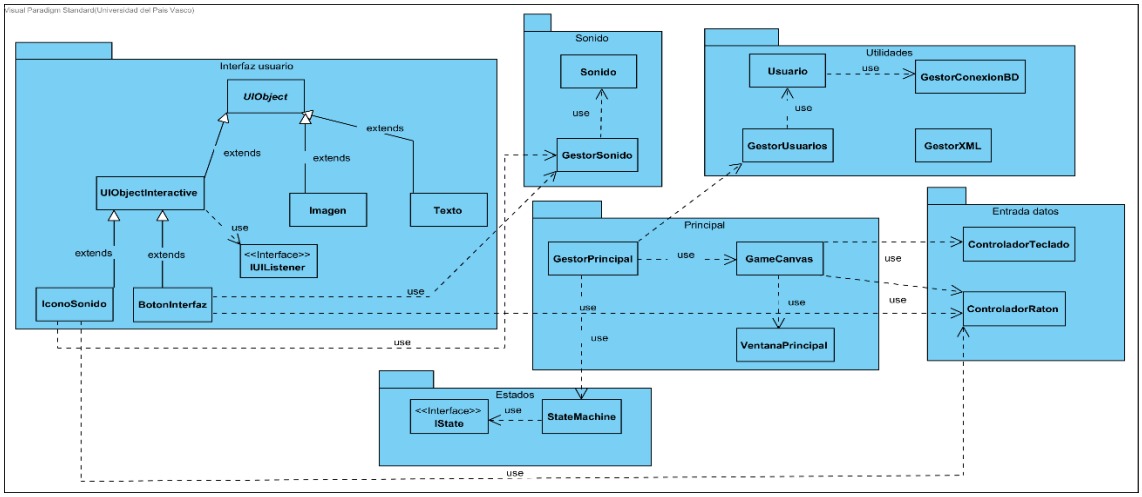



Ilustración 10: DIAGRAMA DE CLASES MOTOR GRÁFICO

### 5.1.2. Clases

#### 5.1.2.1. Módulo de entrada datos

En este módulo se encuentras las clases encargadas de gestionar la entrada de datos por parte del usuario a través de los dispositivos de entrada.

### ControladorRaton

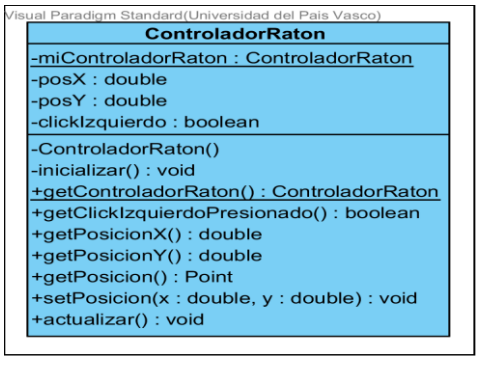



Ilustración 11: CONTROLADOR RATÓN

Clase diseñada con el patrón Singleton encargada de gestionar información relativa a la interacción del usuario mediante el ratón. Ofrece métodos para conocer la posición del puntero del ratón y si se ha hecho clic recientemente.

### ControladorTeclado

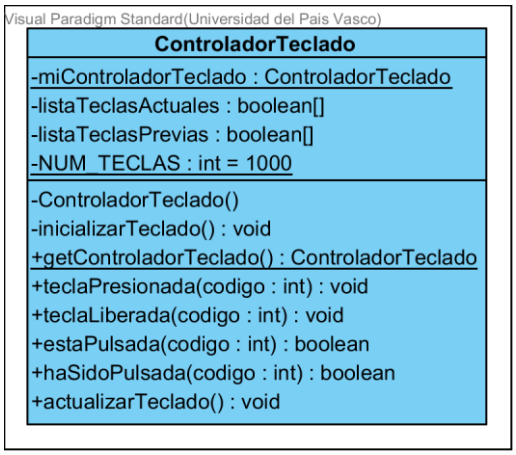



Ilustración 12: CONTROLADOR TECLADO

Esta clase se encarga de gestionar la entrada de datos a través del teclado. Se trata de una clase diseñada con el patrón Singleton para poder utilizarla desde cualquier parte de la aplicación para conocer si una determinada tecla ha sido pulsada o está siendo pulsada.

### 5.1.2.2. Módulo de estados

En este módulo se encuentra todas las clases relacionadas con la gestión de los diferentes estados en los que se puede encontrar la aplicación. Por explicarlo de algún modo, cada estado representa una determinada pantalla o vista de la aplicación.

### IState

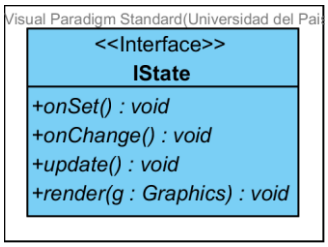



Ilustración 13: ISTATE

Interfaz que deberán implementar todos los estados. Contiene los métodos para actualizar y renderizar el estado, así como avisar cuando se inicia y cuando finaliza.

### StateMachine

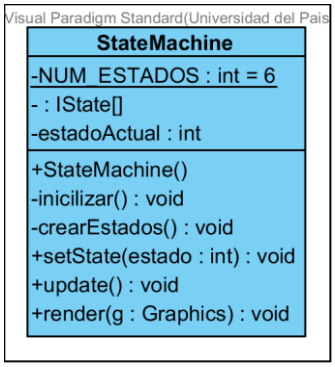



Ilustración 14: STATE MACHINE

Esta clase es la encargada de gestionar los diferentes estados que puede tener la aplicación. Entre sus principales funciones se encuentran actualizar y renderizar el estado actual y cambiar entre los diferentes estados.

#### 5.1.2.3. Módulo principal

Este módulo contiene las clases que se encargan de poner en marcha la aplicación.

### GestorPrincipal

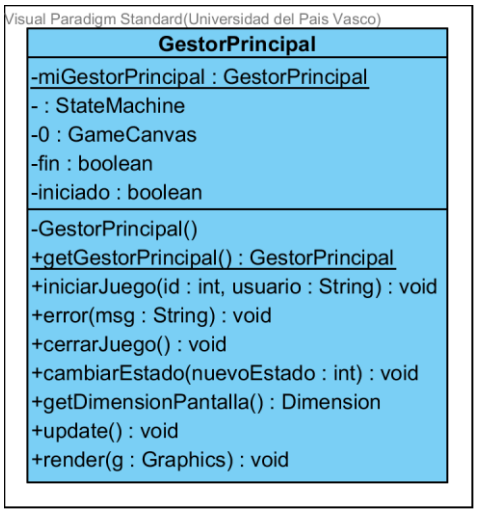



Ilustración 15: GESTOR PRINCIPAL

Clase diseñada mediante el patrón Singleton que se encarga de iniciar la aplicación y gestionar el funcionamiento de la misma.

### VentanaPrincipal

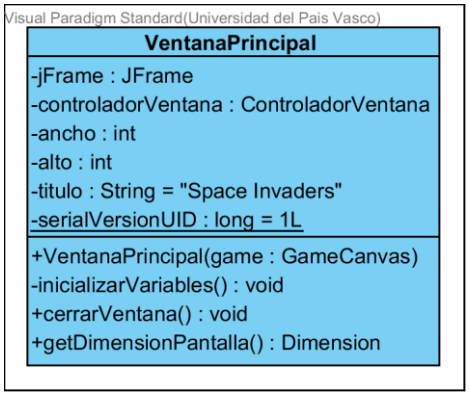



Ilustración 16: VENTANA PRINCIPAL

Esta clase se encarga de crear el JFrame de la aplicación y añadirle una instancia de la clase GameCanvas para dibujar los gráficos en pantalla.

### GameCanvas

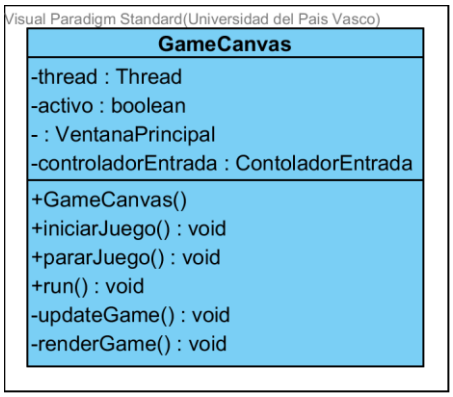



Ilustración 17: GAME CANVAS

Esta clase contiene el algoritmo que se encargará de actualizar y renderizar el juego en un Canvas a través de la clase VentanaPrincipal.

### 5.1.2.4. Módulo de interfaz de usuario

Dentro de este módulo se encuentran las clases diseñadas para crear la interfaz de usuario.

### UIObject

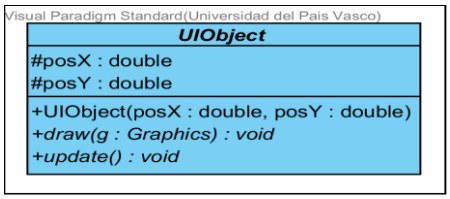



Ilustración 18: UI OBJECT

Clase abstracta que proporciona los atributos y métodos básicos para cualquier elemento de la interfaz.

### UIObjectInteractive

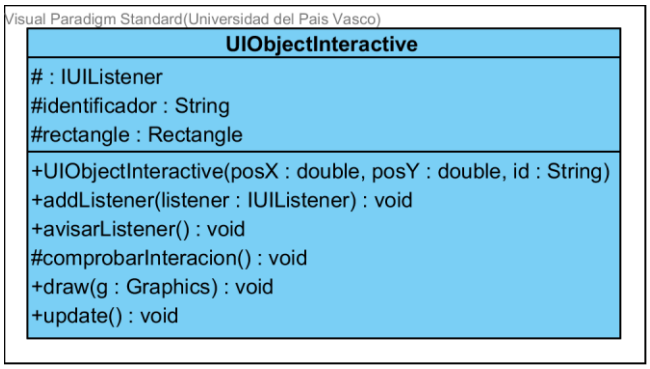



Ilustración 19: UI OBJECT INTERACTIVE

Clase abstracta que extiende de UIObject y de la que heredarán aquellos elementos de la interfaz con los que el usuario puede interactuar.

### IUIListener

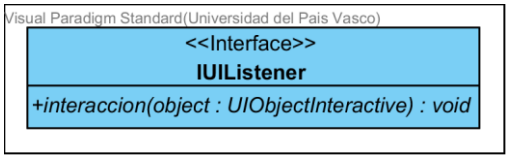



Ilustración 20: IUI LISTENER

Interfaz que deberán implementar aquellas clases que hagan uso de la clase UIObjectInteractive. De esta forma se las proporciona un método para avisar de la interacción del usuario.

### Texto

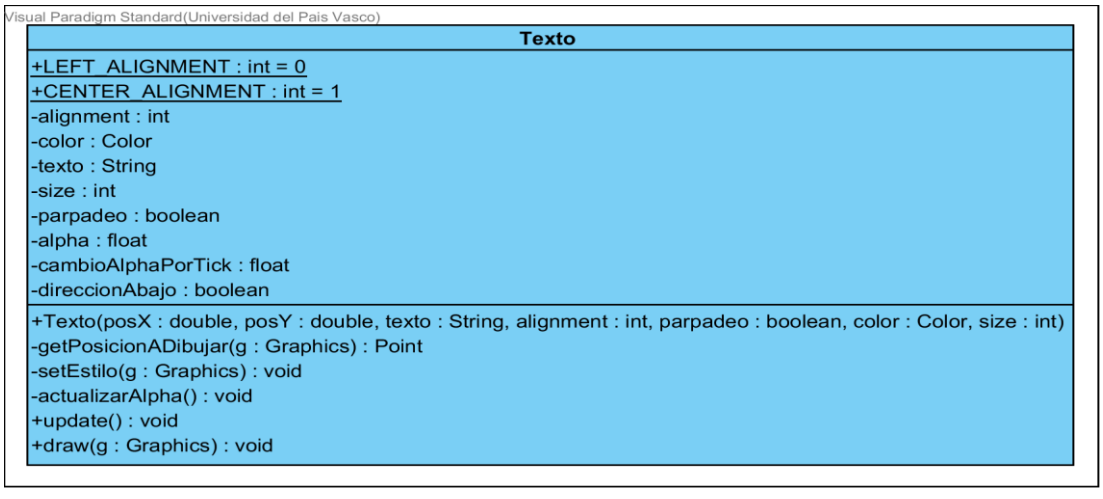



Ilustración 21: TEXTO

Esta clase extiende de UIObject y se encarga de renderizar texto personalizado en pantalla.

### Imagen

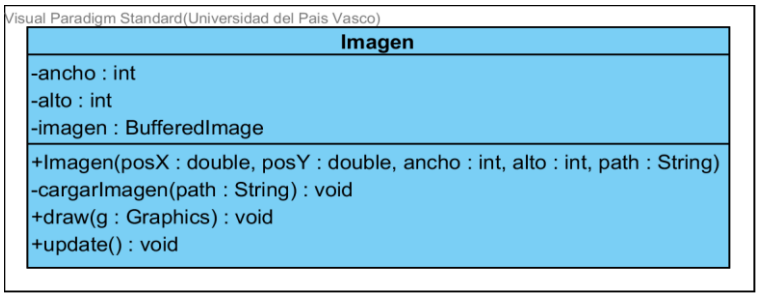



Ilustración 22: IMAGEN

Esta clase extiende de UIObject y se encarga de renderizar una imagen en pantalla.

### BotonInterfaz

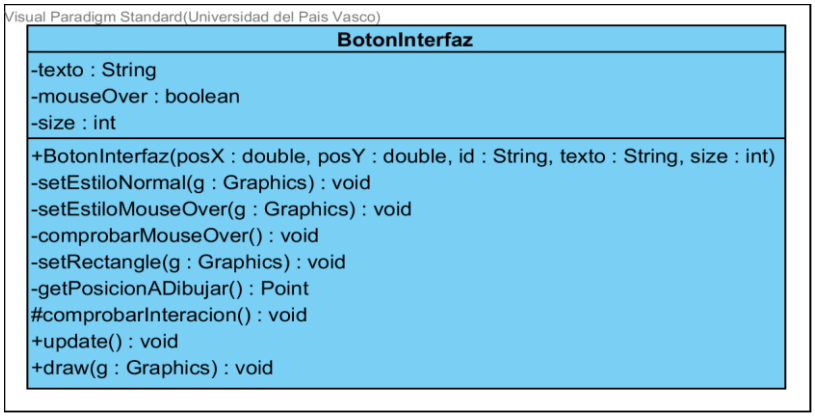



Ilustración 23: BOTÓN INTERFAZ

Esta clase extiende de UIObjectInteractive y representa un botón con el que el usuario puede interactuar para llevar a cabo una acción.

### IconoSonido

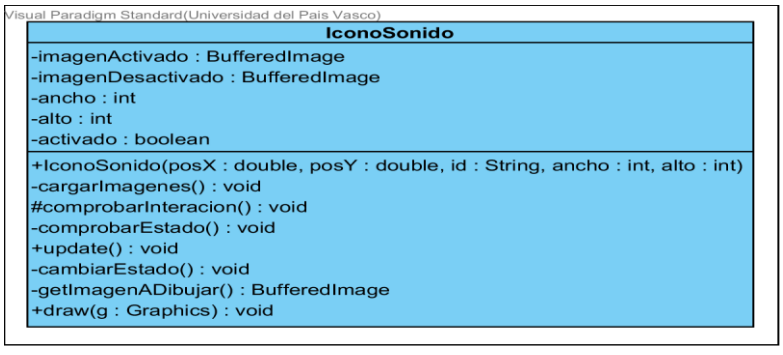



Ilustración 24: ICONO SONIDO

Clase que extiende de UIObjectInteractive y representa el icono de sonido con el que el usuario puede interactuar para activar o desactivar el sonido dentro de la aplicación.

### 5.1.2.5. Módulo de sonido

Las clases de este módulo se encargarán de dar soporte sonoro a la aplicación.

#### Sonido

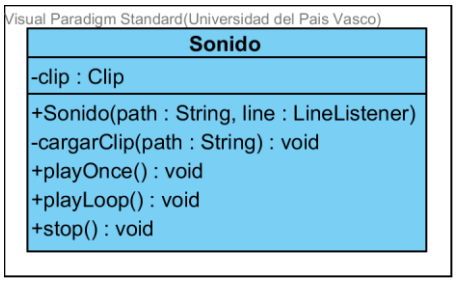



Ilustración 25: SONIDO

Esta clase se encargará de reproducir un determinado fichero de sonido en formato ".wav" en modo Loop (de forma indefinida) o una única vez.

#### GestorSonido

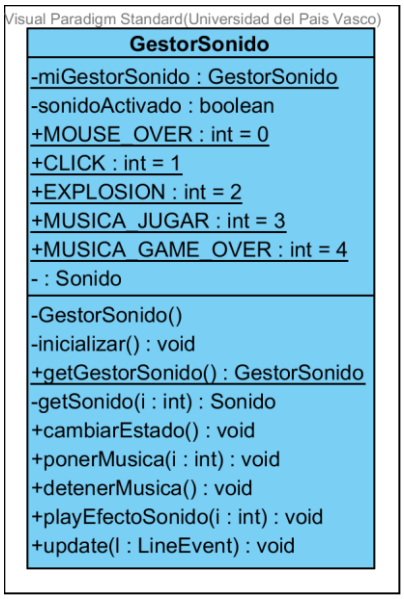



Ilustración 26: GESTOR SONIDO

Clase Singleton que se encarga de gestionar y reproducir tanto música como efectos de sonido haciendo uso de la clase Sonido descrita previamente.

### 5.1.2.6. Módulo de utilidades

Las clases de este módulo servirán de soporte para la aplicación.

#### GestorXML

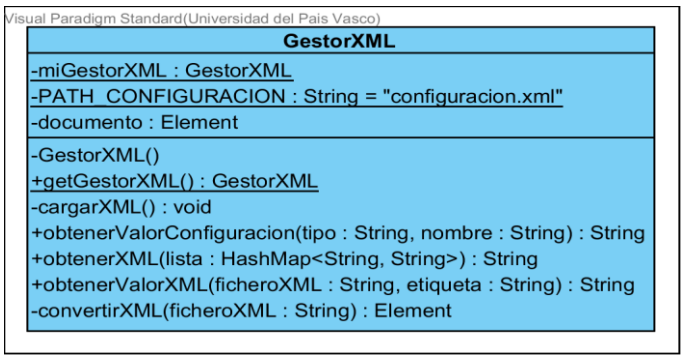



Ilustración 27: GESTOR XML

Clase que sigue el patrón Singleton y se encarga de gestionar ficheros en formato XML.

#### GestorConexionBD

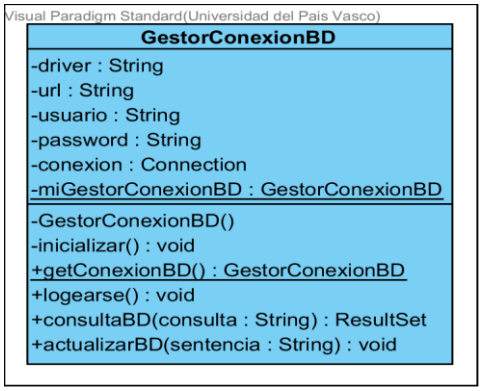



Ilustración 28: GESTOR CONEXIÓN BD

Clase que sigue el patrón Singleton y se encarga de gestionar la conexión con la base de datos. Proporciona métodos para realizar consultas sobre la base de datos.

#### GestorUsuarios

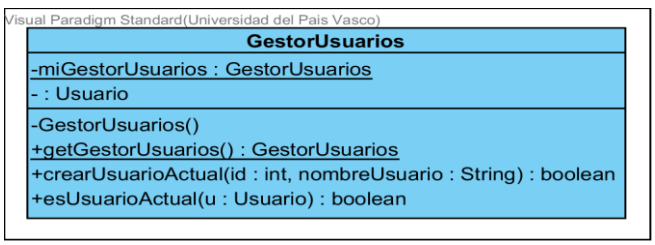



Ilustración 29: GESTOR USUARIOS

Clase que sigue el patrón Singleton y se encarga de gestionar la información del usuario que está utilizando la aplicación.

### Usuario

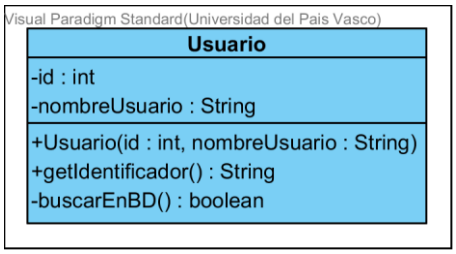



Ilustración 30: USUARIO

Esta clase contiene la información básica de un usuario.

## 5.2. Space-Defense

En este apartado se explicarán las diferentes clases que componen la aplicación Space-Defense.

Dado que muchas de las clases utilizadas están presentes en el motor gráfico, se mostrarán únicamente en el diagrama de clases, a excepción de aquellas que se hayan modificado. Estas clases se mostrarán en el apartado de clases explicando las modificaciones que se han llevado a cabo.

### 5.2.1. Diagrama de clases

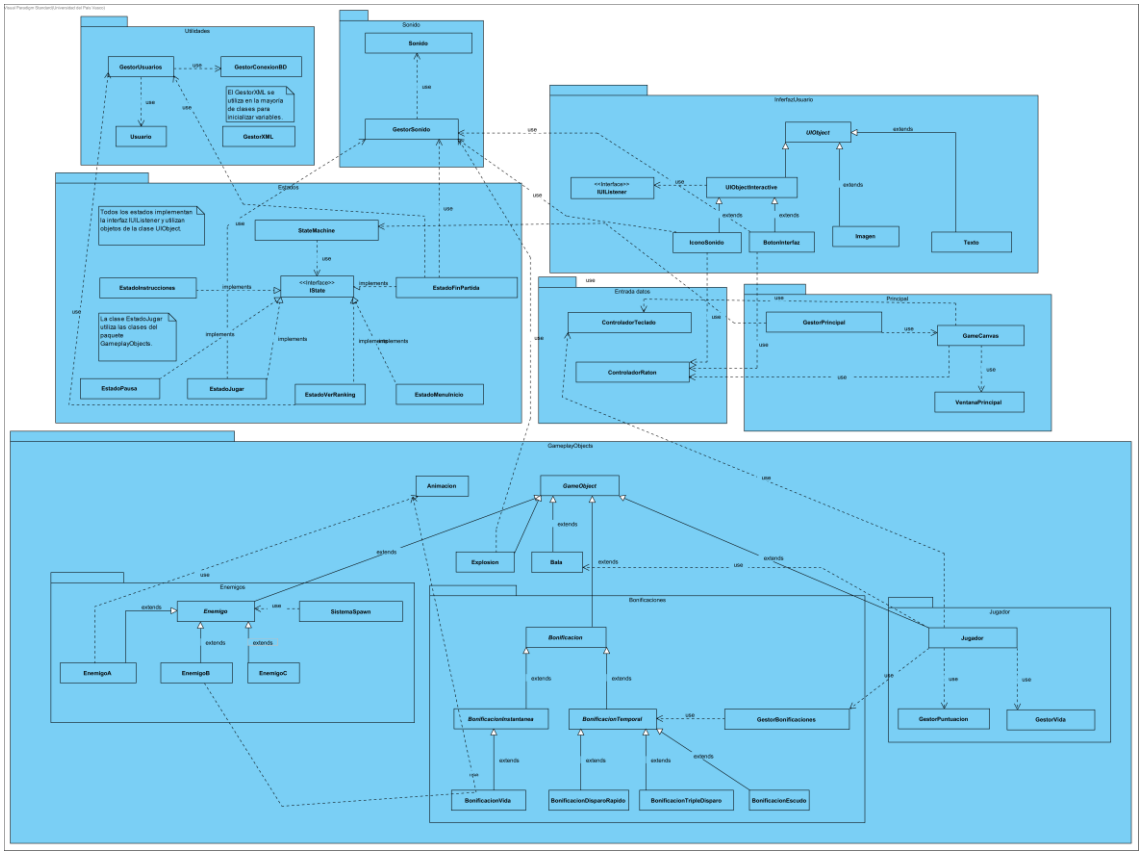



Ilustración 31: DIAGRAMA DE CLASES SPACE-DEFENSE

### 5.2.2. Clases

#### 5.2.2.1. Módulo Estados

#### EstadoMenuInicio

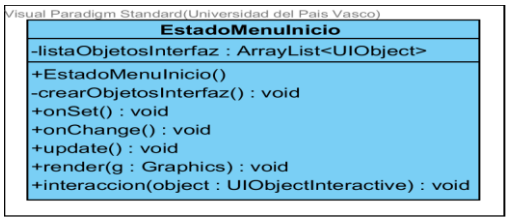



Ilustración 32: ESTADO MENÚ INICIO

Esta clase se encarga de crear y mostrar al usuario la pantalla de inicio de la aplicación y gestionar la interacción del usuario con los botones que contiene.

#### EstadoInstrucciones

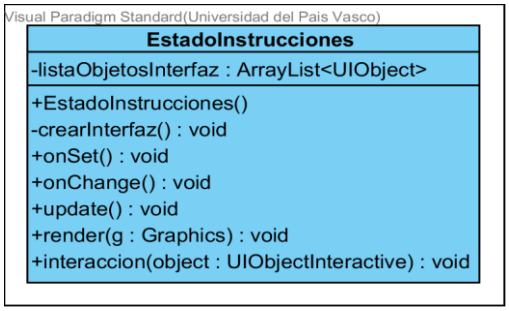



Ilustración 33: ESTADO INSTRUCCIONES

Esta clase se encarga de crear y mostrar al usuario la pantalla de instrucciones de la aplicación con los controles y las reglas para poder jugar, así como gestionar la interacción del usuario con los botones que contiene.

#### EstadoVerRanking

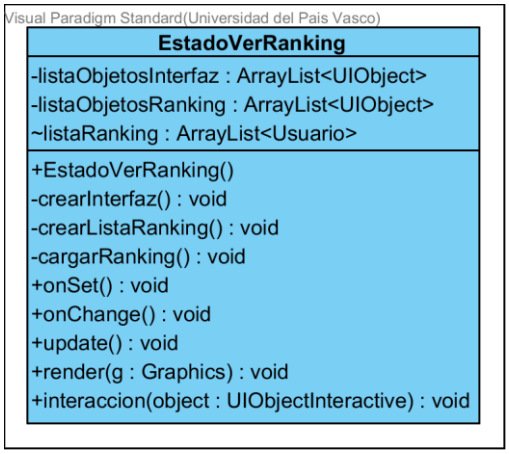



Ilustración 34: ESTADO VER RANKING

Esta clase se encarga de crear y mostrar al usuario la pantalla con el ranking de las mejores puntuaciones obtenidas y la mejor puntuación del propio usuario. También gestiona la interacción del usuario con los botones que contiene.

### EstadoJugar

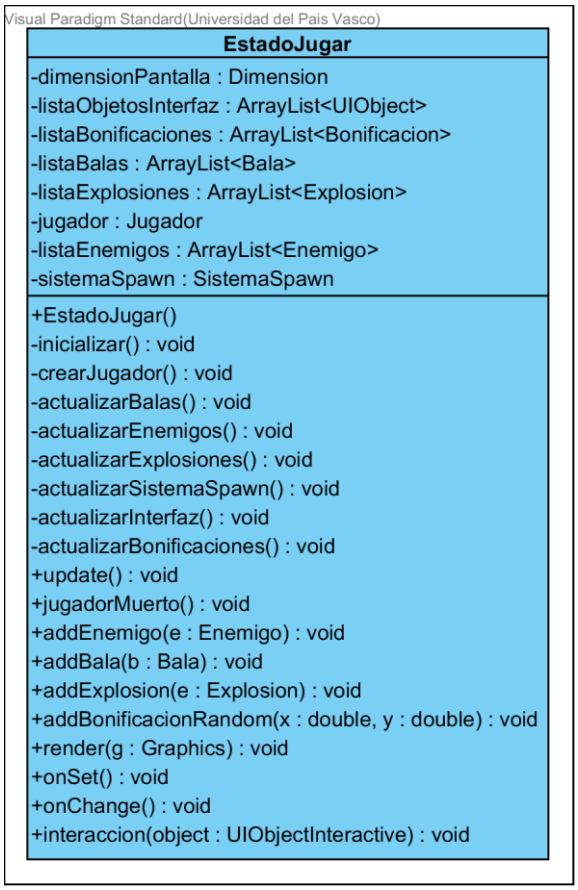



Ilustración 35: ESTADO JUGAR

Esta clase se encarga de crear y mostrar al usuario la pantalla de juego y gestionar la interacción del usuario a través del teclado y ratón, así como los botones que contiene. Se hace uso de las clases del módulo Gameplay Objects.

### EstadoPausa

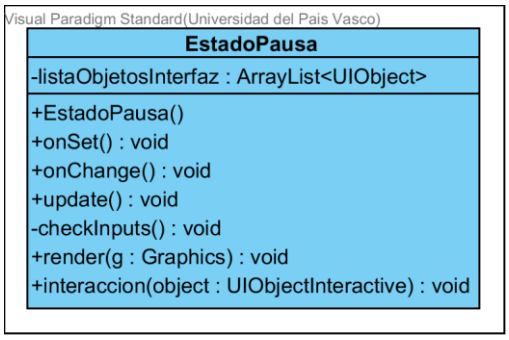



Ilustración 36: ESTADO PAUSA

Esta clase es la encargada de crear y mostrar al usuario la pantalla de pausa y gestionar la interacción del usuario mediante teclado y botones.

### EstadoFinPartida

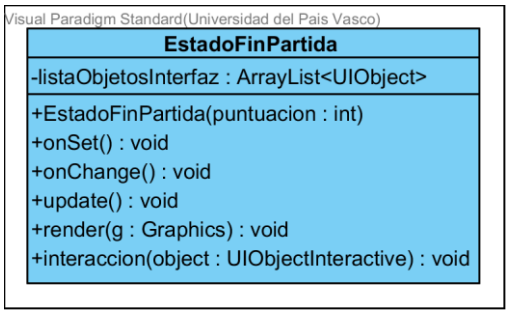



Ilustración 37: ESTADO FIN PARTIDA

Esta clase es la encargada de crear y mostrar al usuario la pantalla una vez finalizada la partida con la puntuación obtenida y actualizar esta puntuación en la base de datos si fuera la mejor obtenida. También es responsable de la gestión de la interacción del usuario con los botones que contiene.

### StateMachine

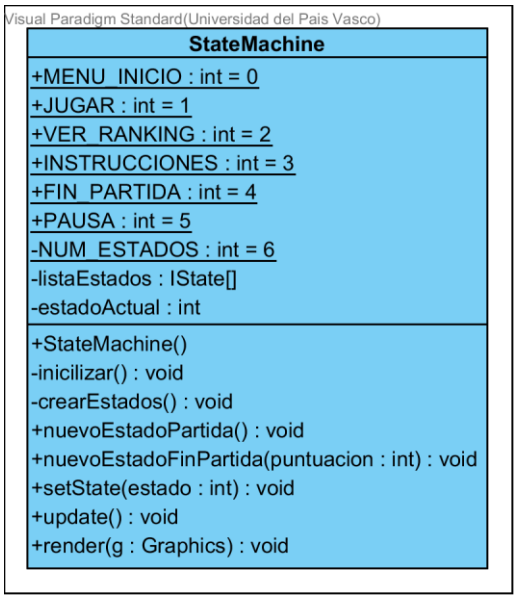



Ilustración 38: STATE MACHINE

Clase presente en el módulo correspondiente del motor gráfico a la que se le han añadido algunos métodos para gestionar el cambio entre algunos estados con un comportamiento diferente al resto.

#### 5.2.2.2. Módulo Gameplay Objects

Dentro de este módulo se encuentran las clases que se encargan de dotar de jugabilidad a la aplicación.

### GameObject

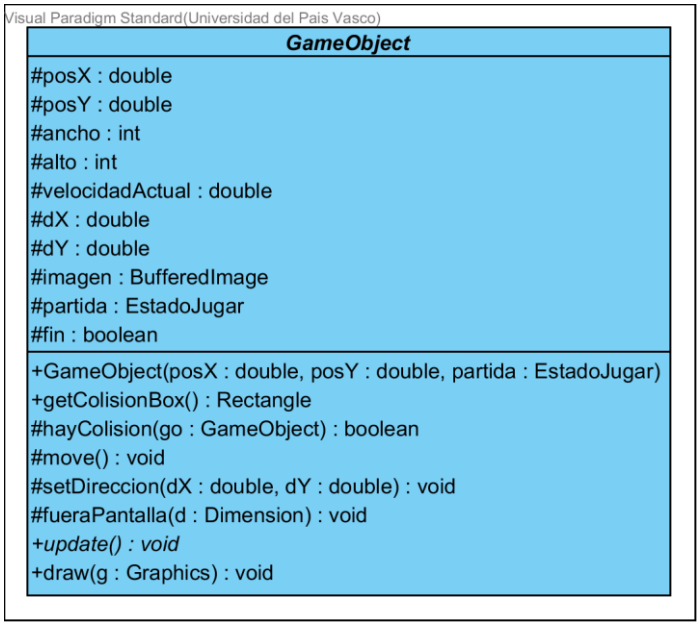



Ilustración 39: GAME OBJECT

Clase abstracta que proporciona los atributos y métodos necesarios para cualquier objeto utilizado en el juego.

### Animacion

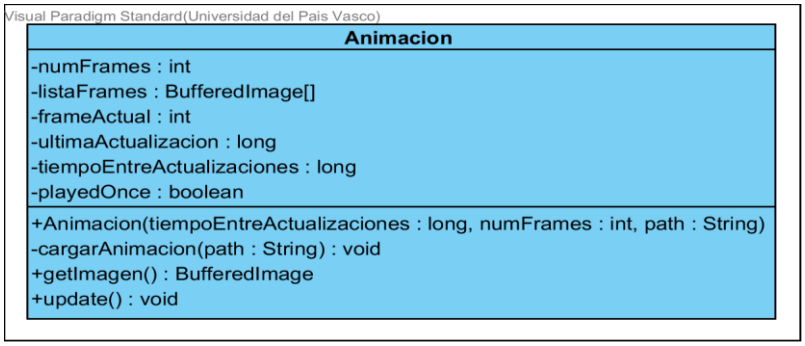



Ilustración 40: ANIMACIÓN

Esta clase contiene una imagen que irá cambiando con el paso del tiempo para intentar representar de este modo una animación.

### Explosion

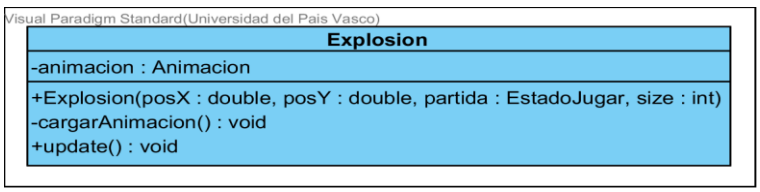



Ilustración 41: EXPLOSIÓN

Esta clase se encarga de renderizar en pantalla una explosión

### Jugador


Ilustración 42: JUGADOR

Clase que extiende de GameObject y representa la nave que controla el jugador en la partida.

### GestorVida

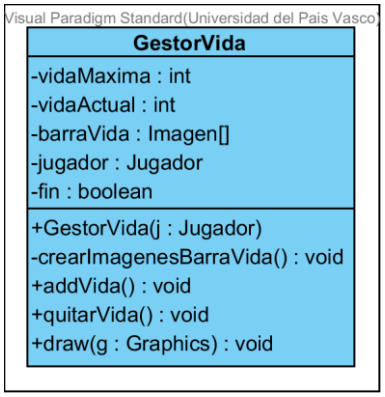



Ilustración 43: GESTOR VIDA

Clase que se encarga de gestionar las vidas del jugador.

### GestorPuntuacion

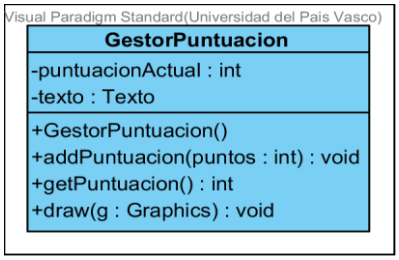



Ilustración 44: GESTOR PUNTUACIÓN

Clase que se encarga de gestionar la puntuación del jugador.

### Enemigo

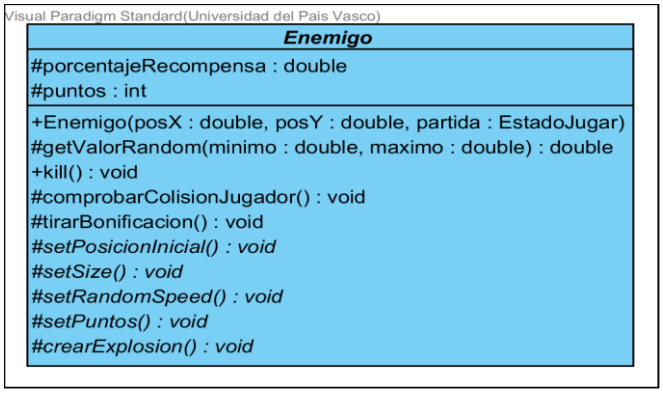



Ilustración 45: ENEMIGO

Clase abstracta que extiende de GameObject de la que heredarán cada uno de los enemigos del juego que se explicarán a continuación. Cada enemigo tendrá una funcionalidad única.

### EnemigoA

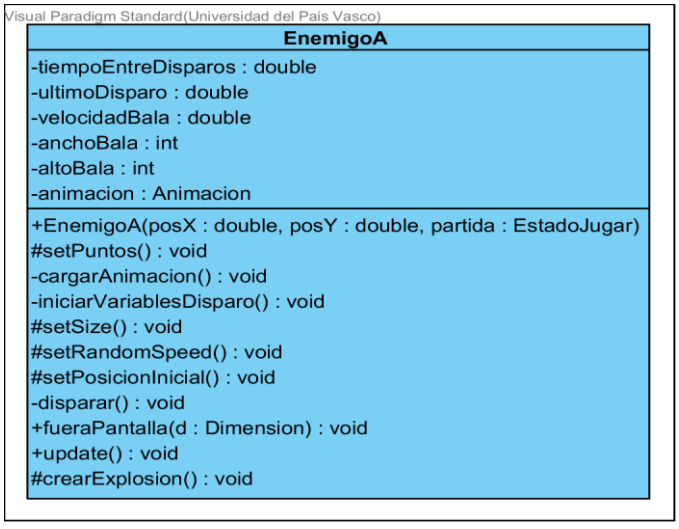



Ilustración 46: ENEMIGO A

La función de este enemigo es moverse en dirección vertical hacia abajo y disparar balas cada cierto intervalo de tiempo.

### EnemigoB

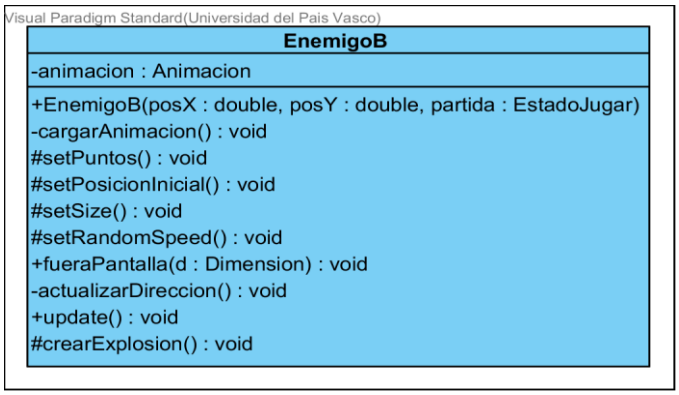



Ilustración 47: ENEMIGO B

La función de este tipo de enemigo es perseguir constantemente al jugador hasta alcanzarle.

### EnemigoC

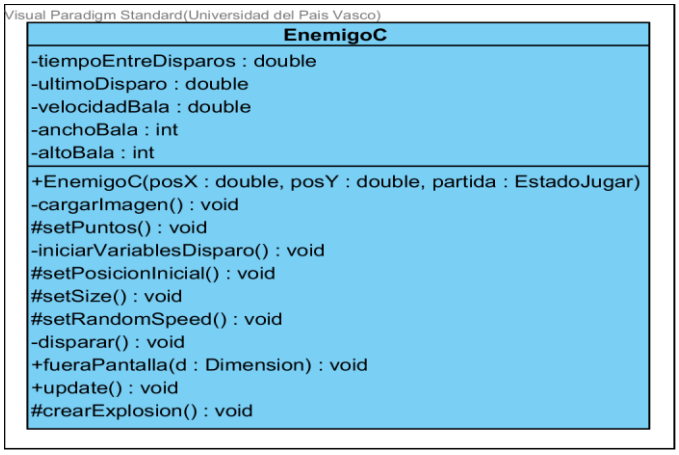



Ilustración 48: ENEMIGO C

La función de este tipo de enemigo es moverse horizontalmente disparando balas cada cierto periodo de tiempo.

### Bala

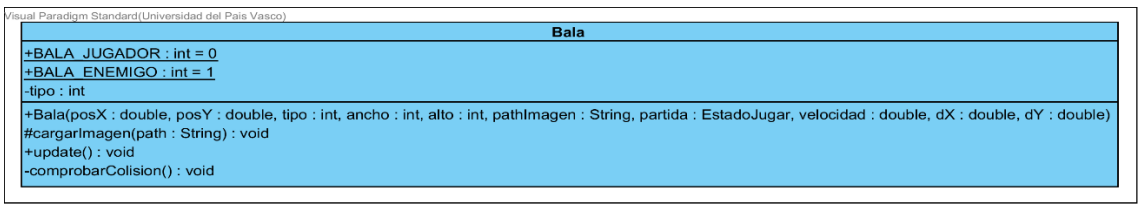



Ilustración 49: BALA

Clase que extiende de GameObject y representa la bala que usarán tanto el jugador como los enemigos para disparar.

### Bonificacion

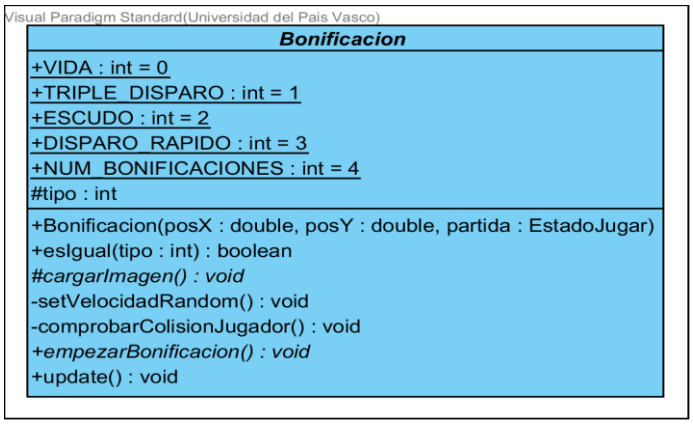



Ilustración 50: BONIFICACIÓN

Clase abstracta que extiende de GameObject de la cual heredarán cada uno de los tipos de bonificaciones que se explicarán a continuación.

### BonificacionInstantanea

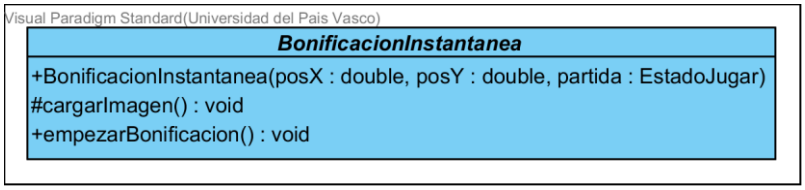



Ilustración 51: BONIFICACIÓN INSTANTÁNEA

Clase abstracta que representa a aquellas bonificaciones que sean de un único uso.

### BonificacionTemporal

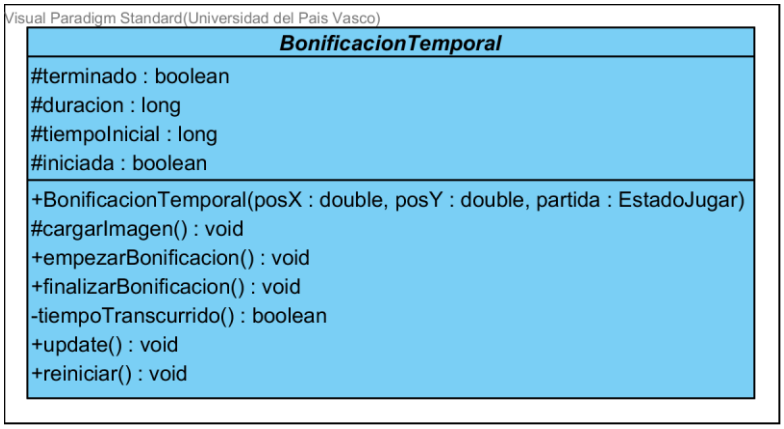



Ilustración 52: BONIFIACIÓN TEMPORAL

Clase abstracta que representa a las bonificaciones que perduran a lo largo de un periodo de tiempo.

### BonificacionVida

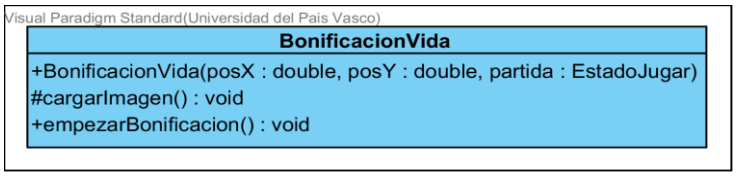



Ilustración 53: BONIFICACIÓN VIDA

Clase que extiende de BonificacionInstantanea y que añade una vida al jugador al recogerla.

### BonificacionEscudo

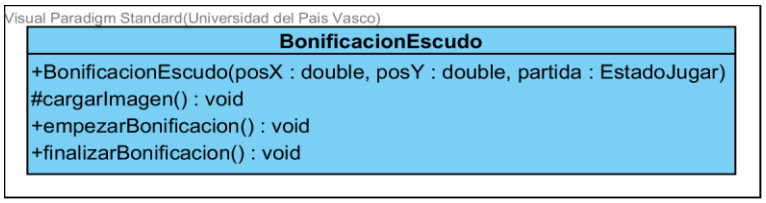



Ilustración 54: BONIFICACIÓN ESCUDO

Clase que extiende de BonificacionTemporal y proporciona inmunidad ante el daño al jugador durante un breve periodo de tiempo al recogerla.

### BonificacionDisparoRapido

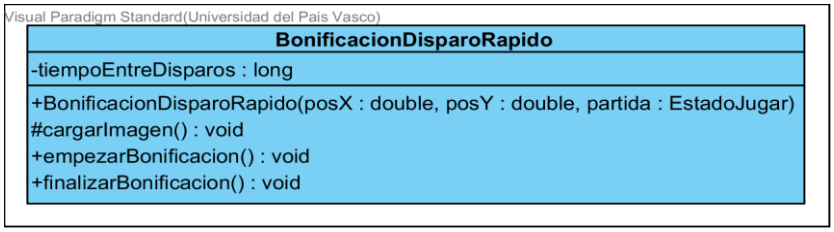



Ilustración 55: BONIFICACIÓN DISPARO RÁPIDO

Clase que extiende de BonificacionTemporal y proporciona al jugador cuando la recoge una frecuencia de disparo mayor durante un breve periodo de tiempo.

### BonificacionTripleDisparo

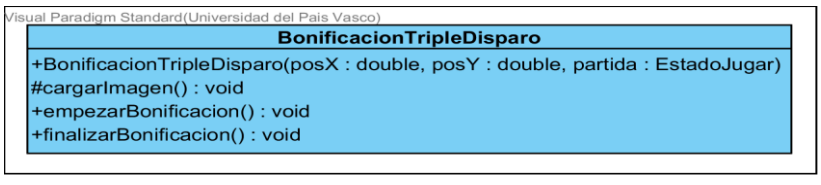



Ilustración 56: BONIFICACIÓN TRIPLE DISPARO

Clase que extiende de BonificacionTemporal y hace que el jugador dispare tres balas en vez de una durante un breve periodo de tiempo después de recogerla.

#### 5.2.2.3. Módulo principal

### GestorPrincipal

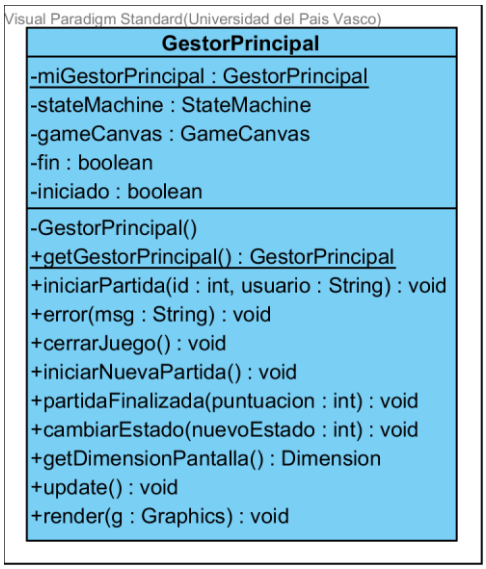



Ilustración 57: GESTOR PRINCIPAL

Clase presente en el módulo correspondiente del motor gráfico a la que se le han añadido algunos métodos para realizar la gestión de los cambios de estado.

#### 5.2.2.4. Módulo Utilidades

### GestorUsuarios

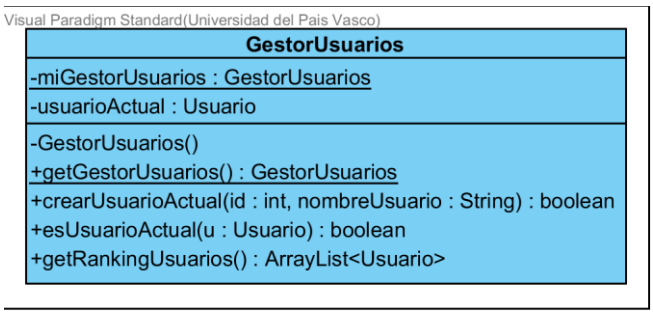



Ilustración 58: GESTOR USUARIOS

Clase presente en el módulo correspondiente del motor gráfico a la que se le ha añadido un método para obtener el ranking de usuarios.

### Usuario

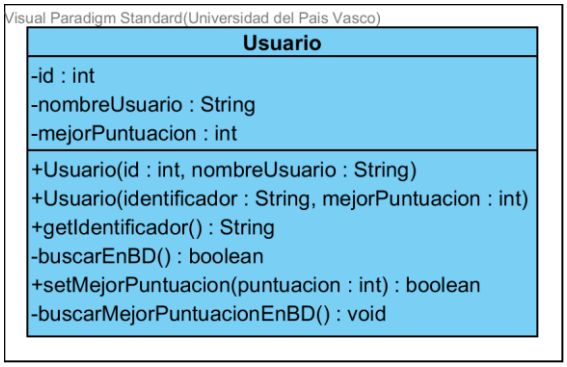



Ilustración 59: USUARIO

Clase presente en el módulo correspondiente del motor gráfico a la cual se le ha añadido la funcionalidad para gestionar la puntuación del jugador.

## 5.3. Ping-Pong

### 5.3.1. Cliente

En este apartado se explicarán las clases que componen la aplicación cliente del Ping-Pong.

Dado que muchas de las clases utilizadas están presentes en el motor gráfico, se mostrarán únicamente en el diagrama de clases, a excepción de aquellas que se hayan modificado. Estas clases se mostrarán en el apartado de clases explicando las modificaciones que se han llevado a cabo.

#### 5.3.1.1. Diagrama de clases

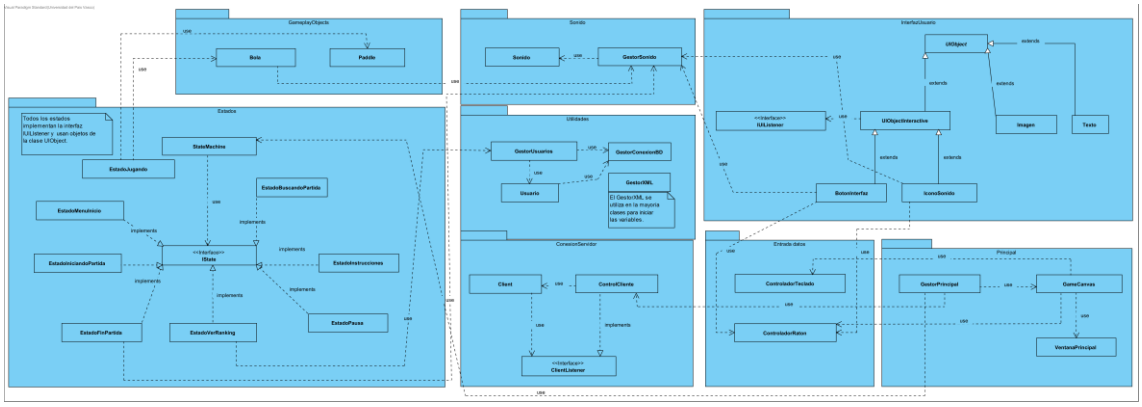


Ilustración 60: DIAGRAMA DE CLASES CLIENTE PING-PONG

#### 5.3.1.2. Clases

##### 5.3.1.2.1. Módulo conexión servidor

Las clases de este módulo se encargan de gestionar la conexión con el servidor.

**Client**

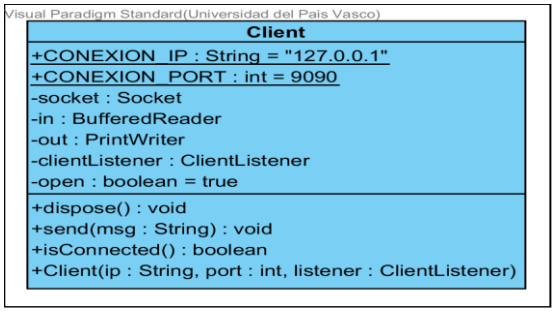



Ilustración 61: CLIENT

Clase encargada de conectarse al servidor a través de una ip y un puerto y mantener la conexion activa.

**ClientListener**

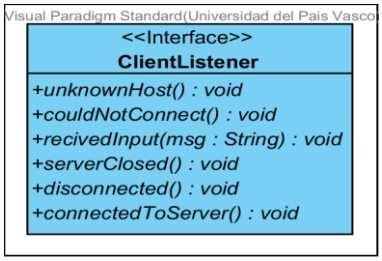



Ilustración 62: CLIENT LISTENER

Interfaz que deberá implementar la clase encargada de gestionar la conexión al servidor.

**ControlCliente**

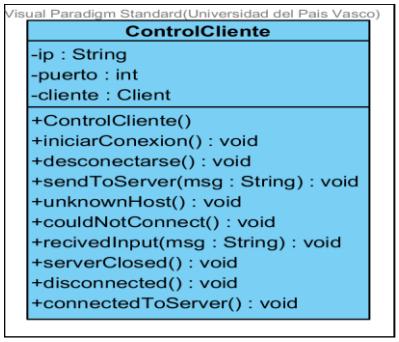



Ilustración 63: CONTROL CLIENTE

Clase que implementa la interfaz ClientListener y gestiona la conexión al servidor (mensajes recibidos, envío mensajes, errores etc).

#### 5.3.1.2.2. Módulo estados

### IState

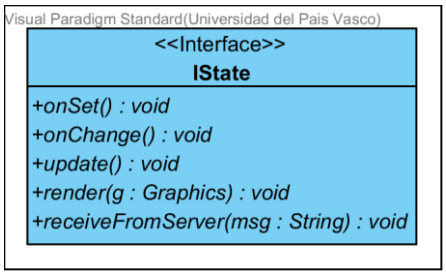



Ilustración 64: ISTATE

Interfaz presente en el motor gráfico a la que se le ha añadido un método para recibir los mensajes del servidor.

### StateMachine


Ilustración 65: STATE MACHINE

Clase presente en el motor gráfico a la que se ha añadido la funcionalidad para recibir mensajes del servidor y gestionar los estados específicos de la aplicación Ping-Pong.

### EstadoMenuInicio

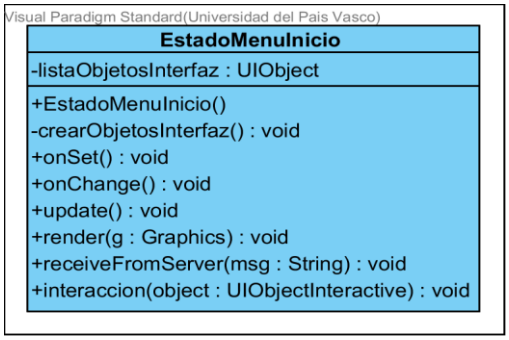



Ilustración 66: ESTADO MENU INICIO

Esta clase se encarga de crear y mostrar al usuario la pantalla de inicio de la aplicación y gestionar la interacción del usuario con los botones que contiene.

### EstadoVerRanking

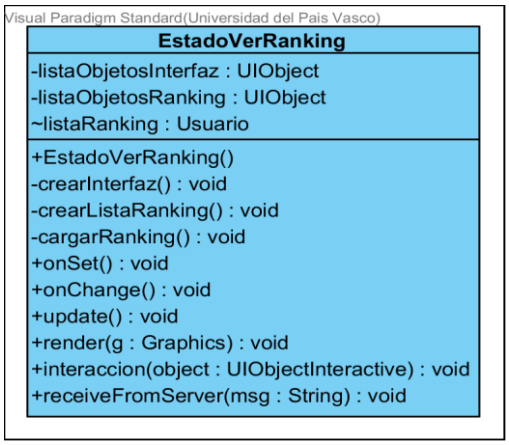



Ilustración 67: ESTADO VER RANKING

Esta clase se encarga de crear y mostrar al usuario la pantalla con el ranking de los mejores usuarios con su Elo y el Elo del propio usuario. También gestiona la interacción del usuario con los botones que contiene.

### EstadoInstrucciones

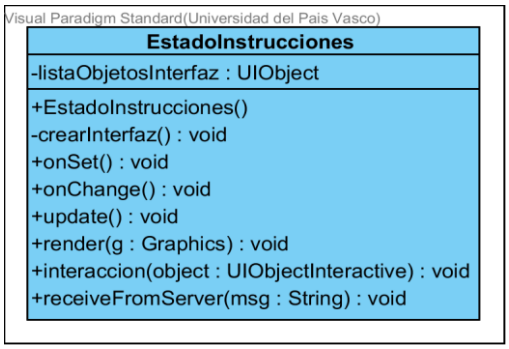



Ilustración 68: ESTADO INSTRUCCIONES

Esta clase se encarga de crear y mostrar al usuario la pantalla de instrucciones de la aplicación con los controles y las reglas para poder jugar, así como gestionar la interacción del usuario con los botones que contiene.

### EstadoBuscandoPartida

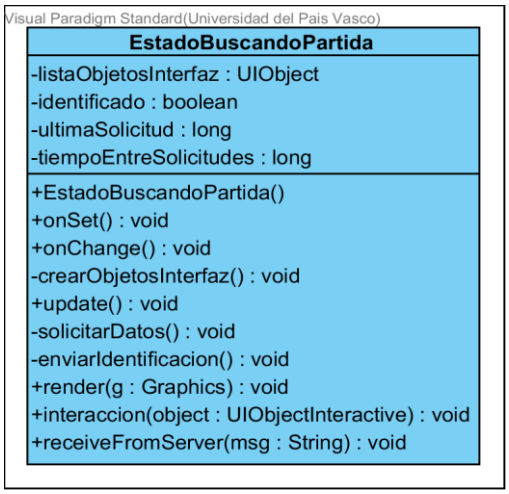



Ilustración 69: ESTADO BUSCANDO PARTIDA

Esta clase se encarga de crear y mostrar al usuario la pantalla de espera hasta el encuentro de una partida. También gestiona la interacción del usuario con los botones que contiene y se comunica con el servidor para el envío y recibo de datos.

### EstadoIniciandoPartida

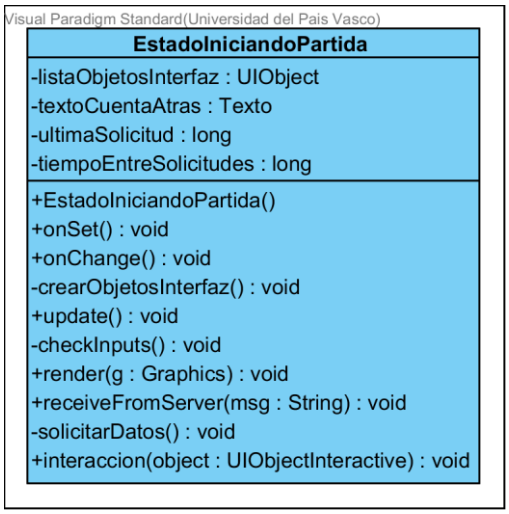



Ilustración 70: ESTADO INICIANDO PARTIDA

Esta clase se encarga de crear y mostrar al usuario la pantalla de espera hasta el inicio de una partida. También gestiona la interacción del usuario mediante los botones que contiene o mediante el teclado y se comunica con el servidor para el envío y recibo de datos.

### EstadoJugando

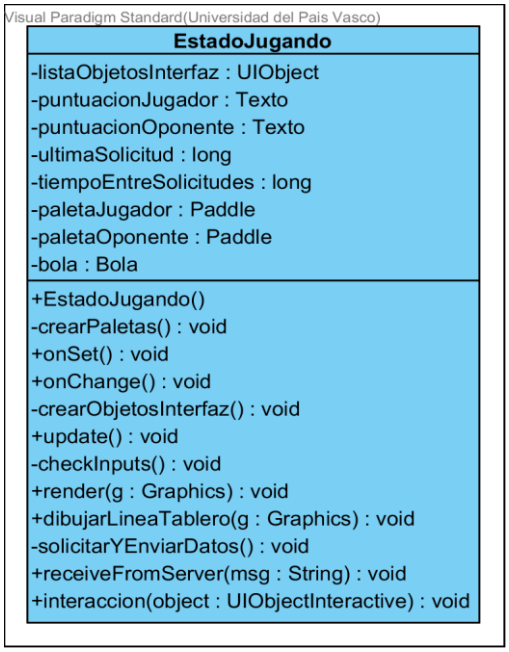



Ilustración 71: ESTADO JUGANDO

Esta clase se encarga de crear y mostrar al usuario la pantalla de juego. También gestiona la interacción del usuario mediante los botones que contiene o mediante el teclado y se comunica con el servidor para el envío y recibo de datos.

### EstadoPausa

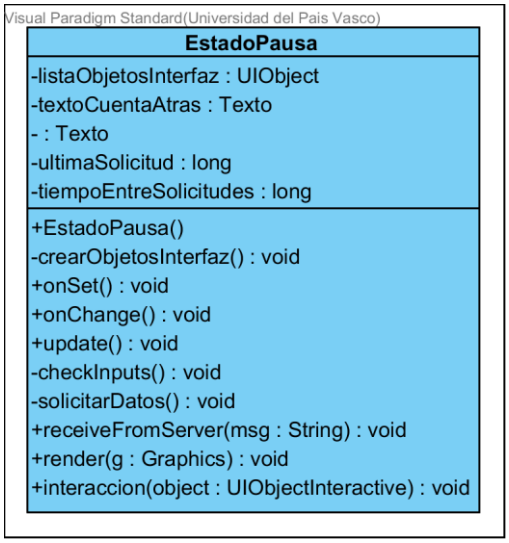



Ilustración 72: ESTADO PAUSA

Esta clase se encarga de crear y mostrar al usuario la pantalla de pausa. También gestiona la interacción del usuario mediante los botones que contiene o mediante el teclado y se comunica con el servidor para el envío y recibo de datos.

### EstadoFinPartida

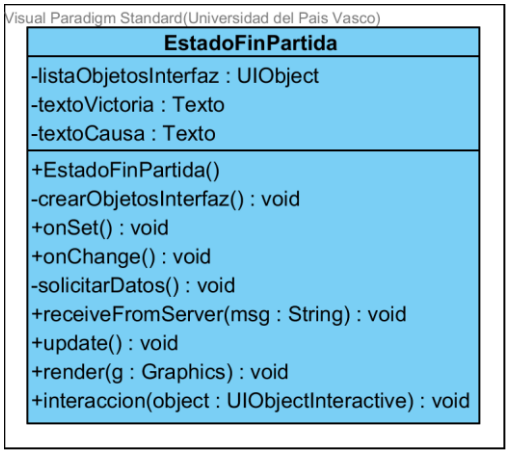



Ilustración 73: ESTADO FIN PARTIDA

Esta clase se encarga de crear y mostrar al usuario la pantalla de fin de partida indicándole si ha ganado o perdido. También gestiona la interacción del usuario con los botones que contiene y se comunica con el servidor para el envío y recibo de datos.

#### 5.3.1.2.3. Módulo Gameplay Objects

Dentro de este módulo se encuentra las clases usadas en la parte jugable de la aplicación.

### Bola

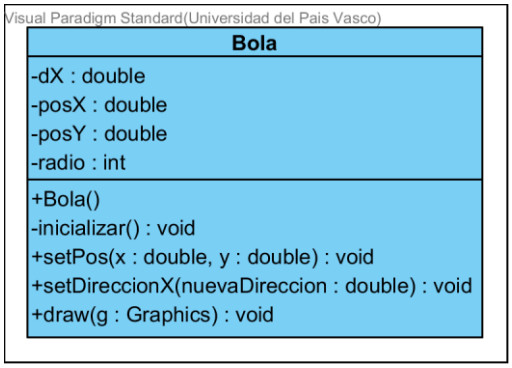



Ilustración 74: BOLA

Esta clase representa la bola con la que se juega una partida.

## Paddle

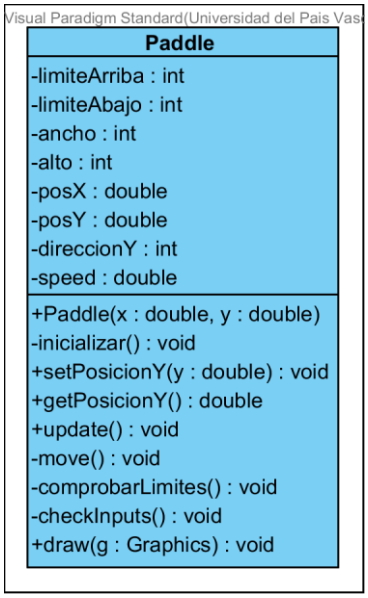



Ilustración 75: PADDLE

Esta clase representa la paleta que utiliza el jugador para golpear la bola.

### 5.3.1.2.4. Módulo principal

## GestorPrincipal

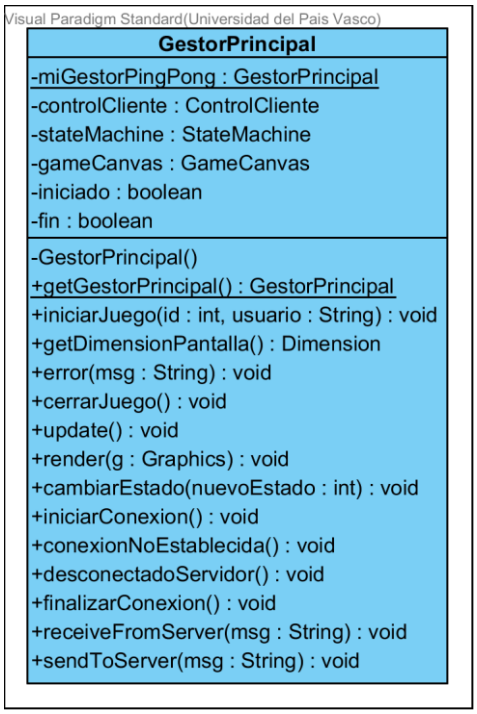



Ilustración 76: GESTOR PRINCIPAL

Clase presente en el módulo correspondiente del motor gráfico a la que se le han añadido algunos métodos para realizar la gestión de los cambios de estado y la conexión con el servidor.

#### 5.3.1.2.5. Módulo utilidades

### GestorUsuarios

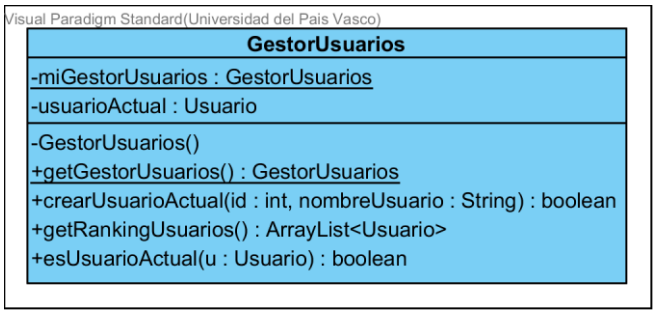



Ilustración 77: GESTOR USUARIOS

Clase presente en el módulo correspondiente del motor gráfico a la que se le ha añadido un método para obtener el ranking de usuarios.

### Usuario

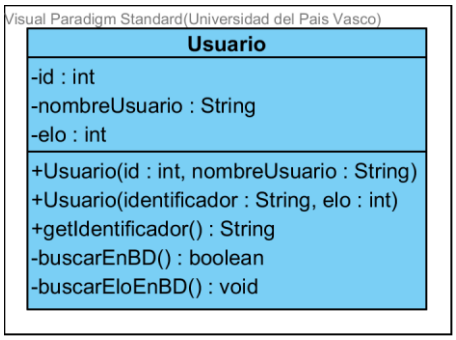



Ilustración 78: USUARIO

Clase presente en el módulo correspondiente del motor gráfico a la cual se le añadido la funcionalidad para gestionar el Elo del jugador que representa su nivel de habilidad.

### 5.3.2. Servidor

En este apartado se explicarán las clases que componen la aplicación servidor del Ping-Pong. Se han dividido las clases en módulos para facilitar la comprensión.

#### 5.3.2.1. Diagrama de clases

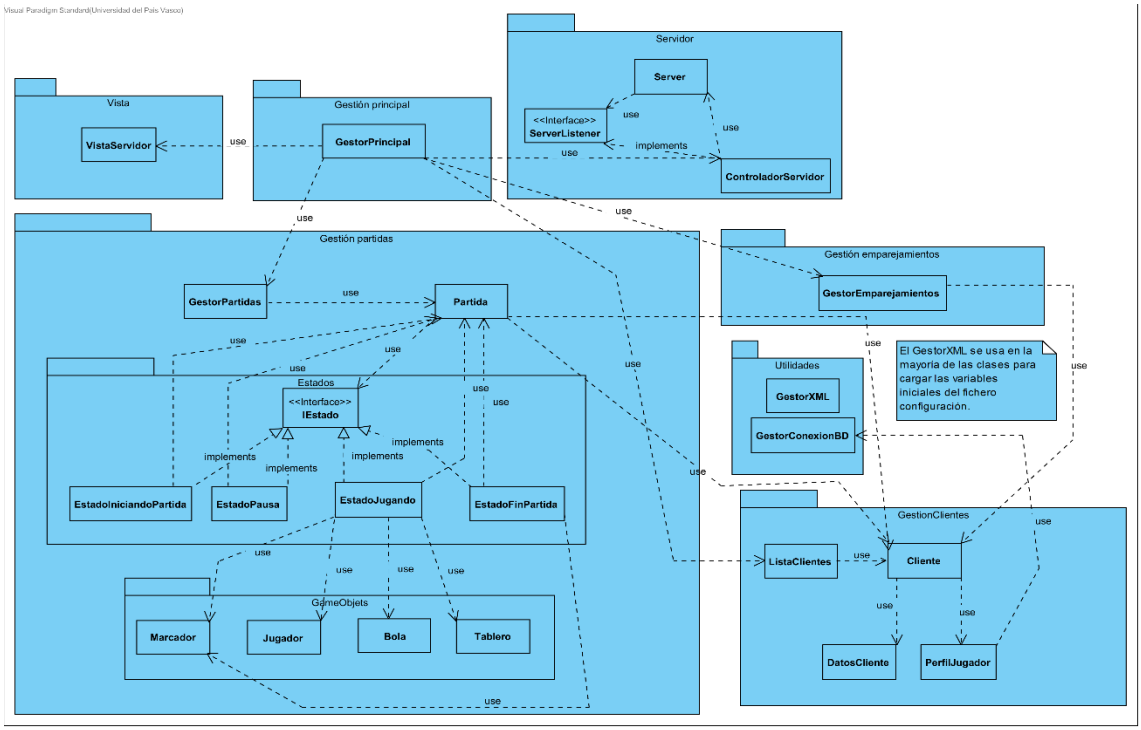



Ilustración 79: DIAGRAMA DE CLASES SERVIDOR PING-PONG

#### 5.3.2.2. Clases

##### 5.3.2.2.1. Módulo gestión clientes

Las clases de este módulo se encargan de gestionar los usuarios que se conectan al servidor.

**DatosCliente**

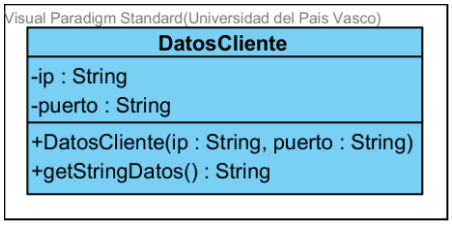



Ilustración 80: DATOS CLIENTE

Clase que sirve para almacenar la ip y puerto de un determinado usuario conectado al servidor.

### PerfilJugador

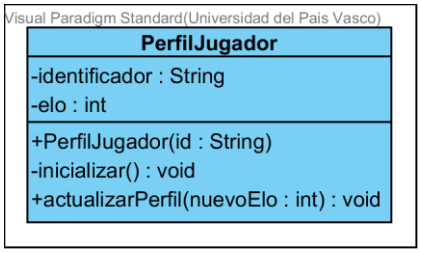



Ilustración 81: PERFIL JUGADOR

Clase que se encarga de buscar en la base de datos y actualizar el Elo de un jugador.

### Cliente

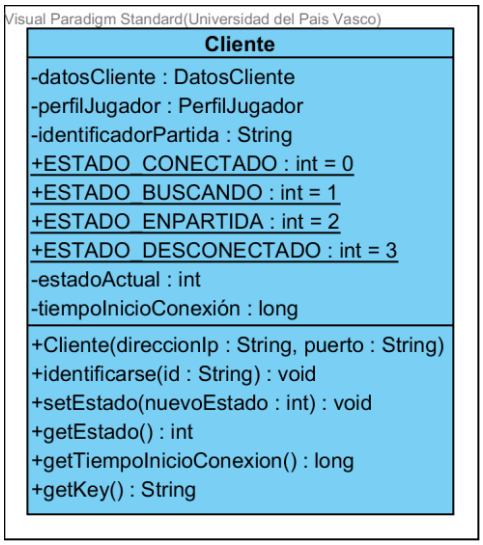



Ilustración 82: CLIENTE

Clase que representa a un usuario conectado al servidor. Almacena y gestiona el estado en el que se encuentra y otro tipo de información haciendo uso de las clases DatosCliente y PerfilJugador.

### ListaClientes

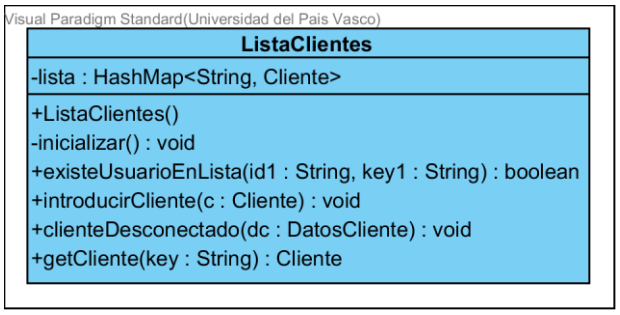



Ilustración 83: LISTA CLIENTES

Clase encargada de gestionar la lista de usuarios conectados al servidor.

#### 5.3.2.2.2. Módulo servidor

Las clases de este módulo se encargan del funcionamiento del servidor.

**ServerListener**

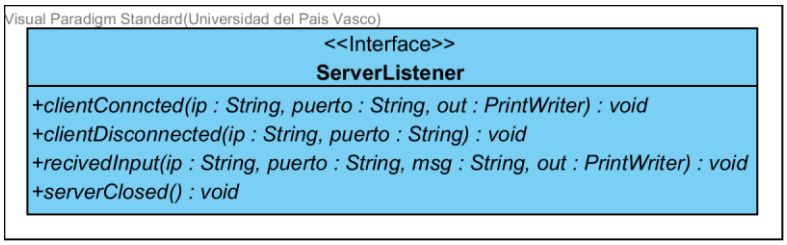



Ilustración 84: SERVER LISTENER

Interfaz que proporciona los métodos necesarios para controlar la gestión de los clientes que se conectan al servidor.

**Server**

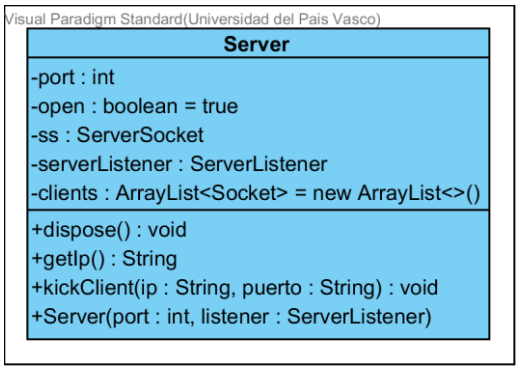



Ilustración 85: SERVER

Clase que trabaja dentro de su propio Thread y se encarga de las siguientes tareas:

- Iniciar y cerrar el servidor.
- Gestionar la conexión de un nuevo cliente.
- Gestionar la desconexión de un cliente.
- Recibir los mensajes de los clientes.
- Echar a un cliente del servidor.

Para todo ello se avisa al ServerListener que se encargará de tomar las acciones correspondientes en cada caso.

### ControladorServidor

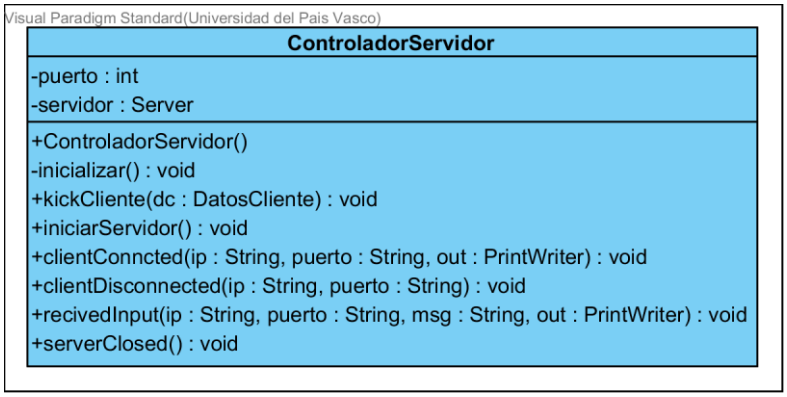



Ilustración 86: CONTROLADOR SERVIDOR

Clase que implemente la interfaz ServerListenner y se encarga de gestionar los clientes que se conectan al servidor.

#### 5.3.2.2.3. Módulo gestión principal

### GestorPrincipal

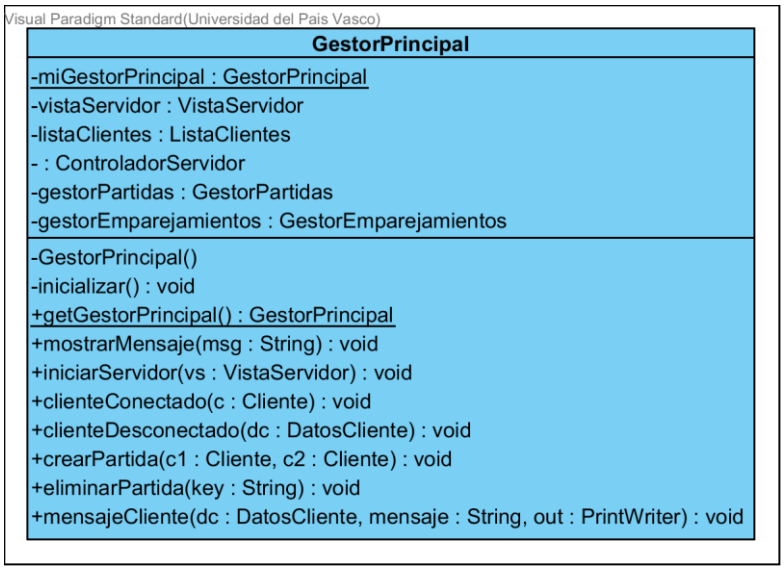



Ilustración 87: GESTOR PRINCIPAL

Clase Singleton que se encarga de gestionar las partidas, el listado clientes y el sistema emparejamiento.

### 5.3.2.2.4. Módulo gestión partidas

### IEstado

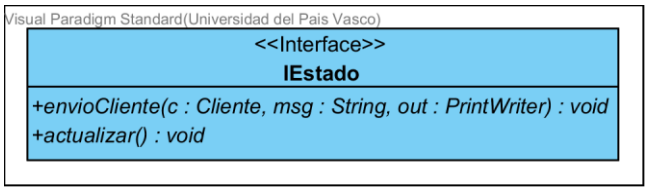



Ilustración 88: IESTADO

Interfaz que proporciona los métodos que deberán implementar cada estado.

### EstadoIniciandoPartida


Ilustración 89: ESTADO INICIANDO PARTIDA

Clase que implemente IState y se encarga de realizar la cuenta atrás desde que se crea una partida hasta que comienza. También se encarga de recibir los mensajes de los clientes y de enviarles la respuesta que corresponda.

### EstadoJugando

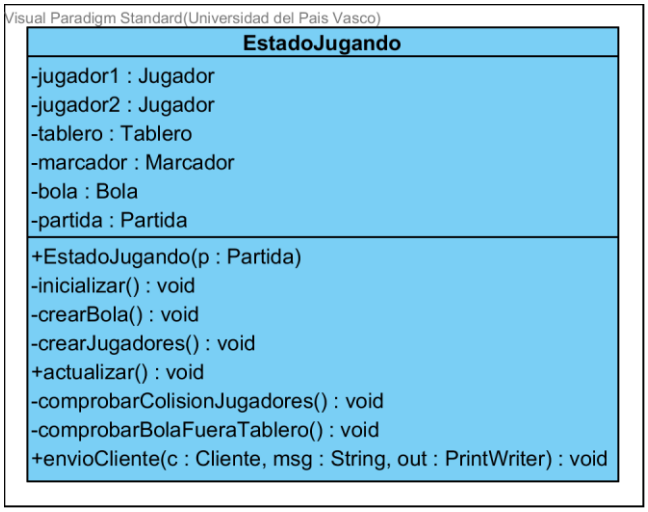



Ilustración 90: ESTADO JUGANDO

Clase que implemente IState y se encarga de gestionar la parte jugable (bola, marcador, tablero y jugadores). También se encarga de recibir los mensajes de los clientes y de enviarles la respuesta que corresponda.

### EstadoPausa

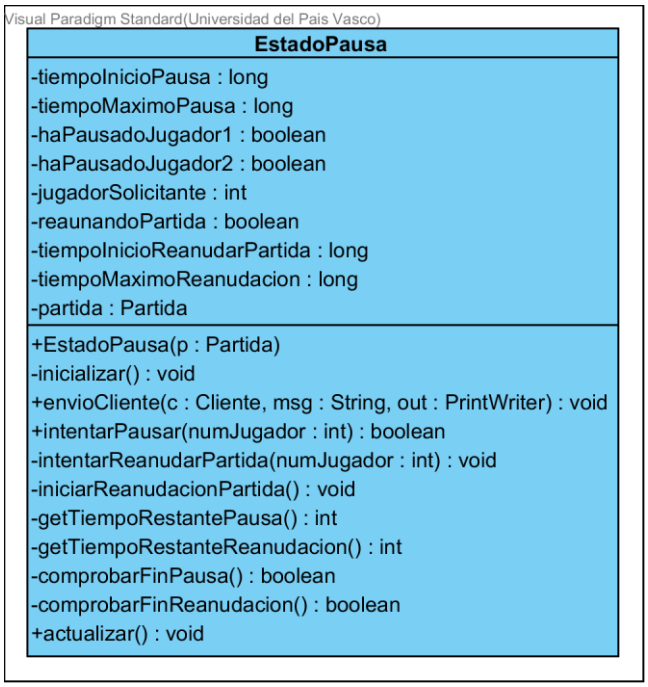



Ilustración 91: ESTADO PAUSA

Clase que implemente IState y se encarga de gestionar las pausas de la partida. También se encarga de recibir los mensajes de los clientes y de enviarles la respuesta que corresponda.

### EstadoFinPartida

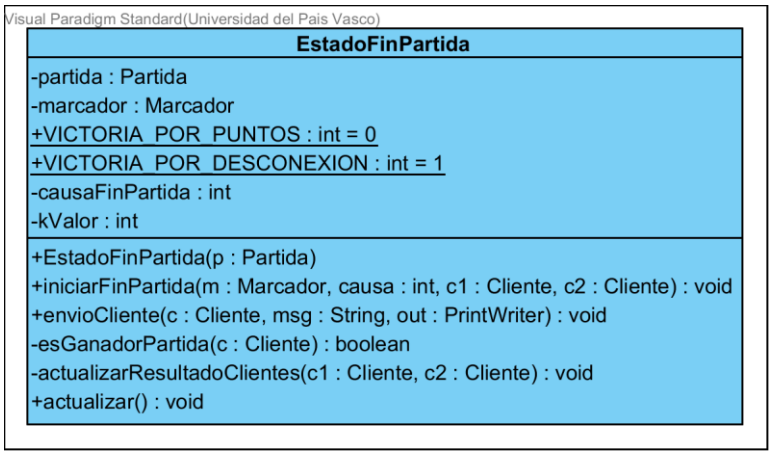



Ilustración 92: ESTADO FIN PARTIDA

Clase que implementa IState y se encarga de determinar el ganador de la partida y actualizar el Elo de los jugadores según corresponda. También se

encarga de recibir los mensajes de los clientes y de enviarles la respuesta que corresponda.

### Bola

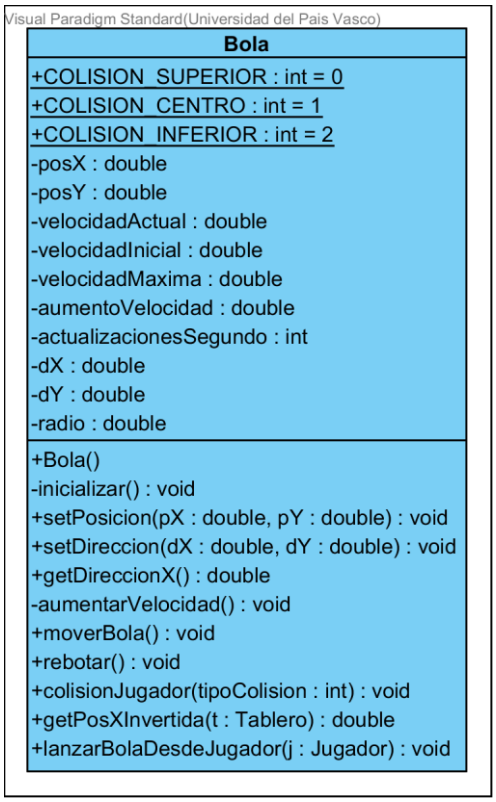



Ilustración 93: BOLA

Representa la bola con la que se juega la partida.

### Jugador

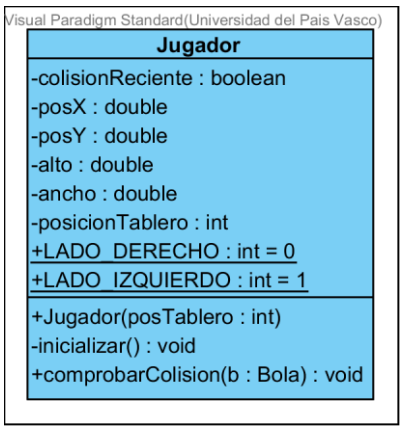



Ilustración 94: JUGADOR

Clase que representa la paleta con la que juega el jugador.

### Marcador

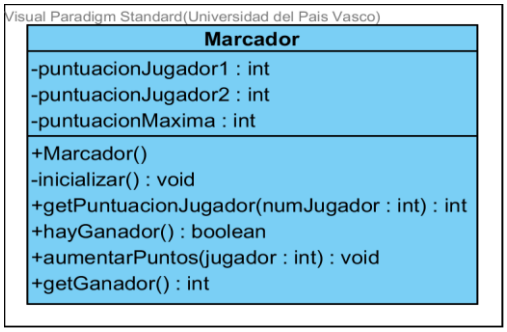



Ilustración 95: MARCADOR

Esta clase se encarga de gestionar la puntuación de los jugadores en una partida.

### Tablero

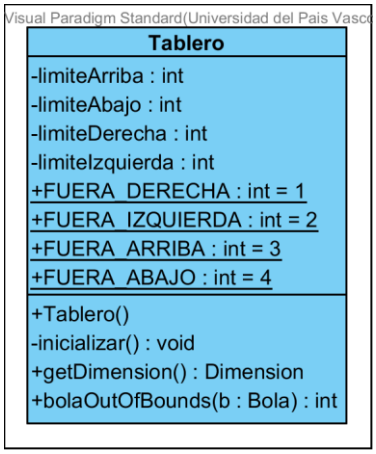



Ilustración 96: TABLERO

Esta clase contiene la información sobre la dimensión del tablero que se usará para determinar si la bola esta fuera de los límites del tablero.

## Partida

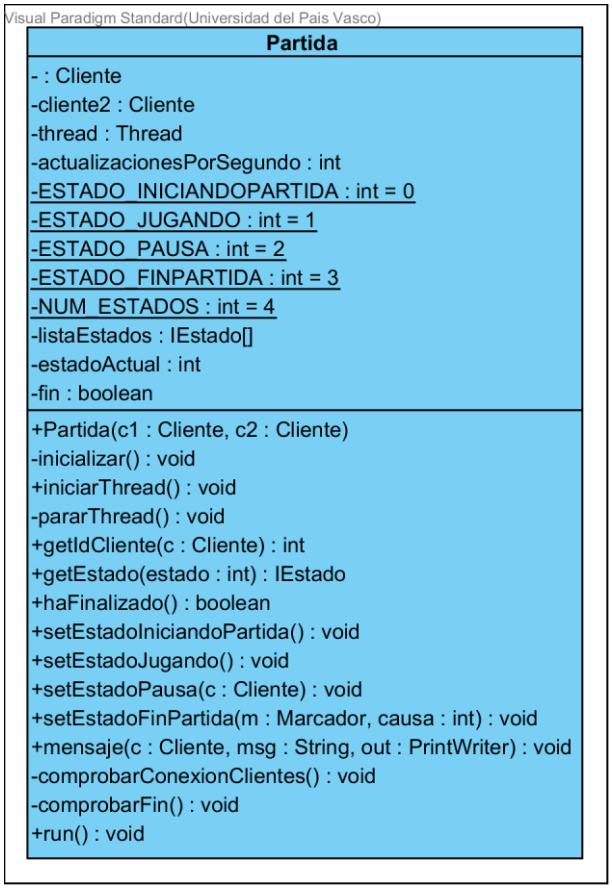



Ilustración 97: PARTIDA

Esta clase corre dentro de su propio Thread y se encarga de gestionar la partida entre dos usuarios cambiando entre los diferentes estados y actualizándolos constantemente.

## GestorPartidas

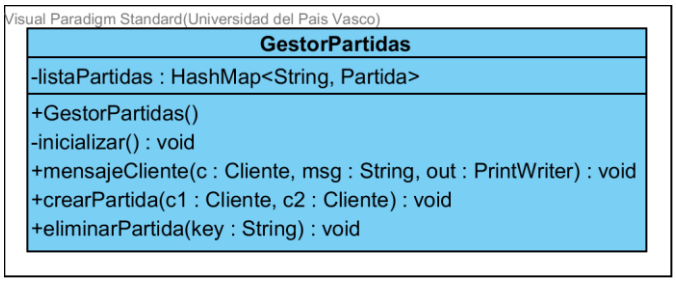



Ilustración 98: GESTOR PARTIDAS

Esta clase se encarga de gestionar todas las partidas activas del servidor.

#### 5.3.2.2.5. Módulo gestión emparejamientos

### GestorEmparejamientos

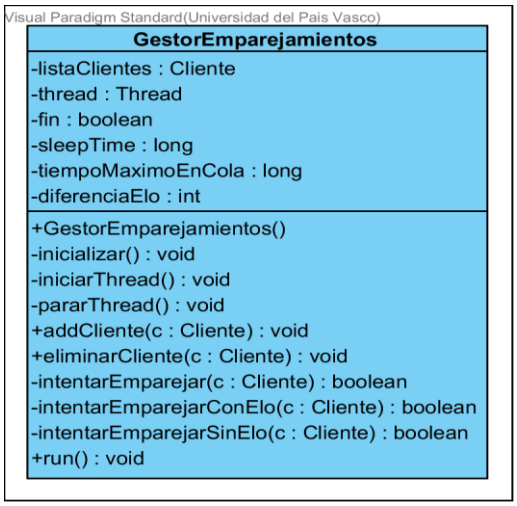



Ilustración 99: GESTOR EMPAREJAMIENTOS

Clase que funciona dentro de su propio Thread y se encarga de emparejar a los jugadores y crear las partidas entre ellos.

## 5.4. Plataforma de juego

En este apartado se mostrarán las clases diseñadas para la plataforma de juego. Esta aplicación seguirá el patrón Modelo-Vista-Controlador.

### 5.4.1. Diagrama de clases

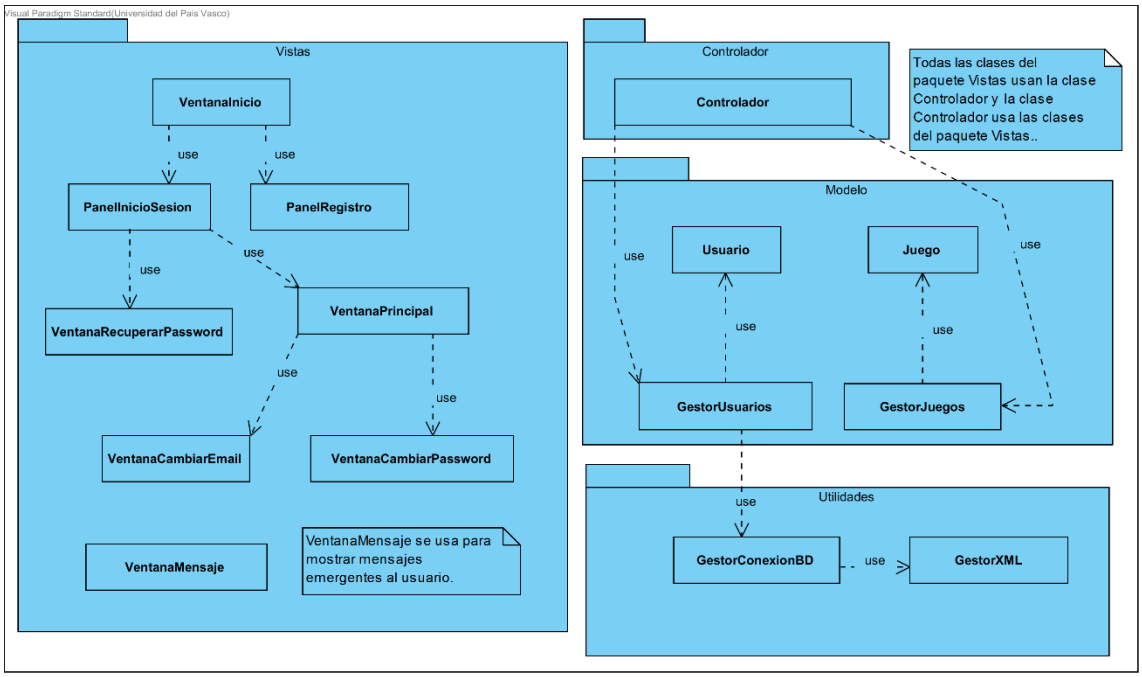



Ilustración 100: DIAGRAMA DE CLASES PLATAFORMA DE JUEGO

### 5.4.2. Clases

#### Usuario

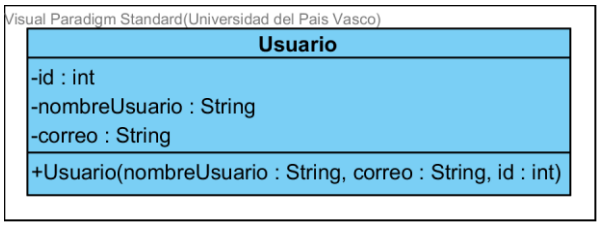



Ilustración 101: USUARIO

Esta clase se utiliza para almacenar la información del usuario que se conecta a la aplicación.

#### GestorUsuarios

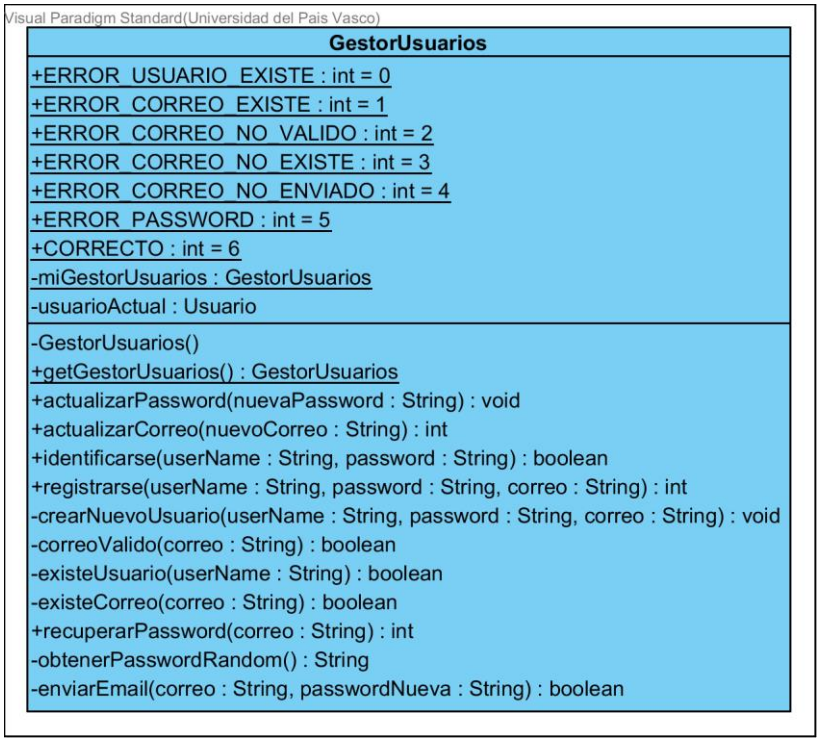



Ilustración 102: GESTOR USUARIOS

Esta clase se encarga de realizar la gestión del usuario conectado a la aplicación.

### Juego

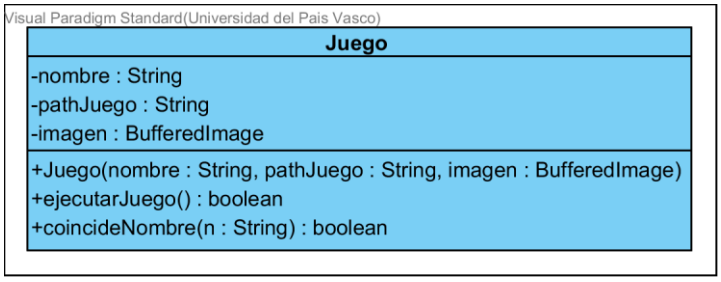



Ilustración 103: JUEGO

Esta clase se utiliza para almacenar la información de un juego y ejecutarlo.

### GestorJuegos

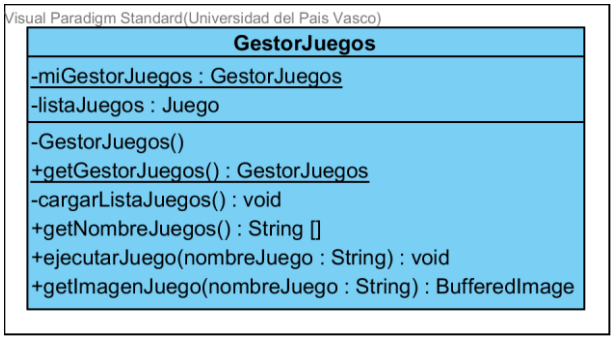



Ilustración 104: GESTOR JUEGOS

Esta clase se encarga de cargar el listado juegos, proporcionar el nombre e imagen asociados a un juego y ejecutar un juego de la lista.

## 5.5. Modelo relacional de bases de datos

En este apartado se muestra el modelo relacional de bases de datos. Se muestra en cada tabla la clave primaria mediante una llave y los tipos de datos de la información almacenada.

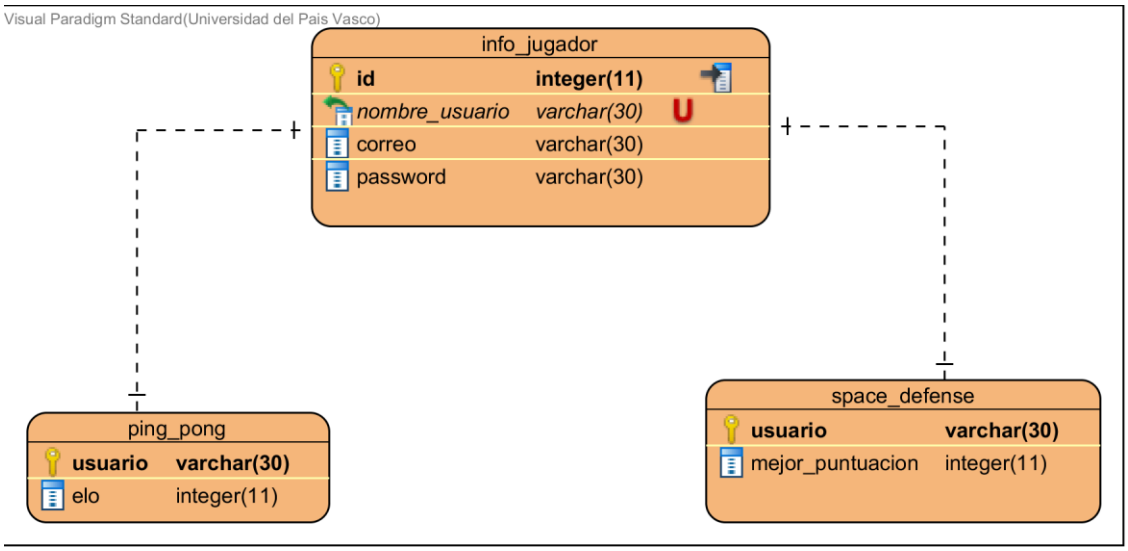



Ilustración 105: MODELO RELACIONAL DE BASES DE DATOS

Se ha utilizado como clave primaria la variable usuario en las tablas ping_pong y space_defense, y también como clave extranjera, haciendo referencia a la variable nombre_usuario presente en la tabla info_jugador.

# 6. DESARROLLO

En este apartado se va a explicar las partes más importantes del proceso de desarrollo de cada una de las aplicaciones implementadas. Para ello se ha utilizado el entorno de desarrollo Eclipse y los lenguajes de programación Java y XML. Para las bases de datos se ha usado MySQL Workbench para uso local.

## 6.1. Motor gráfico

El objetivo de esta parte es desarrollar una serie de módulos que faciliten la implementación de los juegos.

Algunos de estos módulos serán modificados en cada uno de los juegos que se van a desarrollar. Estas modificaciones se explicarán en el apartado de cada juego.

Los módulos son los siguientes:

- **Módulo de entrada datos.**
- **Módulo de estados.**
- **Módulo principal.**
- **Módulo de interfaz de usuario.**
- **Módulo de sonido.**
- **Módulo de utilidades.**

A continuación, se realizará una descripción detallada para cada uno de ellos.

### 6.1.1. Módulo de entrada datos

Este módulo contiene un par de clases diseñadas utilizando el patrón Singleton para que sean de acceso global en toda la aplicación. Estas clases se encargan de gestionar la entrada de datos del usuario. Se controlará la entrada de datos por los periféricos de entrada más comunes que son el teclado y el ratón. El objetivo es que se pueda solicitar si el usuario ha pulsado o está pulsando una tecla o algún botón del ratón.

Para ello se hará uso de las siguientes interfaces que provee Java y que deberá implementar la clase Canvas del módulo principal que se encarga de recoger estos eventos.

- **KeyListener:** informa cada vez que el usuario pulsa una tecla.
- **MouseListener:** informa cada vez que el usuario pulsa un botón del ratón.
- **MouseMotionListener:** informa de la posición actual del puntero del ratón cada vez que este se mueve.

Cada vez que un Listener informe se deberá informar a la clase del módulo de entrada de datos que registrará la entrada.

### 6.1.2. Módulo de estados

Para crear las diferentes vistas de los juegos se utilizará el modelo de diseño basado en estados formado por una máquina de estados y los diferentes estados posibles.

En este modelo cada estado que se cree representará una de las vistas de la aplicación. Estas clases deberán implementar la interfaz IState que proporciona cuatro métodos:

- **onSet:** se llamará cuando el estado se establezca.
- **onChange:** se llamará cuando el estado se cambie por otro.
- **update:** actualiza el estado.
- **render:** renderiza en pantalla los gráficos utilizando el objeto Graphics que se le pasa por parámetro. La clase Graphics permite dibujar en un Canvas tanto imágenes como formas geométricas personalizadas.

La máquina de estados actuará como gestor de los diferentes estados. Sus funciones son las siguientes:

- Actualizar y renderizar el estado actual.
- Cambiar entre los diferentes estados.
- Avisar a los estados cuando comiencen y finalicen.

### 6.1.3. Módulo principal

Este módulo será el encargado de ejecutar las aplicaciones y crear la vista principal mediante un JFrame.

Incluye también una clase Canvas que será la encargada de mostrar los gráficos en pantalla utilizando la clase Graphics que proporciona. Para el renderizado de los gráficos se usa el siguiente algoritmo:

```
BufferStrategy bs = this.getBufferStrategy();
if(bs == null){
    this.createBufferStrategy(3);
    return;
}
Graphics g = bs.getDrawGraphics();
Dimension d = ventana.getDimensionPantalla();
g.clearRect(0, 0, d.width,d.height);
GestorPrincipal.getGestorPrincipal().render(g);
g.dispose();
bs.show();
```

Este algoritmo se encarga de obtener el objeto Graphics del Canvas, limpiar la pantalla y por último renderizar nuevamente la pantalla utilizando el método render() del GestorPrincipal. Este método dibujará el estado actual de la máquina de estados en el Canvas.

Otro de los aspectos importantes de este módulo es el algoritmo conocido como Game Loop que se encarga de actualizar y renderizar constantemente el juego. Su código es el siguiente:

```
long lastTime = System.nanoTime();
double delta = 0;
long timer = System.currentTimeMillis();
long timer2 = System.currentTimeMillis();
int frames = 0;
int aps = 0;
int actualizacionesPorSegundo =1000000000/60;
int segundo = 1000;
boolean activo = true;
while(activo){
    long now = System.nanoTime();
    delta += (now - lastTime) / actualizacionesPorSegundo);
    lastTime = now;
    while(delta >= 1){
        updateGame();
        delta--;
        aps++;
    }
    if(activo){
        renderGame();
        frames++;
    }
    if(System.currentTimeMillis() - timer2 > segundo){
        timer2 +=segundo;
        System.out.println("APS: " + aps);
        aps = 0;
    }
    if(System.currentTimeMillis() - timer > segundo){
        timer +=segundo;
        System.out.println("FPS: " + frames);
        frames = 0;
    }
}
```

Dos de las variables a tener en cuenta a la hora de controlar el rendimiento de un juego son las APS y FPS, los cuales son proporcionadas por este algoritmo.

Las actualizaciones por segundo o APS indican cuantas veces se actualiza el juego en un segundo. Esta variable deberá mantenerse constante ya que de lo contrario el juego iría a diferentes velocidades. En este caso se ha estimado que se mantenga constante a 60 APS.

Los frames por segundo o FPS indican cuantas veces se renderiza la pantalla en un segundo. En este caso no se controla el valor de la variable si no que se permite el renderizado a la velocidad que la computadora donde se ejecute el juego pueda. Sin embargo, hay que tener en cuenta que el valor no disminuya por debajo de 30, momento en el cual el ojo humano puede llegar a percibir que el juego no está ejecutándose de manera fluida. Dado que los juegos a desarrollar son en 2D y los ordenadores de hoy en día son bastante potentes gráficamente, no se espera que los FPS caigan por debajo de ese umbral.

Estas dos variables se deberán tener en cuenta a la hora de realizar las pruebas de rendimiento en cada juego.

### 6.1.4. Módulo de interfaz de usuario

Este módulo incluye una serie de clases que permitirán crear las diferentes interfaces gráficas de los juegos. Se han identificado 4 componentes que deberían ser indispensables para crear las interfaces.

- **Texto:** permite mostrar texto en pantalla en una posición determinada. El constructor de la clase permite la personalización del color, tamaño y alineación. También se incluye la opción de hacer que el texto parpadeé para resaltarlo y llamar la atención del usuario en caso de que sea necesario.
- **Botón:** permite crear un botón interactivo en una posición determinada de la pantalla. Para que el usuario pueda distinguirlo del texto se ha implementado la funcionalidad Mouse Over, la cual hace que cambie de color al pasar el puntero del ratón por encima. Aparte el constructor de la clase permite personalizar el texto y tamaño del botón.
- **Icono sonido:** dado que se pretende incorporar sonido a las aplicaciones, es imprescindible tener un icono mediante el cual el usuario pueda interactuar para activar o desactivar el sonido.
- **Imagen:** permite cargar imágenes y mostrarlas en pantalla en una posición determinada.

Aparte se proporciona también una interfaz IUIListener la cual deberán implementar aquellas clases que usen el botón o el icono de sonido. Esta interfaz avisa cada vez que el usuario interactúa con estos objetos. Para saber cuál es el objeto donde se ha producido la interacción se debe asignar un identificador único a cada objeto distinto que se cree.

### 6.1.5. Módulo de sonido

La función de este módulo es proveer de sonido a las aplicaciones que se van a desarrollar. Para ello se hace uso de la librería javax.sound la cual permite la carga y reproducción de sonidos en formato .wav.

### 6.1.6. Módulo de utilidades

El objetivo de este módulo es proveer una serie de clases que sirvan de apoyo al resto de clases de la aplicación.

En primer lugar, se ha desarrollado una clase Singleton que permite conectarse a una base de datos y realizar consultas sobre ella.

En segundo lugar, se ha creado un gestor de ficheros en formato XML el cual permite:

- Leer valores de un fichero XML de dos niveles dado el nombre de la etiqueta de cada nivel.
- Crear ficheros XML de un nivel dado una lista de pares donde el primer valor representa el nombre de la etiqueta y el segundo el valor correspondiente.
- Leer valores del fichero XML creado por este gestor dado el nombre de la etiqueta.

La primera funcionalidad se utilizará para leer los ficheros de configuración de cada aplicación. Estos ficheros se utilizarán para inicializar múltiples variables desde un único fichero, lo que es bastante útil para personalizar la aplicación sin tener que moverse entre diferentes partes del código modificando las variables.

Esta funcionalidad va a tener gran importancia a la hora de configurar los juegos para lograr una buena experiencia de juego. De esta forma, a través del fichero de configuración se puede modificar rápidamente diferentes aspectos del juego hasta obtener la configuración más idónea. En los apartados de cada juego se explicará con más detalle como se ha utilizado este fichero.

Por último, se proporciona unas clases encargadas de realizar la gestión de los usuarios de las aplicaciones. La función de estas clases es asegurarse de que el usuario que está iniciando la aplicación es válido.

Para ejecutar cada juego se ha implementado un sistema de seguridad para que ningún usuario pueda ejecutar el juego directamente desde el fichero .jar. El único modo de acceso será a través de la plataforma. Para que se pueda ejecutar el juego se le deberá pasar como argumento el nombre usuario único y un identificador aleatorio único creado al registrarse que deberá obtenerse de la base de datos una vez el usuario ha iniciado sesión.

## 6.2. Space-Defense

En este apartado se va a mostrar el proceso de desarrollo del juego Space-Defense. Este proceso se ha dividido en diferentes partes que se explican a continuación.

### 6.2.1. Uso de módulos

En este apartado se explicará con detalle cómo se han utilizado los diferentes módulos del motor gráfico para desarrollar la aplicación.

- **Módulo de entrada datos:** se ha usado para detectar la pulsación de teclas por parte del usuario, así como la interacción mediante ratón.
- **Módulo de estados:** este módulo se encargará de gestionar las diferentes pantallas de la aplicación. Se han añadido algunas funciones nuevas a la clase StateMachine para gestionar el cambio de estados en casos especiales.
- **Módulo principal:** este módulo será el encargado de iniciar y gestionar el funcionamiento de la aplicación. Se han incorporado nuevas funcionalidades a algunas de las clases de este módulo para gestionar los cambios realizados en la clase StateMachine.
- **Módulo de sonido:** se ha utilizado para reproducir la música y los diferentes efectos de sonido del juego.
- **Módulo de interfaz de usuario:** se ha utilizado para crear los elementos de la interfaz de todas las pantallas del juego.
- **Módulo de utilidades:** este módulo se ha utilizado para realizar las conexiones a la base de datos, acceder al fichero de configuración y gestionar los usuarios. En la parte de gestión de usuarios se han incorporado nuevas funcionalidades para adaptarlo al juego y gestionar la puntuación de los usuarios.

### 6.2.2. Máquina estados

Para el diseño de las pantallas que componen el juego se ha utilizado el modelo basado en estados proporcionado en el motor gráfico. Se ha utilizado la interfaz IState para definir cada pantalla como un estado y la clase StateMachine para gestionar los cambios entre los múltiples estados. A continuación, se explicará con detalle cada estado y las condiciones de transición entre cada uno.

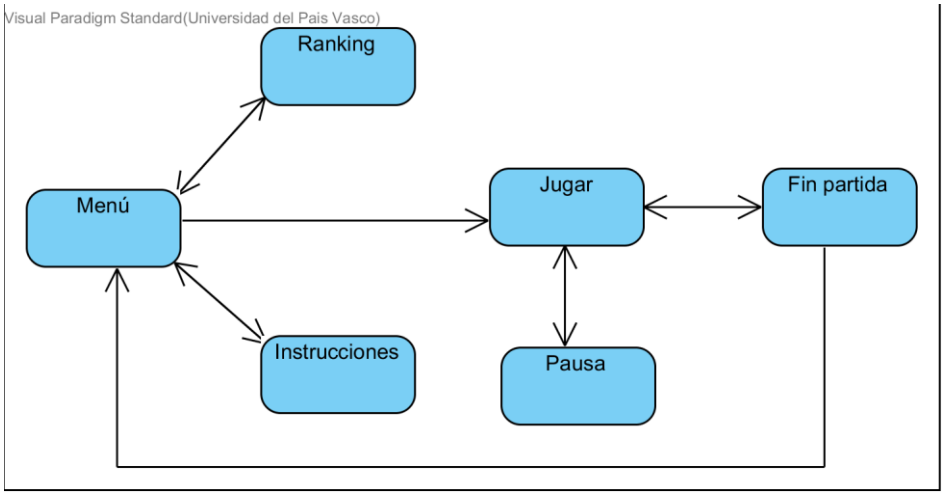



Ilustración 106: MÁQUINA ESTADOS SPACE-DEFENSE

- **Menú:** menú de inicio que vera el usuario al iniciar la aplicación. Dispone de tres botones mediante los cuales se accede a los estados Ranking, Instrucciones y Jugar.
- **Ranking:** pantalla que muestra la lista de los mejores jugadores con la mejor puntuación que han conseguido. Dispone de un botón para volver al estado Menú.
- **Instrucciones:** pantalla que muestra los controles y las reglas del juego. Dispone de un botón para volver al estado Menú.
- **Jugar:** estado en el que el usuario juega una partida. Si el usuario presiona la tecla P se pasará al estado Pausa. Si el usuario presiona la tecla Esc se dará por finalizada la partida y se pasará al estado Fin partida con la puntuación obtenida hasta el momento.
- **Pausa:** pantalla de pausa en la que el usuario puede estar el tiempo que desee. Si el usuario pulsa la tecla P se volverá al estado Jugar.
- **Fin partida:** pantalla en la cual se muestra al usuario la puntuación obtenida en la partida. Si es la mejor que ha obtenido el jugador hasta el momento se actualizará en la base de datos. Dispone de un botón para volver al estado Menú y otro para volver al estado Jugar.

### 6.2.3. Game Objects

Para el desarrollo de todos los componentes que forman la parte jugable de la aplicación se ha implementado la clase abstracta GameObject. De esta clase heredarán todos los objetos del juego y contiene las variables y funcionalidades que todos deben tener como puede ser la posición en pantalla, la velocidad de movimiento, el tamaño, el sistema de colisiones o la función para moverse.

En el sistema de colisiones se han utilizado cajas de colisión en forma rectangular que contendrán el Sprite (imagen) del objeto. Si se produce una intersección entre los dos rectángulos se detecta la colisión.

### 6.2.4. Fichero configuración

Con el objetivo de facilitar el testeo del juego se ha creado un fichero de configuración en formato XML. Este fichero será accesible a través de la clase GestorXML proporcionada en el módulo de utilidades del motor gráfico.

```
<config>
    <ventana>
        <ancho>500</ancho>
        <alto>700</alto>
        <titulo>Space</titulo>
    </ventana>
    <baseDatos>
        <url>jdbc:mysql://127.0.0.1:3306/xjmonasterio004_tfgbd</url>
        <driver>com.mysql.jdbc.Driver</driver>
        <usuario>root</usuario>
        <password>root</password>
    </baseDatos>
    <bonificacion>
        <velocidadMaxima>5</velocidadMaxima>
        <velocidadMinima>3</velocidadMinima>
        <size>30</size>
    </bonificacion>
    <bonificacionDisparoRapido>
        <duracion>8000</duracion>
        <tiempoRecarga>100</tiempoRecarga>
    </bonificacionDisparoRapido>
    <bonificacionEscudo>
        <duracion>6000</duracion>
    </bonificacionEscudo>
    <bonificacionTripleDisparo>
        <duracion>8000</duracion>
    </bonificacionTripleDisparo>
    <naveJugador>
        <posXInicial>200</posXInicial>
        <posYInicial>640</posYInicial>
        <ancho>30</ancho>
        <alto>50</alto>
        <velocidadInicial>5</velocidadInicial>
        <velocidadBala>10</velocidadBala>
        <altoBala>12</altoBala>
        <anchoBala>12</anchoBala>
        <tiempoEntreDisparosInicial>400</tiempoEntreDisparosInicial>
        <vidaMaxima>3</vidaMaxima>
    </naveJugador>
    <sistemaSpawn>
        <duracionOleada>30000</duracionOleada>
```

```
    </sistemaSpawn>
    <enemigoA>
        <maxSize>50</maxSize>
        <minSize>30</minSize>
        <maxVelocidad>1</maxVelocidad>
        <minVelocidad>0.5</minVelocidad>
        <maxTiempoRecarga>5000</maxTiempoRecarga>
        <minTiempoRecarga>2500</minTiempoRecarga>
        <velocidadBala>2.3</velocidadBala>
        <anchoBala>10</anchoBala>
        <altoBala>16</altoBala>
        <puntos>20</puntos>
        <porcentajeRecompensa>0.1</porcentajeRecompensa>
    </enemigoA>
    <enemigoB>
        <maxSize>50</maxSize>
        <minSize>22</minSize>
        <maxVelocidad>3.5</maxVelocidad>
        <minVelocidad>1.2</minVelocidad>
        <puntos>40</puntos>
        <porcentajeRecompensa>0.2</porcentajeRecompensa>

    </enemigoB>
    <enemigoC>
        <ancho>50</ancho>
        <alto>20</alto>
        <maxVelocidad>1.5</maxVelocidad>
        <minVelocidad>1.3</minVelocidad>
        <maxTiempoRecarga>1000</maxTiempoRecarga>
        <minTiempoRecarga>400</minTiempoRecarga>
        <velocidadBala>3</velocidadBala>
        <anchoBala>5</anchoBala>
        <altoBala>20</altoBala>
        <posYMaxima>300</posYMaxima>
        <puntos>80</puntos>
        <porcentajeRecompensa>0.4</porcentajeRecompensa>
    </enemigoC>
</config>
```

Se pueden ajustar las velocidades y tamaño del jugador, enemigos, balas y bonificaciones.

Modificando el porcentaje de recompensa de cada enemigo podremos aumentar o disminuir la probabilidad de que el jugador obtenga bonificaciones.

La variable duracionOleada indicará cuanto tiempo dura cada oleada del juego. Reduciendo el valor la dificultad aumentará y viceversa.

También se puede ajustar la duración de todas las bonificaciones temporales para poner más o menos complicado el juego al usuario.

## 6.3. Ping-Pong

En este apartado se va a mostrar el proceso de desarrollo de las aplicaciones cliente y servidor del juego Ping-Pong. Este proceso se ha dividido en diferentes partes que se explican a continuación.

### 6.3.1. Uso de módulos

En este apartado se explicará con detalle cómo se han utilizado los diferentes módulos del Motor Gráfico para desarrollar el juego.

- **Módulo de entrada datos:** se ha usado para detectar la pulsación de teclas por parte del usuario, así como la interacción mediante ratón.
- **Módulo de estados:** este módulo se encargará de gestionar las diferentes pantallas de la aplicación. Se han añadido algunas funciones nuevas a la clase StateMachine para gestionar el cambio de estados en casos especiales. A la interfaz IState se le ha añadido un método para recibir las respuestas por parte del servidor.
- **Módulo principal:** este módulo será el encargado de iniciar y gestionar el funcionamiento del juego. Se han incorporado nuevas funcionalidades a algunas de las clases de este módulo para gestionar los cambios realizados en la clase StateMachine y gestionar la conexión con el servidor.
- **Módulo de sonido:** se ha utilizado para reproducir la música y los diferentes efectos de sonido del juego.
- **Módulo de interfaz de usuario:** se ha utilizado para crear los elementos de la interfaz de todas las pantallas del juego.
- **Módulo de utilidades:** este módulo se ha utilizado para realizar las conexiones a la base de datos, acceder al fichero de configuración y gestionar los usuarios. En la parte de gestión de usuarios se han incorporado nuevas funcionalidades para adaptarlo al juego y gestionar el Elo de los jugadores.

### 6.3.2. Multithreading

El termino multithreading hace referencia a múltiples procesos (Thread) que se ejecutan de forma paralela al mismo tiempo.

La aplicación servidor que se va a desarrollar está formada por múltiples Thread que tendrán una determinada función asignada.

- **Thread servidor:** este proceso se encargará de establecer la comunicación con cada nuevo cliente que se conecta al servidor.

- **Thread cliente:** una vez que el cliente se ha conectado, se creará un nuevo proceso encargado de gestionar la comunicación entre el cliente y el servidor. Este proceso se mantendrá activo hasta que el cliente se desconecte o se le eche del servidor.
- **Thread sistema emparejamiento:** este proceso se encargará de crear las partidas entre los diferentes clientes conectados al servidor.
- **Thread partida:** cada partida que se cree se ejecutará dentro de un proceso distinto encargado de gestionar la partida.

Muchos de estos procesos acceden a las mismas variables para realizar una lectura o modificación. Por lo tanto, es necesario diseñar un sistema de control de concurrencia para bloquear el uso de las variables cuando están siendo utilizadas en otro proceso distinto al que intenta acceder a ellas.

Para solucionarlo se ha utilizado el siguiente trozo de código para cada variable que sea accesible en múltiples procesos.

```
synchronized (variable) {
    //Uso de la variable
}
```

### 6.3.3. Diseño comunicación Cliente-Servidor

En este apartado se van explicar las elecciones de diseño para realizar la comunicación entre la aplicación cliente y servidor (1).

#### 6.3.3.1. Topología

Para escoger la topología de red que tendrá la aplicación se han analizado dos tipos diferentes:

- **Cliente-Servidor:** en este modelo una aplicación servidor se encarga de proveer de recursos o servicios a las aplicaciones clientes que se conectan a él.

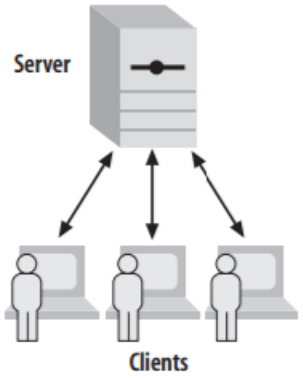



Ilustración 107: MODELO CLIENTE-SERVIDOR (1)

- **¡P2P:** en este modelo los propios clientes comparten sus recursos y servicios entre ellos.

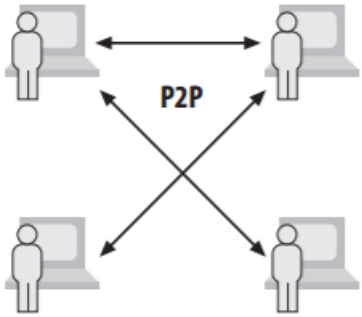



Ilustración 108: MODELO P2P (1)

En la siguiente tabla se muestran las ventanas y desventajas de cada modelo.

| | Ventajas | Desventajas |
|---|---|---|
| **Cliente-Servidor** | • Mayor seguridad<br>• Control centralizado | • Falla todo el sistema en caso de fallo del servidor.<br>• Mayor carga de trabajo cuantos más clientes haya, lo que provoca que vaya más lento. |
| **P2P** | • Intercambio de datos directo entre los clientes.<br>• Más rápido | • Menor seguridad<br>• No centralizado |

Tabla 36: CLIENTE-SERVIDOR VS P2P

Una de las soluciones analizadas ha sido implementar un servidor para el sistema de emparejamiento y el sistema P2P para la gestión de las partidas, de modo que uno de los dos clientes que juegan una partida se encargaría de realizar las tareas del servidor que son exclusivas de una partida. Sin embargo, esta solución tiene el inconveniente de que es no del es todo segura. ¿Qué pasaría si se hackea el cliente? De darse el caso el hacker tendría acceso a las variables que controlan la partida y podría modificarlas de tal manera que ganara la partida.

Por esta razón se ha escogido la opción Cliente-Servidor, ya que además de ser necesario una aplicación central encargada de realizar los emparejamientos entre los jugadores, no tiene el problema de la seguridad mencionado.

#### 6.3.3.2. Protocolo

Los protocoles de comunicación entre el cliente y el servidor que se han tenido en cuenta son los siguientes:

- **TCP:** permite que la conexión entre el cliente y el servidor se mantenga activa durante el tiempo de vida del juego. Además, se asegura que todos los mensajes enviados llegan a su destino. En definitiva, es un protocolo fiable, aunque más lento que el UDP.
- **UDP:** este protocolo destaca por la rapidez en comparación con TCP. Sin embargo, el principal inconveniente es que no se asegura la entrega de los mensajes por lo que no es adecuado para mensajes que se envían una única vez. Su principal uso en los juegos se da en el envío de datos de manera constante como puede ser la posición de un objeto en el juego que varía constantemente.

Una vez analizado las ventajas e inconvenientes de cada protocolo se ha decidido optar por la implementación del protocolo TCP debido principalmente a que se asegura el envío de cada mensaje. Esta característica permite que muchos de los datos que tenemos que enviar al servidor se envíen una única vez con la garantía de que van a llegar al destino.  Ejemplos de estos mensajes se mostrarán en el siguiente apartado.

#### 6.3.3.3. Transmisión de mensajes

En este apartado se va a explicar el sistema desarrollado para la transmisión de mensajes entre el cliente y el servidor.

El envío de mensajes se producirá mediante ficheros XML. Utilizando la clase GestorXML se crearán los ficheros con la información y con la misma clase se podrán leer los datos que contiene. Estos ficheros XML tendrán la estructura de un árbol de un único nivel (sin contar la raíz) donde ira la información en diferentes etiquetas.

Para facilitar la implementación de la aplicación se ha establecido un sistema en el que el cliente es el encargado de enviar los mensajes y el servidor se encarga de responder a estos.

En los mensajes enviados desde el cliente al servidor será necesario colocar la etiqueta <tipo></tipo>. Esta etiqueta ayudará al servidor a identificar el mensaje. Los valores posibles que pueden encontrarse dentro de esta etiqueta son:

- **obtenerDatos:** mediante estos mensajes se solicita la obtención de datos del servidor.
- **envioFuncion:** mediante estos mensajes se solicita una transición de estado. Debe ir acompañado de la etiqueta idFuncion que identifica la

transición a realizar. Un ejemplo de esto podría ser la solicitud de una pausa.

- **envioDatos:** los mensajes de este tipo van acompañados de múltiples etiquetas con datos que se envían al servidor. Un ejemplo podría ser el envío de la posición del jugador en el tablero.

En los mensajes enviados como respuesta desde el servidor al cliente será necesario colocar la etiqueta <estado></estado>. De esta forma la aplicación cliente podrá realizar las transiciones entre los diferentes estados.

Este sistema tiene el principal inconveniente de que en muchas ocasiones se puede estar solicitando datos desde el cliente que no se hayan modificado desde la última solicitud. Una de las posibles soluciones a este problema sería permitir que el servidor enviara estos mensajes de forma autónoma cuando se produzca un cambio. Sin embargo, para corregir este problema se ha optado por el envío de este tipo de mensajes de forma controlada desde el cliente. Por ejemplo, para la solicitud de datos en el estado de pausa, las solicitudes se envían cada segundo ya que los datos solicitados (tiempo restante) a este estado únicamente cambian cada segundo. De esta misma forma se ha ido determinando el tiempo entre solicitudes para cada uno de los estados.

### 6.3.4. Máquina estados

En este apartado se va a mostrar cómo se ha utilizado la máquina de estados en las aplicaciones cliente y servidor.

#### 6.3.4.1. Cliente

Para el diseño de las pantallas del cliente del se ha utilizado el modelo basado en estados proporcionado en el motor gráfico. Se ha utilizado la interfaz IState para definir cada pantalla como un estado y la clase StateMachine para gestionar los cambios entre los múltiples estados.

A continuación, se explicará con detalle cada estado y como se realizan las transiciones entre ellos.

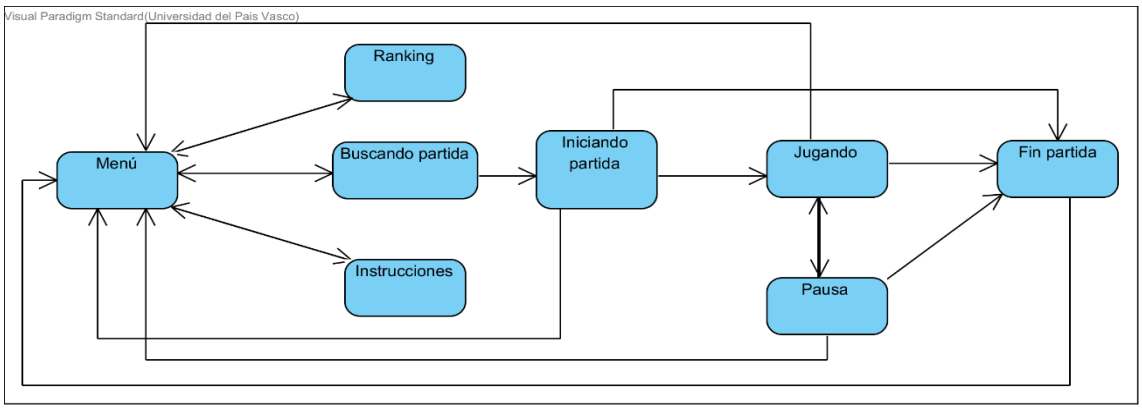



Ilustración 109: MÁQUINA ESTADOS CLIENTE PING-PONG

- **Menú:** menú de inicio que verá el usuario al iniciar la aplicación. Dispone de tres botones mediante los cuales se accede a los estados Ranking, Instrucciones y Buscando partida.
- **Ranking:** pantalla que muestra la lista de los mejores jugadores con su respectivo Elo y el Elo del propio usuario. Dispone de un botón para volver al estado Menú.
- **Instrucciones:** pantalla que muestra los controles y las reglas del juego. Dispone de un botón para volver al estado Menú.
- **Buscando partida:** el usuario se encontrará en este estado hasta que encuentre una partida, momento en el cual cambiará al estado Iniciando partida. Aparte dispone de un botón para volver al estado Menú en caso de que no se encuentre partida.
- **Iniciando partida:** en esta pantalla se realiza una cuenta atrás para comenzar la partida. Cuando la cuenta llegue a 0 se cambiará al estado Jugando. El usuario puede presionar la tecla Esc para volver al menú dándole por perdida la partida en cuyo caso el oponente pasaría al estado Fin partida.
- **Jugando:** estado en el que el usuario juega la partida contra un oponente. Si el usuario presiona la tecla P y no ha pausado previamente la partida se pasará al estado Pausa. Si el oponente pausa la partida también se cambiará al estado Pausa. Si el usuario presiona la tecla Esc vuelve al estado Menú dándole por perdida la partida en cuyo caso el oponente pasaría al estado Fin partida. Por ultimo si hay un ganador de la partida se pasará al estado Fin partida.
- **Pausa:** en esta pantalla se realiza una cuenta atrás hasta dar por finalizada la pausa, momento en el cual se cambia al estado Jugando. Si el usuario que ha solicitado la pausa pulsa la tecla P se volverá tras una corta cuenta atrás al estado Jugando. El usuario también puede presionar la tecla Esc para volver al estado Menú dándole por perdida la partida en cuyo caso el oponente pasaría al estado Fin partida.
- **Fin partida:** pantalla en la cual se muestra al usuario si ha ganado la partida y la causa de la victoria (victoria por puntos o por desconexión del oponente). Dispone de un botón para volver al estado Menú.

### 6.3.4.2. Servidor

La máquina de estados se ha utilizado en la aplicación servidor para gestionar los diferentes estados en los que se puede encontrar una partida en curso entre dos usuarios. Estos estados coinciden con parte de los estados de la máquina de estados de la aplicación cliente ya que están directamente relacionados entre sí.

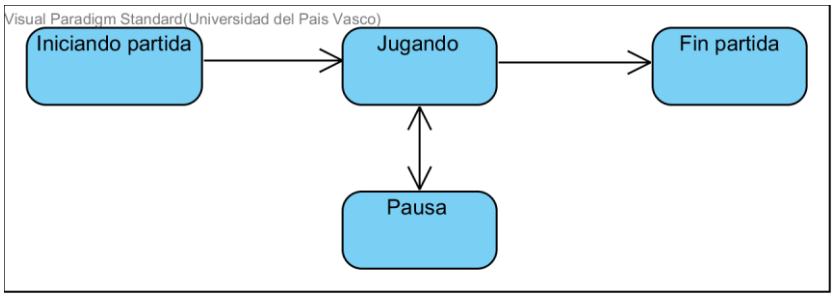



Ilustración 110: MÁQUINA ESTADOS SERVIDOR PING-PONG

La transición entre estos estados se llevará a cabo desde el servidor y tendrá repercusión en las aplicaciones de los clientes.

Una transición se puede dar o bien por qué se ha cumplido una cierta condición en la aplicación servidor (cuenta atrás finalizada, cliente desconectado) o por que el cliente ha enviado un mensaje solicitando una determinada acción como puede ser una pausa.

### 6.3.5. Sistema Elo

En este apartado se va a explicar el sistema creado para calcular el nivel de habilidad de cada usuario (Elo). Este sistema está basado en el sistema utilizado por los jugadores del ajedrez para determinar su habilidad.

El sistema tiene en cuenta la diferencia de Elo que existe entre los jugadores que se enfrentan. De modo que si un jugador con Elo muy superior a su oponente gana la partida, el ganador obtendrá muy poco Elo y el perdedor verá mínimamente reducido su Elo. De lo contrario, si el jugador con el Elo inferior gana, el ganador verá aumentado su Elo enormemente y el perdedor reducido en gran medida su Elo. Esto ocurre así debido a que el sistema interpreta que un jugador con más Elo debería ganar a uno con Elo inferior.

A continuación, se explica el proceso de cálculo del Elo (5) para cada jugador.

En primer lugar se van a obtener las variables R(1) y R(2) utilizando el Elo actual de cada jugador r(1) y r(2).

$$R(1) = 10^{r(1)/400}$$

$$R(2) = 10^{r(2)/400}$$

El segundo paso consiste en calcular el ganador esperado de la partida utilizando los valores calculados previamente.

$$E(1) = \frac{R(1)}{R(1) + R(2)}$$

$$E(2) = \frac{R(2)}{R(1) + R(2)}$$

Aquel jugador que obtenga un valor mayor se esperará que gane la partida y esto influirá a la hora de calcular el nuevo Elo para cada jugador. Por ejemplo, si un jugador obtiene E(1) = 0.7, el otro jugador obtendrá un valor E(2) = 0.3. Esto indica que se espera que de cada 10 partidas jugadas el primer jugador gane 7 y el segundo gane 3.

El tercer paso consiste en inicilizar las variables S(1) y S(2) para cada jugador. Estas variables tomaran el valor de 1 en caso de que el jugador correspondiente haya ganado o el valor 0 en caso de que haya perdido. En caso de un empate tomarían el valor 0.5, aunque para este juego este caso no es posible.

Finalmente se calcula el nuevo Elo de cada jugador usando los datos obtenidos previamente.

$$r'(1) = r(1) + k \times (S(1) - E(1))$$

$$r'(2) = r(2) + k \times (S(2) - E(2))$$

Para esta última fórmula se ha utilizado la variable K con un valor de 32. Cuanto mayor sea el valor de K mayor será la perdida de Elo para el perdedor. Del mismo modo el ganador obtendrá un aumento de Elo mayor. De esta forma, variando el valor de K se puede ajustar la ganancia y perdida de Elo.

### 6.3.6. Sistema emparejamiento

El sistema de emparejamiento es el encargado de crear las partidas entre los diferentes usuarios que se conectan al servidor. Dado que este sistema debe estar en constante funcionamiento se ha creado un nuevo Thread en el que se ejecutará el algoritmo de emparejamiento.

```
while(!fin){
    synchronized (listaClientes) {
        if(listaClientes.size() > 1){
            for(Cliente c : listaClientes){
                boolean emparejado = intentarEmparejar(c);
                if(emparejado){
                    break;
                }
```

```
                    }
                }
            }
            try {
                thread.sleep(sleepTime);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
```

Por cada vuelta del loop se está pausando el Thread para facilitar la ejecución del resto de procesos de la aplicación.

Para realizar el emparejamiento entre dos usuarios se van a tener en cuenta dos factores:

- **Elo:** nivel de habilidad de cada jugador.
- **Tiempo en cola:** tiempo que lleva el usuario buscando una partida.

En cada búsqueda del mejor candidato para el usuario se tendrá en cuenta en primer lugar el tiempo que lleva buscando. Si este tiempo supera un límite establecido se procederá a emparejar al usuario con el usuario con el Elo más cercano aun pudiendo haber una gran diferencia de habilidad entre los dos.

```
double eloCliente = c.getPerfilJugador().getElo();
Cliente mejorCandidato = null;
double mejorDiferencia = 10000000;
for(Cliente c2 : listaClientes){
    if(c != c2){
        double eloC2 = c2.getPerfilJugador().getElo();
        double diferencia = Math.abs(eloCliente -eloC2);
        if(diferencia < mejorDiferencia){
            mejorCandidato = c2;
            mejorDiferencia = diferencia;
        }
    }
}
if(mejorCandidato != null){
    GestorPrincipal gestorPrincipal = GestorPrincipal.getGestorPrincipal();
    gestorPrincipal.crearPartida(c, mejorCandidato);
    eliminarCliente(mejorCandidato);
    eliminarCliente(c);
    return true;
}else{
    return false;
}
```

En caso de que no se sobrepase el tiempo límite se procederá a emparejar al usuario con el usuario con el Elo más cercano dentro de un rango. De esta forma se empareja a usuarios con niveles de habilidad muy similares.

```
double eloCliente = c.getPerfilJugador().getElo();
Cliente mejorCandidato = null;
double mejorDiferencia = 10000000;
for(Cliente c2 : listaClientes){
    if(c != c2){
        double eloC2 = c2.getPerfilJugador().getElo();
        double diferencia = Math.abs(eloCliente -eloC2);
        if((diferencia <= diferenciaElo) && (diferencia < mejorDiferencia)){
            mejorCandidato = c2;
            mejorDiferencia = diferencia;
        }
    }
}
if(mejorCandidato != null){
    GestorPrincipal gestorPrincipal = GestorPrincipal.getGestorPrincipal();
    gestorPrincipal.crearPartida(c, mejorCandidato);
    eliminarCliente(mejorCandidato);
    eliminarCliente(c);
    return true;
}else{
    return false;
}
```

### 6.3.7. Fichero configuración

Con el objetivo de facilitar el testeo del juego se han creado ficheros de configuración en formato XML tanto en la aplicación cliente como en el servidor. Estos ficheros serán accesibles a través de la clase GestorXML proporcionada en el módulo de utilidades del motor gráfico.

Utilizando estos ficheros se podrá modificar fácilmente algunas de las variables del juego hasta crear una experiencia de juego con la que el jugador pueda disfrutar.

A continuación, se explica el fichero de configuración utilizado en cada aplicación.

#### 6.3.7.1. Cliente

Algunas de las variables que se podrán modificar en este fichero serán los valores de acceso a la base de datos y al servidor o el tamaño y la velocidad de la paleta del jugador.

```
<config>
    <ventana>
        <ancho>900</ancho>
        <alto>450</alto>
        <titulo>Ping-Pong</titulo>
    </ventana>
    <servidor>
        <ip>127.0.0.1</ip>
        <puerto>9090</puerto>
    </servidor>
    <baseDatos>
        <url>jdbc:mysql://127.0.0.1:3306/xjmonasterio004_tfgbd</url>
        <driver>com.mysql.jdbc.Driver</driver>
        <usuario>root</usuario>
        <password>root</password>
    </baseDatos>
    <tablero>
        <limiteArriba>0</limiteArriba>
        <limiteAbajo>450</limiteAbajo>
    </tablero>
    <jugador>
        <alto>90</alto>
        <ancho>25</ancho>
        <velocidad>8</velocidad>
        <posXDerecha>870</posXDerecha>
        <posXIzquierda>30</posXIzquierda>
        <posYInicial>225</posYInicial>
    </jugador>
    <bola>
        <radio>12</radio>
    </bola>
</config>
```

#### 6.3.7.2. Servidor

En este fichero se podrá configurar el funcionamiento del sistema de emparejamiento, la puntuación máxima para determinar el ganador de la partida o los tiempos de pausa permitidos.

También hay que tener en cuenta que algunas variables como el tamaño de la paleta del jugador o el tablero están directamente relacionadas con las variables del fichero del cliente. Por esta razón hay que asegurarse de que el valor sea el mismo en ambos ficheros.

```
<config>
    <servidor>
        <puerto>9090</puerto>
    </servidor>
    <bola>
        <velocidadInicial>600</velocidadInicial>
        <velocidadMaxima>1800</velocidadMaxima>
        <aumentoVelocidad>20</aumentoVelocidad>
        <radio>12</radio>
        <actualizacionesSegundo>60</actualizacionesSegundo>
    </bola>
    <partida>
        <actualizacionesSegundo>60</actualizacionesSegundo>
        <kValor>32</kValor>
    </partida>
    <tablero>
        <limiteArriba>0</limiteArriba>
        <limiteAbajo>450</limiteAbajo>
        <limiteDerecha>900</limiteDerecha>
        <limiteIzquierda>0</limiteIzquierda>
    </tablero>
    <jugador>
        <alto>90</alto>
        <ancho>25</ancho>
        <posXDerecha>870</posXDerecha>
        <posXIzquierda>30</posXIzquierda>
        <posYInicial>225</posYInicial>
    </jugador>
    <estadoPausa>
        <tiempoMaximoPausa>60000</tiempoMaximoPausa>

    <tiempoMaximoReanudacion>3000</tiempoMaximoReanudacion>
    </estadoPausa>
    <estadoIniciandoPartida>
        <tiempoMaximoInicio>5000</tiempoMaximoInicio>
    </estadoIniciandoPartida>
    <marcador>
        <puntuacionMaxima>15</puntuacionMaxima>
    </marcador>
    <baseDatos>
        <url>jdbc:mysql://127.0.0.1:3306/xjmonasterio004_tfgbd</url>
        <driver>com.mysql.jdbc.Driver</driver>
        <usuario>root</usuario>
        <password>root</password>
    </baseDatos>
    <sistemaEmparejamiento>
        <diferenciaElo>200</diferenciaElo>
```

```
            <tiempoMaximoEnCola>20000</tiempoMaximoEnCola>
            <sleepTime>500</sleepTime>
        </sistemaEmparejamiento>
    </config>
```

## 6.4. Plataforma de juego

A continuación, se explicará cómo se han implementado las partes más importantes de esta aplicación.

### 6.4.1. Diseño

Se ha seguido el patrón de diseño Modelo-Vista-Controlador.

Para crear las diferentes pantallas que componen la aplicación se ha utilizado la librería java.swing con todos los componentes que ofrece.

Para aquellas vistas donde se requiera que los datos introducidos en los formularios se mantengan al cambiar de pantalla se ha utilizado el patrón Singleton en las clases. De esta forma conseguimos una instancia única de la vista que no se destruye por lo que los datos se mantienen.

### 6.4.2. Envío de correo

En este apartado se va explicar cómo se ha implementado la funcionalidad de envío de correos al usuario para recuperar la contraseña utilizando la librería javax.mail (4).

Para que el usuario pueda recuperar la contraseña, este deberá introducir el correo con el que se registró, el cual será siempre válido ya que se realiza la comprobación en el registro con el siguiente algoritmo:

```
boolean valido = true;
InternetAddress emailAddr;
try {
    emailAddr = new InternetAddress(correo);
    emailAddr.validate();
} catch (AddressException e) {
    valido = false;
}
return valido;
```

Una vez introducido el correo, si es el mismo con el que se registró se creará una nueva contraseña para el usuario de dicho correo (el correo es único para cada usuario) y se actualizará en la base de datos.

La contraseña se ha creado usando el siguiente algoritmo:

```
String valoresPosibles =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ012345
6789";
int longitud = valoresPosibles.length();
StringBuilder sb = new StringBuilder(8);
Random r = new Random();
for (int i = 0; i < 8; i++) {
      sb.append(valoresPosibles.charAt(r.nextInt(longitud)));
}
return sb.toString();
```

Como se ve en el algoritmo, se genera un String alfanumérico de 8 caracteres donde cada carácter es obtenido de forma totalmente aleatoria.

Por último, se procederá al envío del correo al usuario. Para esta parte se ha tenido que crear una cuenta de usuario en Gmail para poder realizar el envío de mensajes.

El algoritmo utilizado es el siguiente:

```
String usuario = "recuperarpass1234";
String password = "7zmWxhNZ";
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.socketFactory.port", "465");
props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.port", "465");
Session session = Session.getDefaultInstance(props,
      new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                  return new PasswordAuthentication(usuario,password);
}});
try {
      Message message = new MimeMessage(session);
      message.setFrom(new InternetAddress(usuario));
      message.setRecipients(Message.RecipientType.TO,
      InternetAddress.parse(correo));
      message.setSubject("Recover password");
      message.setText("Your new password is: "+passwordNueva);
      Transport.send(message);
} catch (MessagingException e) {
      e.printStackTrace();
}
```

### 6.4.3. Carga de juegos

En este apartado se va explicar cómo se cargan los juegos en la aplicación para que el usuario pueda acceder a ellos.

En primer lugar, es necesario explicar la forma en la que debe estar organizada la carpeta donde se encuentra la plataforma de juego para que esta carga sea posible.

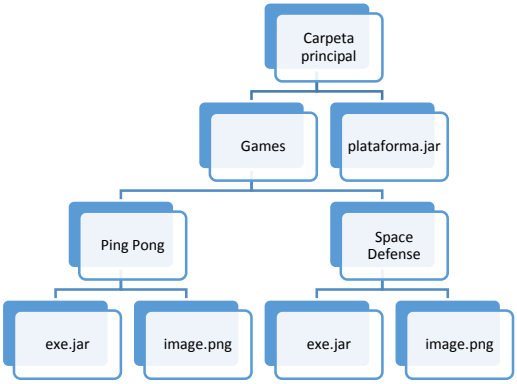



Ilustración 111: ESTRUCTURA CARPETA JUEGOS

Como se ve en la imagen, la carpeta donde está ubicado el ejecutable de la plataforma deberá contener una carpeta llamada “Games”. Esta carpeta contendrá a su vez múltiples carpetas donde se ubican cada uno de los juegos. El nombre que se le dé a estas carpetas deberá coincidir con el nombre del juego ya que se usará en la aplicación para mostrárselo al usuario.

Por último, las carpetas de los juegos deberán contener el ejecutable del juego llamado “exe” en formato .jar y una imagen relacionada con el juego llamada “image” en formato .png.

El algoritmo para cargar los juegos en la aplicación es el siguiente:

```
listaJuegos = new ArrayList<Juego>();
try {
    File f1 = new File(System.getProperty("java.class.path"));
    File f2 = f1.getAbsoluteFile().getParentFile();
    String direccionJuegos = f2.getAbsolutePath()+"/Games";
    File ficheroJuegos = new File(direccionJuegos);
    for (File carpeta : ficheroJuegos.listFiles()) {
        String nombreJuego = carpeta.getName();
        String pathJuego = "";
        BufferedImage imagen = null;
        for (File f : carpeta.listFiles()) {
            String n = f.getName();
            if(n.contentEquals("exe.jar")){
                pathJuego = f.getAbsolutePath();
```

```
                    }else if(n.contentEquals("image.png")){
                        imagen = ImageIO.read(f);
                    }
                }
                Juego j = new Juego(nombreJuego, pathJuego, imagen);
                listaJuegos.add(j);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

### 6.4.4. Ejecución juegos

Para le ejecución de los juegos se utilizará la dirección absoluta del juego obtenida en la carga del listado de juegos para ejecutar mediante código el comando de ejecución de ficheros .jar. También se le deberá pasar como argumento el nombre de usuario el usuario que ha iniciado sesión en la aplicación y su identificador único obtenido de la base de datos.

```
GestorUsuarios gestorUsuarios =
GestorUsuarios.getGestorUsuarios();
Usuario u = gestorUsuarios.getUsuarioActual();
int id = u.getId();
String nombre = u.getNombreUsuario();
String comando = "java -jar \""+ pathJuego + "\" "
            + id + " \"" + nombre+"\"";
try {
    Runtime.getRuntime().exec(comando);
} catch (IOException e) {
    e.printStackTrace();
}
```

En un principio este código no funcionaba correctamente y se tuvo que modificar. El error se producía cuando la dirección absoluta o el nombre de usuario contenía espacios por lo que el comando los reconocía como múltiples argumentos separados por espacios. Para solucionarlo se han puesto los argumentos que daban problemas entre comillas de forma que el intérprete de comandos los interprete como un único argumento.

# 7. VERIFICACIÓN Y EVALUACIÓN

En este apartado se va a mostrar el plan de pruebas diseñado para cada una de las aplicaciones y los resultados obtenidos, así como las correcciones que se hayan llevado a cabo para solucionar los problemas encontrados.

Se distinguen dos tipos de pruebas diferentes:

- **Pruebas de rendimiento:** este tipo de pruebas se llevarán a cabo en los juegos implementados y se encargarán de asegurar que el juego es capaz de ejecutarse correctamente en la computadora. Aunque se deberían llevar a cabo en diferentes ordenadores con diferente hardware, no va a ser posible dado que no se dispone de tal material. Sin embargo, como los juegos a desarrollar son bastante sencillos gráficamente y se cuenta con un ordenador moderno se podría decir que si los juegos superan las pruebas serían capaces de funcionar correctamente en cualquier ordenador moderno.

- **Pruebas de funcionalidad:** estas pruebas se encargarán de asegurar que las distintas funcionalidades de la aplicación funcionan correctamente. Para ello se ha diseñado la siguiente tabla:

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| | | | |

Tabla 37: PRUEBAS

  - **Código:** identificador de la prueba.
  - **Descripción:** breve descripción de en qué consiste la prueba.
  - **Resultado esperado:** resultado que se espera obtener.
  - **Resultado:** se indicará si el resultado es correcto o no.

## 7.1. Space-Defense

### 7.1.1. Pruebas de rendimiento

En esta parte se va a analizar el correcto funcionamiento del juego a través de una serie de variables.

- **FPS (frames por segundo):** esta variable indica el número de veces que la pantalla se renderiza en un segundo. Para que el ojo humano no note que la pantalla se está refrescando continuamente se deberá obtener un valor superior a 30. De otro modo se puede percibir los cambios entre las diferentes imágenes.
- **APS (actualizaciones por segundo):** esta variable indica el número de veces que el juego se actualiza en un segundo. Esta variable requiere que su valor sea constante e igual a 60. De lo contrario se

podría percibir que el juego aumenta o disminuye de velocidad constantemente, siempre y cuando la variable varíe considerablemente.

| Variable | Valor requerido | Valor obtenido | Observaciones |
|---|---|---|---|
| **FPS** | >=30 | ~10000 | Correcto |
| **APS** | =60 | ~60 | Correcto |

Tabla 38: PRUEBAS RENDIMIENTO SPACE-DEFENSE

Como se muestra en la tabla se obtienen los valores deseados en ambas variables.

### 7.1.2. Pruebas de funcionalidad

En primer lugar, se van a mostrar las pruebas relacionadas con la navegación entre las diferentes pantallas y las opciones que cada una ofrece.

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.1** | Usuario hace clic en Play. | Se inicia una nueva partida. | Correcto |
| **1.2** | Usuario hace clic en Ranking. | Se muestra la pantalla con el ranking. | Correcto |
| **1.3** | Usuario hace clic en Instructions. | Se muestra una pantalla con las instrucciones para jugar. | Correcto |
| **1.4** | Usuario hace clic en Exit. | Se cierra la aplicación. | Correcto |
| **1.5** | El usuario hace clic en el icono de sonido en modo ON. | Se desactiva el sonido y se cambia el icono por el de modo OFF. | Correcto |
| **1.6** | El usuario hace clic en el icono de sonido en modo OFF. | Se activa el sonido y se cambia el icono por el de modo ON. | Correcto |
| **1.7** | El usuario se encuentra en la pantalla de Game Over y hace clic en Play Again. | Se inicia una nueva partida. | Correcto |
| **1.8** | El usuario se encuentra en la pantalla de Game Over y hace clic en Main Menu. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.9** | El usuario se encuentra en la pantalla de ranking y hace clic en Back. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |
| **1.10** | El usuario se encuentra en la pantalla de instrucciones y hace clic en Back. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |

Tabla 39: PRUEBAS FUNCIONALIDAD NAVEGACION SPACE-DEFENSE

En la siguiente tabla se van a mostrar las pruebas relacionadas con la parte jugable de la aplicación.

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **2.1** | El usuario presiona clic izquierdo sobre la pantalla y ha transcurrido el tiempo de recarga. | Se dispara una bala hacia esa dirección. | Correcto |
| **2.2** | El usuario presiona clic izquierdo sobre la pantalla y no ha transcurrido el tiempo de recarga. | No ocurre nada. | Correcto |
| **2.3** | El usuario presiona la tecla D. | La nave se mueve hacia la derecha. | Correcto |
| **2.4** | El usuario presiona la tecla A. | La nave se mueve hacia la izquierda | Correcto |
| **2.5** | El usuario presiona la tecla D. El jugador se encuentra en el límite derecho de la pantalla. | La nave no se mueve. | Correcto |
| **2.6** | El usuario presiona la tecla A. El jugador se encuentra en el límite izquierdo de la pantalla. | La nave no se mueve | Correcto |
| **2.7** | El usuario  pulsa la tecla P. | Se muestra la pantalla de pausa. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **2.8** | El usuario pulsa la tecla P. El usuario se encuentra en la pantalla de pausa. | Se reanuda la partida como estaba previamente a la pausa. | Correcto |
| **2.9** | El usuario pulsa la tecla Esc. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza en base de datos la puntuación si fuera la mejor. | Correcto |
| **2.10** | La bala del jugador colisiona con un enemigo. | El enemigo muere y se añade la puntuación correspondiente al jugador. | Correcto |
| **2.11** | Una bala de un enemigo colisiona con el jugador. El jugador tiene más de 1 vida. | El jugador pierde una vida. | Correcto |
| **2.12** | Una bala de un enemigo colisiona con el jugador. El jugador tiene 1 vida. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza en base de datos la puntuación si fuera la mejor. | Correcto |
| **2.13** | Un enemigo colisiona con el jugador. El jugador tiene más de 1 vida. | El jugador pierde una vida. El enemigo muere y se añade la puntuación correspondiente al jugador. | Correcto |
| **2.14** | Un enemigo colisiona con el jugador. El jugador tiene 1 vida. | El enemigo muere y se añade la puntuación correspondiente al jugador. Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza en base de datos la puntuación si fuera la mejor. | Correcto |
| **2.15** | El jugador recoge una bonificación de tipo vida. El jugador no tiene el máximo de vidas. | Se suma una vida al jugador. Desaparece la bonificación de la pantalla. | Correcto |
| **2.16** | El jugador recoge una bonificación de tipo vida. El jugador | Desaparece la bonificación de la pantalla. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| | tiene el máximo de vidas. | | |
| **2.17** | El jugador recoge una bonificación de tipo disparo rápido. | El jugador ve incrementado durante un breve periodo de tiempo el número de disparos por segundo.<br><br>Desaparece la bonificación de la pantalla. | Correcto |
| **2.18** | El jugador recoge una bonificación de tipo triple disparo. | El jugador dispara tres bolas en lugar de una durante un breve periodo de tiempo.<br><br>Desaparece la bonificación de la pantalla. | Correcto |
| **2.19** | El jugador recoge una bonificación de tipo escudo. | El jugador no puede perder vidas durante un breve periodo de tiempo.<br><br>Desaparece la bonificación de la pantalla. | Correcto |

Tabla 40: PRUEBAS FUNCIONALIDAD JUGABILIDAD SPACE-DEFENSE

## 7.2. Ping-Pong

### 7.2.1. Pruebas de rendimiento

En esta parte se va a analizar el correcto funcionamiento del juego a través de una serie de variables.

- **FPS (frames por segundo):** esta variable indica el número de veces que la pantalla se renderiza en un segundo. Para que el ojo humano no note que la pantalla se está refrescando continuamente se deberá obtener un valor superior a 30. De otro modo se puede percibir los cambios entre las diferentes imágenes.
- **APS (actualizaciones por segundo):** esta variable indica el número de veces que el juego se actualiza en un segundo. Esta variable requiere que su valor sea constante e igual a 60. De lo contrario se podría percibir que el juego aumenta o disminuye de velocidad constantemente, siempre y cuando la variable varíe considerablemente.
- **Latencia:** esta variable indica el tiempo que transcurre en milisegundos desde que el cliente envía un mensaje al servidor y le llega la respuesta del servidor. Para que se pueda tener una buena experiencia de juego

en un entorno multijugador este valor deberá ser menor a 100 milisegundos.

| Variable | Valor requerido | Valor obtenido | Observaciones |
|---|---|---|---|
| **FPS** | >=30 | ~8000 | Correcto |
| **APS** | =60 | ~60 | Correcto |
| **Latencia** | <100 | ~1 | Correcto |

Tabla 41: PRUEBAS RENDIMIENGO PING-PONG

Como se muestra en la tabla se obtienen los valores deseados en todas las variables.

El valor de la latencia es muy bajo debido a que las pruebas se han realizado en un entorno local, por lo que los mensajes no tienen que viajar a través de la red. En caso de que el cliente y el servidor se encontrasen en diferentes puntos esta variable dependería de la velocidad de la conexión, la distancia entre el cliente y servidor y el protocolo de comunicación implementando.

### 7.2.2. Pruebas de funcionalidad

En primer lugar, se van a mostrar las pruebas relacionadas con la navegación entre las diferentes pantallas y las opciones que cada una ofrece.

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.1** | El usuario hace clic en Play. El usuario tiene conexión a internet. | Se inicia una nueva partida | Correcto |
| **1.2** | El usuario hace clic en Play. El usuario no tiene conexión a internet. | Se muestra un mensaje de error. | Correcto |
| **1.3** | Usuario hace clic en Ranking. | Se muestra la pantalla con el ranking. | Correcto |
| **1.4** | Usuario hace clic en Instructions. | Se muestra una pantalla con las instrucciones para jugar. | Correcto |
| **1.5** | Usuario hace clic en Exit. | Se cierra la aplicación. | Correcto |
| **1.6** | El usuario hace clic en el icono de sonido en modo ON. | Se desactiva el sonido y se cambia el icono por el de modo OFF. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.7** | El usuario hace clic en el icono de sonido en modo OFF. | Se activa el sonido y se cambia el icono por el de modo ON. | Correcto |
| **1.8** | El usuario se encuentra en la pantalla de Game Over y hace clic en Main Menu. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |
| **1.9** | El usuario se encuentra en la pantalla de ranking y hace clic en Back. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |
| **1.10** | El usuario se encuentra en la pantalla de instrucciones y hace clic en Back. | Se redirecciona al usuario a la pantalla de inicio. | Correcto |

Tabla 42: PRUEBAS FUNCIONALIDAD NAVEGACIÓN PING-PONG

En la siguiente tabla se van a mostrar las pruebas relacionadas con la parte jugable de la aplicación.

La situación en la que se van a realizar las pruebas es la de un usuario al que se le denominará como jugador, se encuentra jugando una partida contra otro usuario al que se le llamará oponente. Se ha supuesto en todos los casos que el servidor se encuentra funcionando correctamente.

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **2.1** | El usuario presiona la tecla W. | El jugador se mueve hacia arriba. | Correcto |
| **2.2** | El usuario presiona la tecla S. | El jugador se mueve hacia abajo. | Correcto |
| **2.3** | El usuario presiona la tecla W. El jugador se encuentra en el límite superior de la pantalla. | El jugador no se mueve. | Correcto |
| **2.4** | El usuario presiona la tecla S. El jugador se encuentra en el límite inferior de la pantalla. | El jugador no se mueve. | Correcto |
| **2.5** | El usuario se encuentra jugando y pulsa la tecla P. El usuario todavía no ha pausado la partida. | Se muestra la pantalla de pausa. | Correcto |
| **2.6** | El usuario se encuentra jugando y pulsa la tecla | No ocurre nada. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| | P. El usuaria ya ha pausado previamente la partida. | | |
| **2.7** | El usuario se encuentra en la pantalla pausa y causó la pausa de la partida. | Comienza la reanudación de la partida. | Correcto |
| **2.8** | El usuario se encuentra en la pantalla pausa y no causó la pausa de la partida. | No ocurre nada. | Correcto |
| **2.9** | El usuario pulsa la tecla Esc. | Se sale de la partida y se vuelve al menú principal. Se actualiza el Elo del jugador y del oponente. | Correcto |
| **2.10** | La bola colisiona con la paleta del jugador. | La bola rebota hacia el campo del oponente. | Fallo |
| **2.11** | La bola colisiona con la paleta del oponente. | La bola rebota hacia el campo del jugador. | Fallo |
| **2.12** | La bola se sale por el lado del jugador. | Se suma un punto al oponente. El jugador saca la bola al campo del oponente. | Correcto |
| **2.13** | La bola se sale por el lado del oponente. | Se suma un punto al jugador. El oponente saca la bola al campo del jugador. | Correcto |
| **2.14** | La bola colisiona con el límite superior del tablero. | La bola rebota hacia abajo. | Fallo |
| **2.15** | La bola colisiona con el límite inferior del tablero. | La bola rebota hacia arriba. | Fallo |
| **2.16** | El jugador llega a 15 puntos. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza el Elo del jugador y del oponente. | Correcto |
| **2.17** | El oponente llega a 15 puntos. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| | | el Elo del jugador y del oponente. | |
| **2.18** | El oponente pierde la conexión a internet. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza el Elo del jugador y del oponente. | Correcto |
| **2.19** | El oponente sale de la partida o cierra la aplicación. | Se finaliza la partida y se muestra la pantalla de Game Over. Se actualiza el Elo del jugador y del oponente. | Correcto |

Tabla 43: PRUEBAS FUNCIONALIDAD JUGABILIDAD PING-PONG

En este parte se han obtenido varios fallos relacionados con la colisión de la bola tanto contra el tablero como las paletas. Esto es debido a que una misma colisión se detecta múltiples veces lo que provoca que la bola no se mueva del punto de colisión.

Para solucionarlo se ha modificado el sistema de colisiones de forma que cuando la bola detecte una colisión, no vuelva a detectar colisiones hasta que esté fuera del rango de colisión del objeto contra el que colisionó.

## 7.3. Plataforma de juego

### 7.3.1. Pruebas de funcionalidad

En la siguiente tabla se muestra el plan de pruebas diseñado para las funcionalidades que proporciona la aplicación.

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.1** | El usuario inicia sesión con un nombre usuario correcto y una contraseña correcta. | Se muestra la ventana inicial de la plataforma de juego. | Correcto |
| **1.2** | El usuario inicia sesión con un nombre usuario correcto y una contraseña incorrecta. | Se muestra un mensaje avisándole del error. | Correcto |
| **1.3** | El usuario inicia sesión con un nombre usuario y una contraseña incorrecta. | Se muestra un mensaje avisándole del error. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.4** | El usuario intenta recuperar la contraseña e introduce un correo que no es válido. | Se muestra un mensaje avisándole del error. | Correcto |
| **1.5** | El usuario intenta recuperar la contraseña e introduce un correo válido. | Se muestra un mensaje avisándole de que se le ha enviado un correo con la nueva contraseña. | Correcto |
| **1.6** | El usuario se registra con un nombre usuario que ya existe. | Se muestra un mensaje avisándole del error. | Correcto |
| **1.7** | El usuario se registra con un correo ya utilizado. | Se muestra un mensaje avisándole del error. | Correcto |
| **1.8** | El usuario se registra con un nombre usuario y correo correcto pero las contraseñas de ambos campos no coinciden. | Se muestra un mensaje avisándole del error. | Correcto |
| **1.9** | El usuario se registra con nombre usuario y correo correcto y las contraseñas de ambos coinciden. | Se muestra un mensaje de éxito y se redirecciona a la ventana inicial de la plataforma de juego. | Correcto |
| **1.10** | El usuario selecciona un juego de la lista y pulsa en Play. | Se inicia el juego seleccionado. | Fallo |
| **1.11** | El usuario pulsa en cambiar contraseña pero ambas contraseñas no coinciden. | Se muestra un mensaje de error. | Correcto |
| **1.12** | El usuario pulsa en cambiar contraseña y ambas contraseñas coinciden. | Se cambia la contraseña de la cuenta del usuario. | Correcto |
| **1.13** | El usuario pulsa en cambiar correo y el correo ya está siendo utilizado por otro usuario. | Se muestra un mensaje de error. | Correcto |

| Código | Descripción | Resultado esperado | Resultado |
|---|---|---|---|
| **1.14** | El usuario pulsa en cambiar correo y el correo es válido. | Se cambia el correo de la cuenta del usuario. | Correcto |

Tabla 44: PRUEBAS FUNCIONALIDAD PLATAFORMA DE JUEGO

En esta parte se ha obtenido un error a la hora de iniciar los juegos que provoca que los juegos no se ejecuten. Esto es debido a que el comando utilizado para ejecutar los juegos interpreta mal los argumentos.

Por ejemplo, en el comando siguiente, se interpreta la dirección como dos argumentos totalmente diferentes ya que contiene espacios.

*Java –jar C:\\Carpeta Juegos\exe.jar*

Para solucionarlo se han puesto los argumentos del comando entre comillas de forma que se interprete cada uno como un único argumento.

# 8. CONCLUSIONES Y TRABAJO FUTURO

## 8.1. Cumplimiento de objetivos y plazos

Los objetivos establecidos al inicio del proyecto se han cumplido en su totalidad. Como se mencionó en el apartado de objetivos, se ha desarrollado una plataforma de juego a través de la cual se tiene acceso a los juegos Space-Defense y Ping-Pong, los cuales han sido implementados utilizando un motor gráfico propio.

En cuanto a los tiempos estimados en la planificación para la realización de las tareas se han cumplido en la mayoría de los casos. En el siguiente gráfico se muestra la comparativa entre las horas estimadas y las reales para cada una de las fases del proyecto.

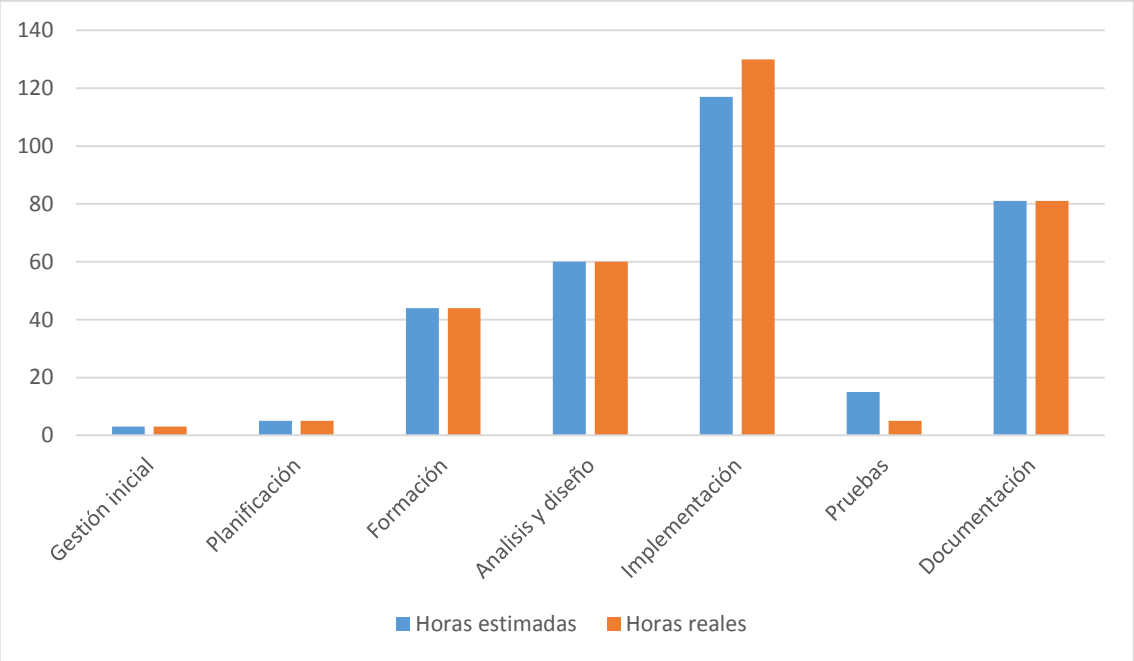



Ilustración 112: COMPARATIVA PLANIFICACIÓN

Como se ve en el gráfico, únicamente se han necesitado más horas de las estimadas en la parte de implementación, ya que no se tenía experiencia en el desarrollo de juegos multijugador online por lo que ha llevado más tiempo.

Por otra parte, debido a una buena fase de implementación, la fase de pruebas ha sido mucho más breve de lo estimado ya que apenas se han encontrado errores para corregir. Esta fase probablemente sea la más difícil de estimar ya que siempre hay que dejar un margen de horas ya que no se sabe el tipo de problema que te puedes encontrar y el tiempo que se va a tardar en encontrar una solución.

## 8.2. Conclusiones y trabajo futuro

En este apartado se van a exponer las conclusiones de cada apartado del proyecto y las nuevas funcionalidades que se podrían implementar.

Empezando por el motor gráfico el resultado obtenido es el deseado desde un comienzo, aunque se pueden llevar a cabo varias mejoras. En la mayoría de sus módulos se pueden incorporar nuevas funcionalidades, así como crear nuevos módulos como un sistema de colisiones o uno de gráficos en 3D. En el módulo de interfaz de usuario se pueden añadir nuevos componentes como barras de progreso y añadir más opciones de personalización a los componentes ya creados. Por último, estaría bien incorporar al motor algunas de las clases implementadas de forma exclusiva para los juegos desarrollados como puede ser el paquete de conexión cliente-servidor.

En cuanto a los juegos implementados se han desarrollado según lo planificado. Sin embargo, el apartado gráfico se puede mejorar bastante y debería ser una prioridad en el futuro. La parte visual es una de las partes más importantes de un juego ya es en lo primero que se fijan los jugadores a la hora de decidir si le dan una oportunidad al juego o no. Respecto a la jugabilidad de los juegos se puede decir que es bastante completa en ambos casos y se ha logrado lo que se esperaba desde un principio. Por añadir algún nuevo elemento al juego Space-Defense se podrían incorporar nuevos tipos de enemigos o bonificaciones.

Finalmente, en la plataforma de juego se han implementado todas las características pensadas en un principio menos una, el apartado social. Si se pretende que la plataforma siga creciendo y tenga éxito deberá implementarse una parte social donde los usuarios puedan comunicarse entre sí desde dentro de la aplicación y formar así una comunidad de jugadores. También será necesario incorporar una tienda en caso de que se desee obtener beneficios tal y como se especifica en la evaluación económica.

En cuanto a la parte personal, la realización de este proyecto ha permitido el aprendizaje en el desarrollo de videojuegos tal y como se pretendía desde un principio. Hay que destacar que se han adquirido más conocimientos de los requeridos para la realización del proyecto en lo relacionado con el desarrollo de videojuegos en 2D. Esto debería permitir crear juegos más complejos en un futuro para incorporar a la plataforma, así como realizar las diferentes mejoras expuestas anteriormente en los apartados de juegos y motor gráfico.

# 9. BIBLIOGRAFÍA

En este apartado se muestran los libros y los recursos electrónicos utilizados para la realización del proyecto.

## 9.1. Libros


- [1] Davison, A. (2005). *Killer Game Programming in Java.*


## 9.2. Recursos electrónicos


- [2] Modelos de negocio: http://pelipaja.centria.fi/wp-content/uploads/2015/05/Business-Models-in-Video-Game-Industry-autum2014.pdf
- [3] Ventas videojuegos en formato físico y digital: https://www.statista.com/statistics/190225/digital-and-physical-game-sales-in-the-us-since-2009/
- [4] Uso de la librería JavaMail: https://www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/
- [5] Como calcular el Elo: https://www.mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/

# 10. ANEXO I. – CASOS DE USO EXTENDIDOS

## 10.1. Space-Defense

| | |
|---|---|
| **Nombre:** | Gestionar sonido |
| **Descripción:** Permite al usuario activar o desactivar el sonido. | |
| **Actores:** Usuario | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario hace clic en el icono de sonido.<br>[Si el sonido está activado]<br>1.1. Se desactiva el sonido.<br>[Si el sonido no está activado]<br>1.2. Se activa el sonido. | |
| **Postcondiciones:** Se activa o desactiva el sonido en función del estado. | |
| **Interfaz gráfica:** | |


Ilustración 113: MENÚ SPACE-DEFENSE

Tabla 45: CASO DE USO GESTIONAR SONIDO SPACE-DEFENSE

| **Nombre:** | Ver ranking |
|---|---|
| **Descripción:** Permite al usuario ver el ranking con las mejores puntuaciones y su mejor puntuación obtenida. | |
| **Actores:** Usuario | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br><br>1. El usuario hace clic en la opción "Ranking".<br><br>2. Se muestra una pantalla con la clasificación de los 10 mejores jugadores. Se incluye el nombre usuario y su mejor puntuación.<br><br>[Si el usuario no está en la clasificación]<br><br>2.1. Se muestra la mejor puntuación del usuario debajo de la lista. | |
| **Postcondiciones:** - | |
| **Interfaz gráfica:**<br><br>  <br><br><br>Ilustración 114: MENÚ SPACE-DEFENSE<br><br>  <br><br><br>Ilustración 115: VENTANA RANKING SPACE-DEFENSE | |

Tabla 46: CASO DE USO VER RANKING SPACE-DEFENSE

| **Nombre:** | Jugar |
|---|---|
| **Descripción:** Permite al usuario jugar una partida. | |
| **Actores:** Usuario | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario hace clic en la opción "Play".<br>2. Se muestra al usuario la pantalla de juego.<br>[Mientras la partida no haya finalizado]<br>[Si el usuario quiere moverse]<br>2.1. EXTEND MOVERSE<br>[Si el usuario quiere disparar]<br>2.2. EXTEND DISPARAR<br>[Si el usuario quiere pausar la partida]<br>2.3. EXTEND PAUSAR<br>[Si el usuario quiere reanudar la partida]<br>2.4. EXTEND REANUDAR<br>[Si el usuario quiere finalizar la partida]<br>2.5. EXTEND FINALIZAR PARTIDA | |
| **Postcondiciones:** - | |
| **Interfaz gráfica:** | |


Ilustración 116: MENÚ SPACE-DEFENSE


Ilustración 117: VENTANA JUEGO SPACE-DEFENSE

Tabla 47: CASO DE USO JUGAR SPACE-DEFENSE

| **Nombre:** | Moverse |
|---|---|
| **Descripción:** Permite al usuario mover la nave horizontalmente a través del tablero. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario presiona la tecla A o D.<br>[Si ha pulsado A]<br>1.1. Se mueve la nave hacia la izquierda.<br>[Si ha pulsado D]<br>1.2. Se mueve la nave hacia la derecha. | |
| **Postcondiciones:** Se actualiza la posición de la nave. | |
| **Interfaz gráfica:** | |


Ilustración 118: VENTANA JUEGO SPACE-DEFENSE

Tabla 48: CASO DE USO MOVERSE SPACE-DEFENSE

| | |
|---|---|
| **Nombre:** | Pausar |
| **Descripción:** Permite al usuario pausar la partida. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario presiona la tecla P.<br>2. Se pausa la partida y se muestra la pantalla de pausa. | |
| **Postcondiciones:** La partida queda pausada. | |
| **Interfaz gráfica:** | |


Ilustración 119: VENTANA JUEGO SPACE-DEFENSE


Ilustración 120: VENTANA PAUSA SPACE-DEFENSE

Tabla 49: CASO DE USO PAUSAR SPACE-DEFENSE

| **Nombre:** | Reanudar |
|---|---|
| **Descripción:** Permite al usuario reanudar una partida pausada. | |
| **Actores:** Usuario | |
| **Precondición:** La partida debe estar pausada. | |
| **Requisitos no funcionales**: - | |
| **Flujo de eventos:**<br>1. El usuario presiona la tecla P.<br>2. Se vuelve a la ventana de juego. | |
| **Postcondiciones:** Se reanuda la partida con la misma situación del tablero previo a la pausa. | |
| **Interfaz gráfica:** | |


Ilustración 121: VENTANA PAUSA SPACE-DEFENSE


Ilustración 122: VENTANA JUEGO SPACE-DEFENSE

Tabla 50: CASO DE USO REANUDAR SPACE-DEFENSE

| **Nombre:** | Disparar |
|---|---|
| **Descripción:** Permite al usuario disparar balas. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida y debe haber pasado un determinado tiempo desde el último disparo. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario hace clic izquierdo sobre un punto del tablero.<br>2. Se dispara una bala hacia ese punto. | |
| **Postcondiciones:** Se dispara una bala. | |
| **Interfaz gráfica:** | |


Ilustración 123: VENTANA JUEGO SPACE-DEFENSE

Tabla 51: CASO DE USO DISPARAR SPACE-DEFENSE

| **Nombre:** | Finalizar partida |
|---|---|
| **Descripción:** Permite al usuario finalizar la partida antes de que esta acabe por muerte del jugador. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario presiona la tecla Esc.<br>2. Se finaliza la partida y se muestra la pantalla de Game Over con la puntuación obtenida.<br>[Si el usuario hace clic en la opción Play Again]<br>2.1. EXTEND JUGAR | |
| **Postcondiciones**: Se actualiza la puntuación del jugador si esta fuera la mejor obtenida hasta el momento. | |
| **Interfaz gráfica:**<br>  <br><br>Ilustración 124: VENTANA JUEGO SPACE-DEFENSE | |

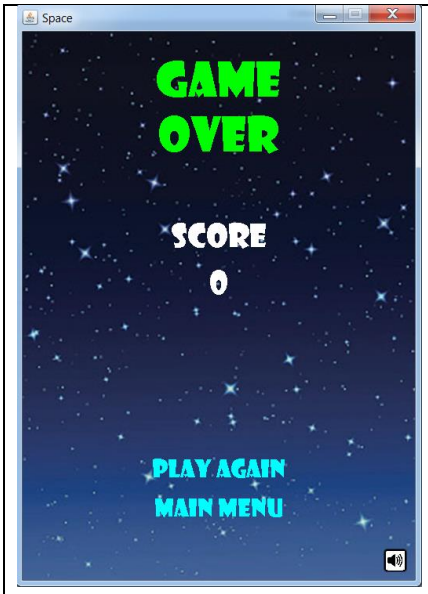



Ilustración 125: VENTANA GAME OVER SPACE-DEFENSE

Tabla 52: CASO DE USO FINALIZAR PARTIDA SPACE-DEFENSE

## 10.2. Ping-Pong

| **Nombre:** | Gestionar sonido |
|---|---|
| **Descripción:** Permite al usuario activar o desactivar el sonido. | |
| **Actores:** Usuario | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario hace clic en el icono de sonido.<br>[Si el sonido está activado]<br>1.1. Se desactiva el sonido.<br>[Si el sonido no está activado]<br>1.2. Se activa el sonido. | |
| **Postcondiciones:** Se activa o desactiva el sonido en función del estado. | |
| **Interfaz gráfica:** | |


Ilustración 126: MENÚ PING-PONG

Tabla 53: CASO DE USO GESTIONAR SONIDO PING-PONG

| **Nombre:** | Ver ranking |
|---|---|
| **Descripción:** Permite al usuario ver el ranking con los mejores usuarios y su Elo correspondiente. | |
| **Actores:** Usuario | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br><br>1. El usuario hace clic en la opción "Ranking".<br><br>2. Se muestra una pantalla con la clasificación de los 10 mejores jugadores. Se incluye el nombre usuario y el Elo correspondiente que posee actualmente.<br><br>[Si el usuario no está en la clasificación]<br><br>2.1. Se muestra el Elo del jugador debajo de la lista. | |
| **Postcondiciones:** - | |
| **Interfaz gráfica:** | |


Ilustración 127: MENÚ PING-PONG


Ilustración 128: VENTANA RANKING PING-PONG

Tabla 54: CASO DE USO VER RANKING PING-PONG

| **Nombre:** | Jugar |
|---|---|

**Descripción:** Permite al usuario jugar una partida.

**Actores:** Usuario

**Precondición:** El usuario debe tener conexión a internet.

**Requisitos no funcionales:** -

**Flujo de eventos:**

1. El usuario hace clic en la opción "Play".
2. Se muestra al usuario la pantalla de búsqueda de oponente.

   [Si encuentra un oponente]

   2.1. Se muestra la pantalla de juego.

   [Mientras la partida no haya finalizado]

   [Si el usuario quiere moverse]

   2.1.1. EXTEND MOVERSE

   [Si el usuario quiere pausar la partida]

   2.1.2. EXTEND PAUSAR

   [Si el usuario quiere reanudar la partida]

   2.1.3. EXTEND REANUDAR

   [Si el usuario quiere finalizar la partida]

   2.1.4. EXTEND FINALIZAR PARTIDA

**Postcondiciones:** -

**Interfaz gráfica:**


Ilustración 129: MENÚ PING-PONG

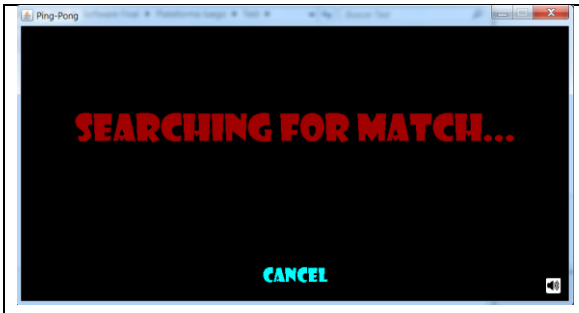



Ilustración 130: VENTANA BUSCAR PARTIDA

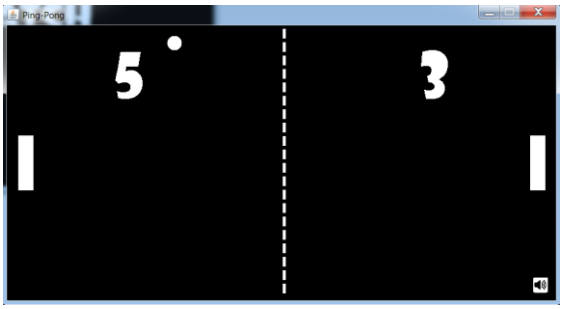



Ilustración 131: VENTANA JUEGO PING-PONG

Tabla 55: CASO DE USO JUGAR

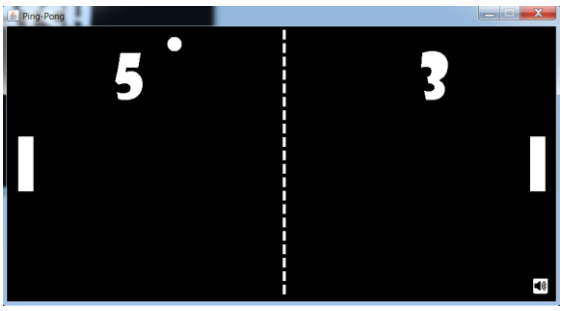
| **Nombre:** | Moverse |
|---|---|
| **Descripción:** Permite al usuario moverse con su paleta a través del tablero de juego verticalmente. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>2. El usuario presiona la tecla W o S.<br>[Si ha pulsado W]<br>2.1. Se mueve la paleta del usuario hacia arriba.<br>[Si ha pulsado S]<br>2.2. Se mueve la paleta del usuario hacia abajo. | |
| **Postcondiciones:** Se actualiza la posición de la paleta. | |
| **Interfaz gráfica:**<br>  <br><br>Ilustración 132: VENTANA JUEGO PING-PONG | |

Tabla 56: CASO DE USO MOVERSE

| | |
|---|---|
| **Nombre:** | Pausar |
| **Descripción:** Permite al usuario pausar la partida. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>3. El usuario presiona la tecla P.<br>[Si el usuario no ha pausado todavía la partida]<br>3.1. Se pausa la partida y se muestra la pantalla de pausa. | |
| **Postcondiciones:** La partida queda pausada siempre y cuando sea posible. | |
| **Interfaz gráfica:** | |

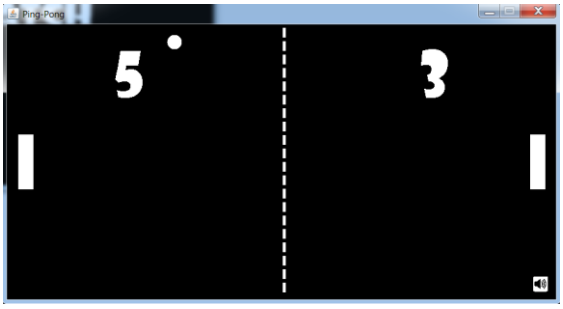



Ilustración 133: VENTANA JUEGO PING-PONG

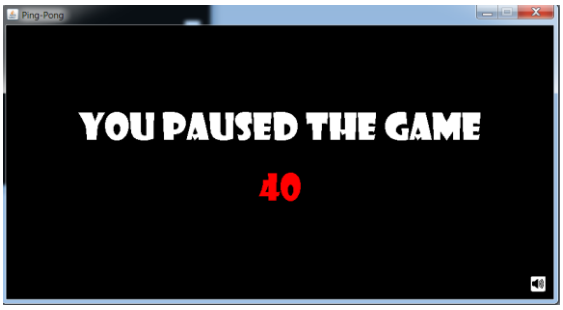



Ilustración 134: VENTANA PAUSA PING-PONG

Tabla 57: CASO DE USO PAUSAR

| **Nombre:** | Reanudar |
|---|---|
| **Descripción:** Permite al usuario reanudar una partida pausada. | |
| **Actores:** Usuario | |
| **Precondición:** La partida debe estar pausada. | |
| **Requisitos no funcionales**: - | |
| **Flujo de eventos:**<br>3. El usuario presiona la tecla P.<br>[Si el usuario ha sido el causante de la pausa]<br>3.1. Se vuelve a la ventana de juego tras una cuenta atras. | |
| **Postcondiciones:** Se reanuda la partida con la misma situación del tablero previo a la pausa. | |
| **Interfaz gráfica:** | |

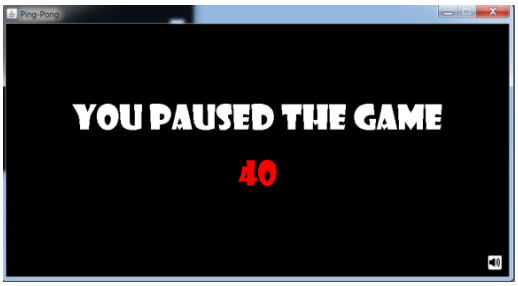



Ilustración 135: VENTANA PAUSA PING-PONG


Ilustración 136: VENTANA REANUDACIÓN PING-PONG

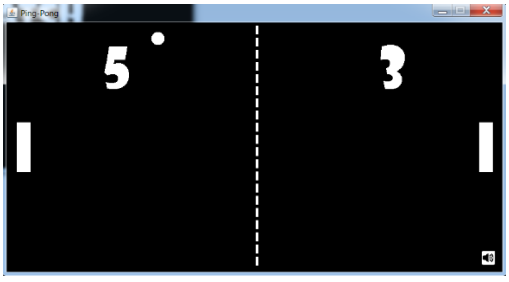



Ilustración 137: VENTANA JUEGO PING-PONG

Tabla 58: CASO DE USO REANUDAR

| | |
|---|---|
| **Nombre:** | Finalizar partida |
| **Descripción:** Permite al usuario finalizar la partida antes de que esta acabe. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario tiene que estar jugando una partida. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario presiona la tecla Esc.<br>2. Se finaliza la partida y se muestra la pantalla de Game Over. | |
| **Postcondiciones:** Se actualiza el Elo del usuario. | |
| **Interfaz gráfica:** | |

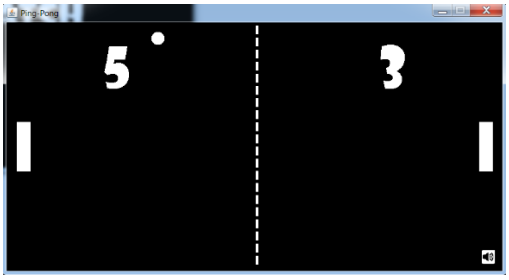



Ilustración 138: VENTANA JUEGO PING-PONG


Ilustración 139: VENTANA GAME OVER PING-PONG

Tabla 59: CASO DE USO FINALIZAR PARTIDA

| **Nombre:** | Iniciar servidor |
|---|---|
| **Descripción:** Permite al administrador iniciar el servidor para que los usuarios puedan jugar. | |
| **Actores:** Administrador | |
| **Precondición:** - | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br><br>1. El administrador pulsa el botón de inicio de servidor. | |
| **Postcondiciones:** Se inicia el servidor y se permite que los usuarios puedan jugar al juego Ping-Pong. | |
| **Interfaz gráfica:** | |

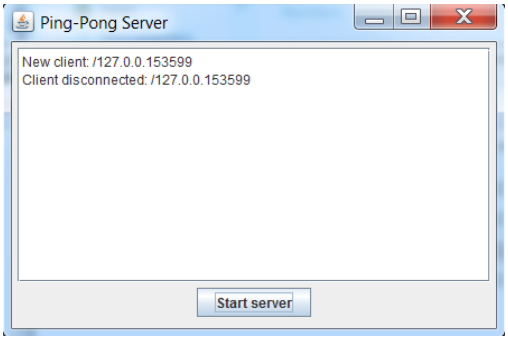



Ilustración 140: VENTANA SERVIDOR

Tabla 60: CASO DE USO INICIAR SERVIDOR

## 10.3. Plataforma de juego

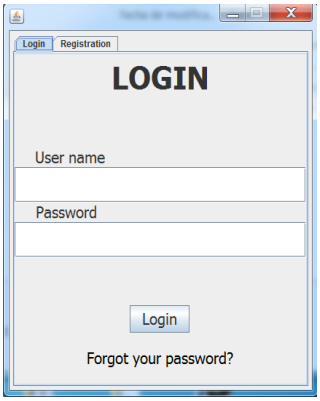
| **Nombre:** | Iniciar sesión |
|---|---|
| **Descripción:** Permite acceder a la plataforma de juego. | |
| **Actores:** Cualquiera | |
| **Precondición:** El usuario tiene que haberse registrado previamente. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario introduce un nombre de usuario y una contraseña.<br>2. El usuario pulsa el botón de inicio de sesión.<br>3. Se muestra la ventana principal de la aplicación. | |
| **Postcondiciones:** Se inicia sesión en la plataforma de juego. | |
| **Interfaz gráfica:**<br>  <br> | |

Ilustración 141: VENTANA INICIO SESIÓN


Ilustración 142: VENTANA MENÚ PRINCIPAL

Tabla 61: CASO DE USO INICIAR SESIÓN

| **Nombre:** | Registrarse |
|---|---|
| **Descripción:** Permite crear una cuenta para poder acceder a la plataforma de juego. | |
| **Actores:** Cualquiera | |
| **Precondición:** El nombre usuario y email introducidos no deben existir en el sistema. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario introduce un nombre usuario, email y contraseña.<br>2. El usuario pulsa el botón de registro. | |
| **Postcondiciones:** Se crea una nueva cuenta de usuario en el sistema. | |
| **Interfaz gráfica:**<br>  <br><br>Ilustración 143: VENTANA REGISTRO | |

Tabla 62: CASO DE USO REGISTRARSE

| **Nombre:** | Recuperar contraseña |
|---|---|
| **Descripción:** Permite recuperar la contraseña mediante el envío de un correo electrónico. | |
| **Actores:** Cualquiera | |
| **Precondición**: El usuario tiene que haberse registrado previamente. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario pulsa el botón de olvido de contraseña.<br>2. El usuario introduce un email.<br>3. El usuario pulsa el botón de recuperar. | |
| **Postcondiciones:** Se cambia la contraseña del usuario por una aleatoria y se le envía un mensaje al correo con la nueva contraseña. | |
| **Interfaz gráfica:** | |


Ilustración 144: VENTANA INICIAR SESIÓN

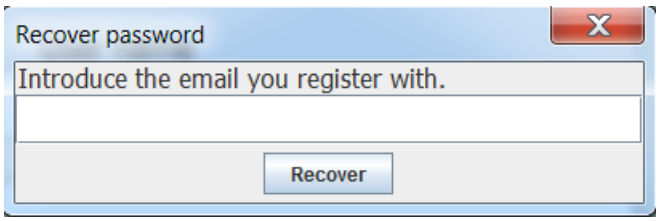



Ilustración 145: VENTANA RECUPERAR CONTRASEÑA

Tabla 63: CASO DE USO RECUPERAR CONTRASEÑA

| **Nombre:** | Cambiar contraseña |
|---|---|
| **Descripción:** Permite al usuario cambiar la contraseña de su cuenta. | |
| **Actores:** Usuario | |
| **Precondición**: El usuario ha iniciado sesión. | |
| **Requisitos no funcionales**: - | |
| **Flujo de eventos:**<br>1. El usuario selección la opción del menú de cambiar contraseña.<br>2. El usuario introduce una contraseña.<br>3. El usuario pulsa el botón de cambiar. | |
| **Postcondiciones:** Se actualiza la contraseña del usuario. | |
| **Interfaz gráfica:** | |

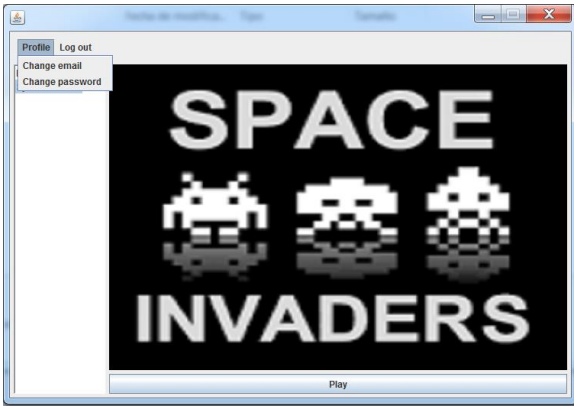



Ilustración 146: VENTANA MENÚ PRINCIPAL

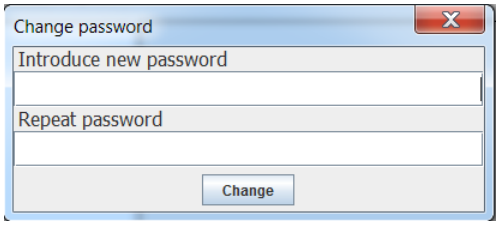



Ilustración 147: VENTANA CAMBIAR CONTRASEÑA

Tabla 64: CASO DE USO CAMBIAR CONTRASEÑA

| **Nombre:** | Cambiar correo |
|---|---|
| **Descripción:** Permite al usuario cambiar el correo electrónico de su cuenta. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario ha iniciado sesión. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario selección la opción del menú de cambiar correo.<br>2. El usuario introduce un correo electrónico.<br>3. El usuario pulsa el botón de cambiar. | |
| **Postcondiciones:** Se actualiza el correo electrónico del usuario. | |
| **Interfaz gráfica:** | |

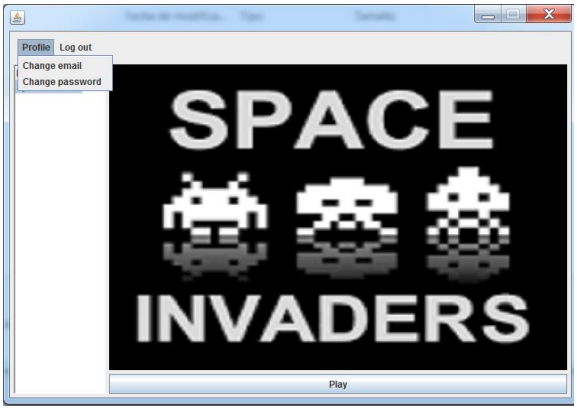



Ilustración 148: VENTANA MENÚ PRINCIPAL

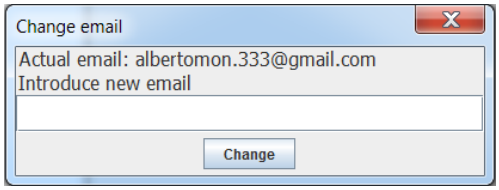



Ilustración 149: VENTANA CAMBIAR CORREO

Tabla 65: CASO DE USO CAMBIAR CORREO

| **Nombre:** | Iniciar juego |
|---|---|
| **Descripción:** Permite al usuario jugar a uno de los juegos que están disponibles. | |
| **Actores:** Usuario | |
| **Precondición:** El usuario ha iniciado sesión. | |
| **Requisitos no funcionales:** - | |
| **Flujo de eventos:**<br>1. El usuario selecciona un juego de la lista.<br>2. El usuario pulsa el botón de jugar. | |
| **Postcondiciones:** Se inicia el juego seleccionado. | |
| **Interfaz gráfica:** | |


Ilustración 150: VENTANA MENÚ PRINCIPAL

Tabla 66: CASO DE USO INICIAR JUEGO

# 11. ANEXO II.- DIAGRAMAS DE SECUENCIA

En este apartado se mostrarán los diagramas de secuencia del software desarrollado.

Para evitar que los diagramas se extiendan horizontalmente se ha decidido separar los diagramas por clases en la mayoría de los casos. De esta forma se mostrará la secuencia de las principales funcionalidades de la clase.

## 11.1. Motor gráfico

### GestorPrincipal

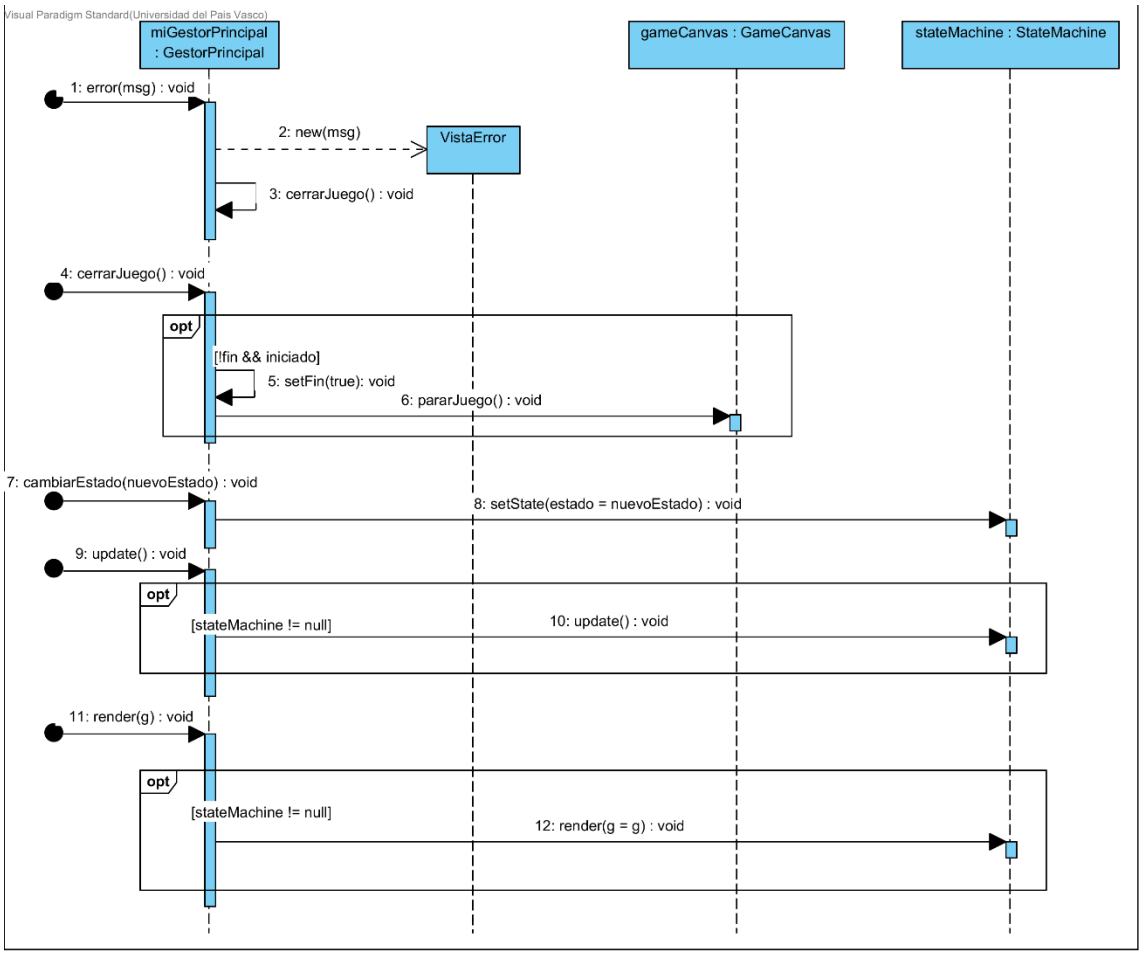



Ilustración 151: SECUENCIA GESTOR PRINCIPAL

## Inicio juego

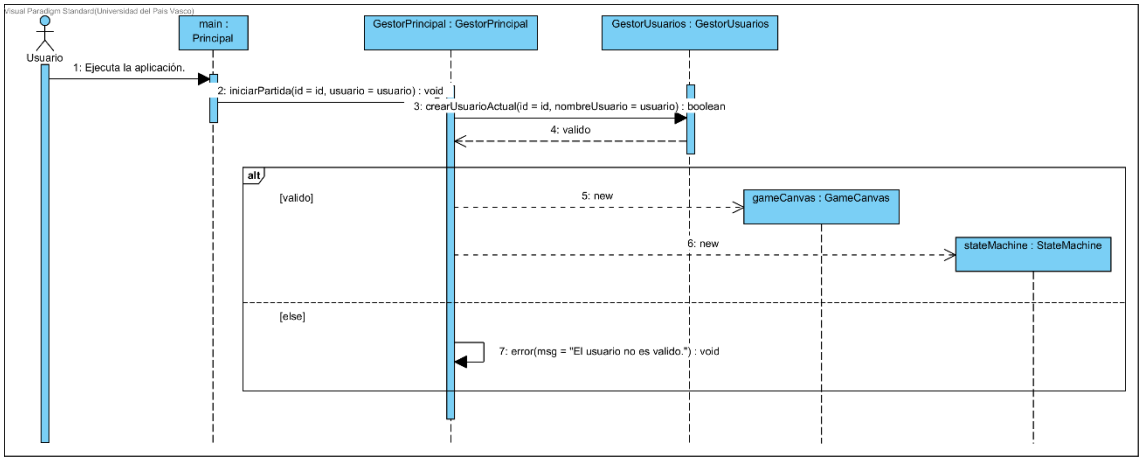



Ilustración 152: SECUENCIA INICIO JUEGO

## GameCanvas

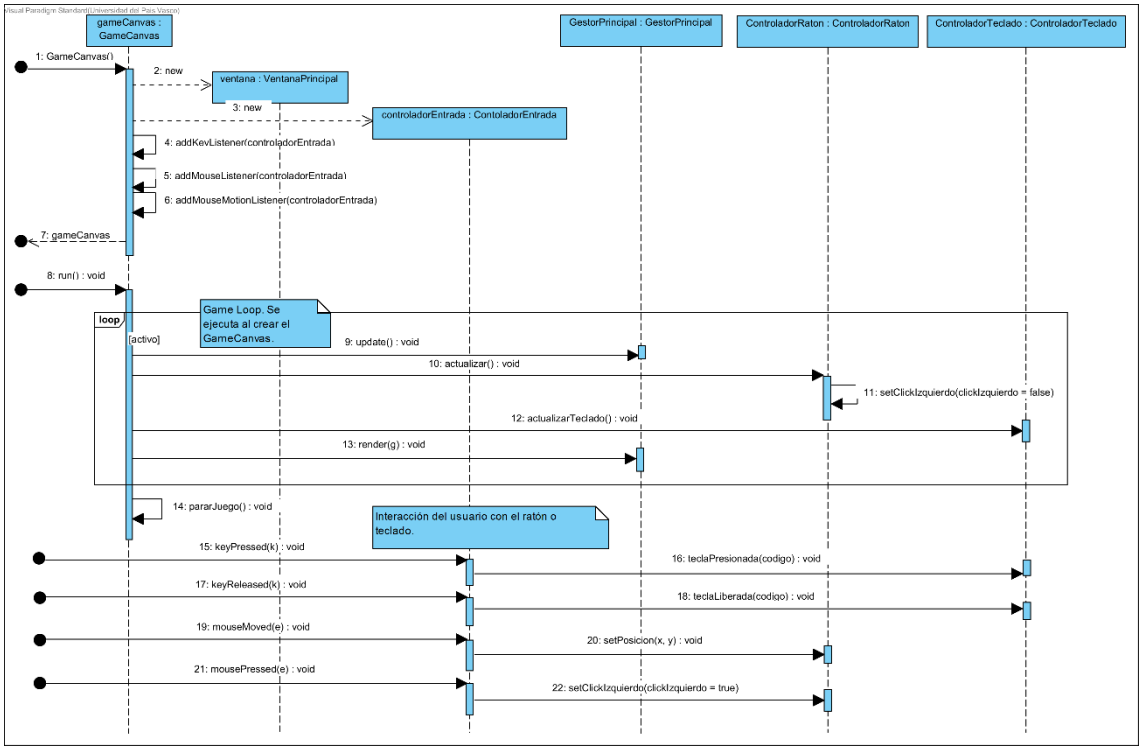



Ilustración 153: SECUENCIA GAME CANVAS

## GestorUsuarios

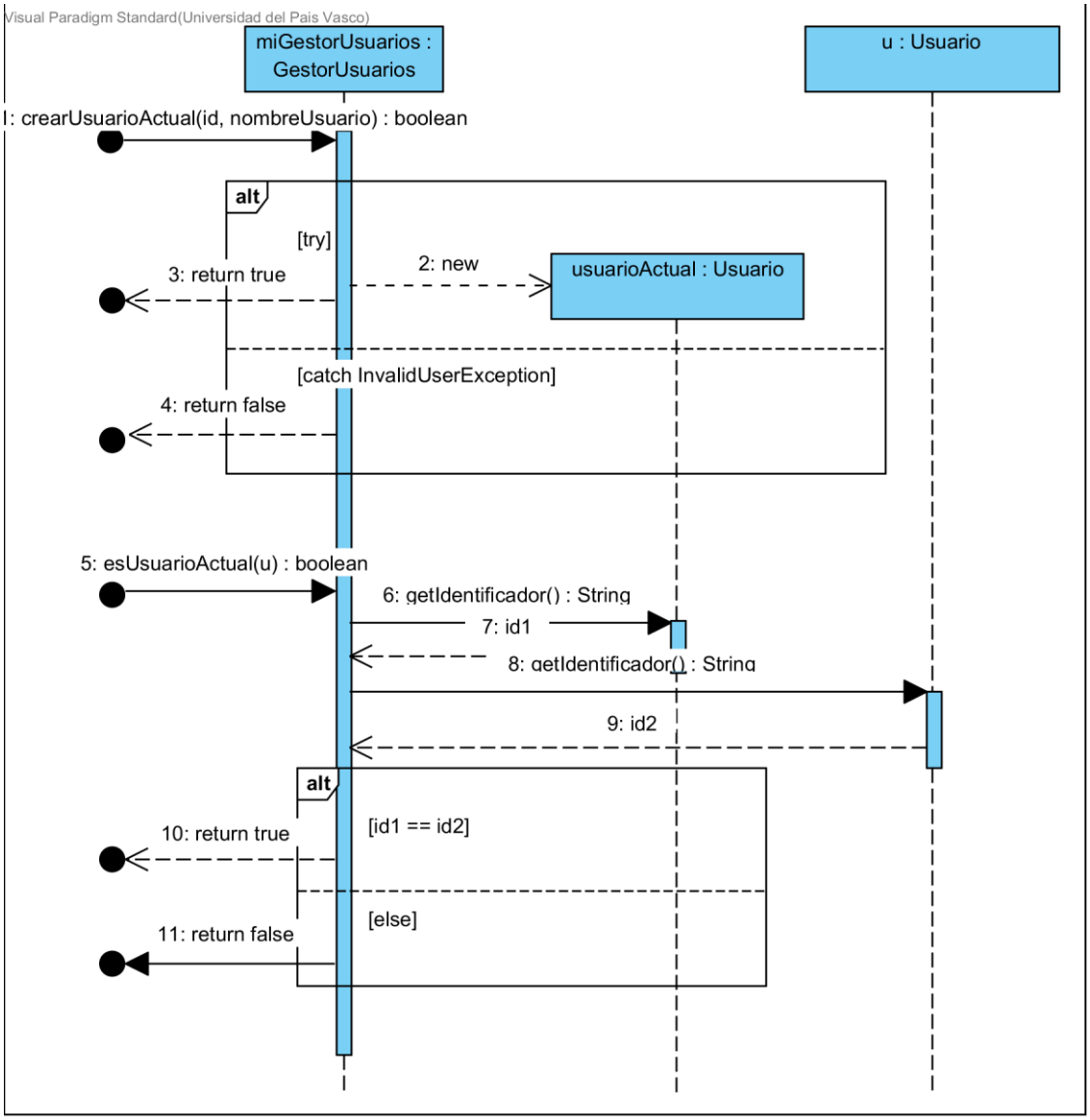



Ilustración 154: SECUENCIA GESTOR USUARIOS

## GestorSonido

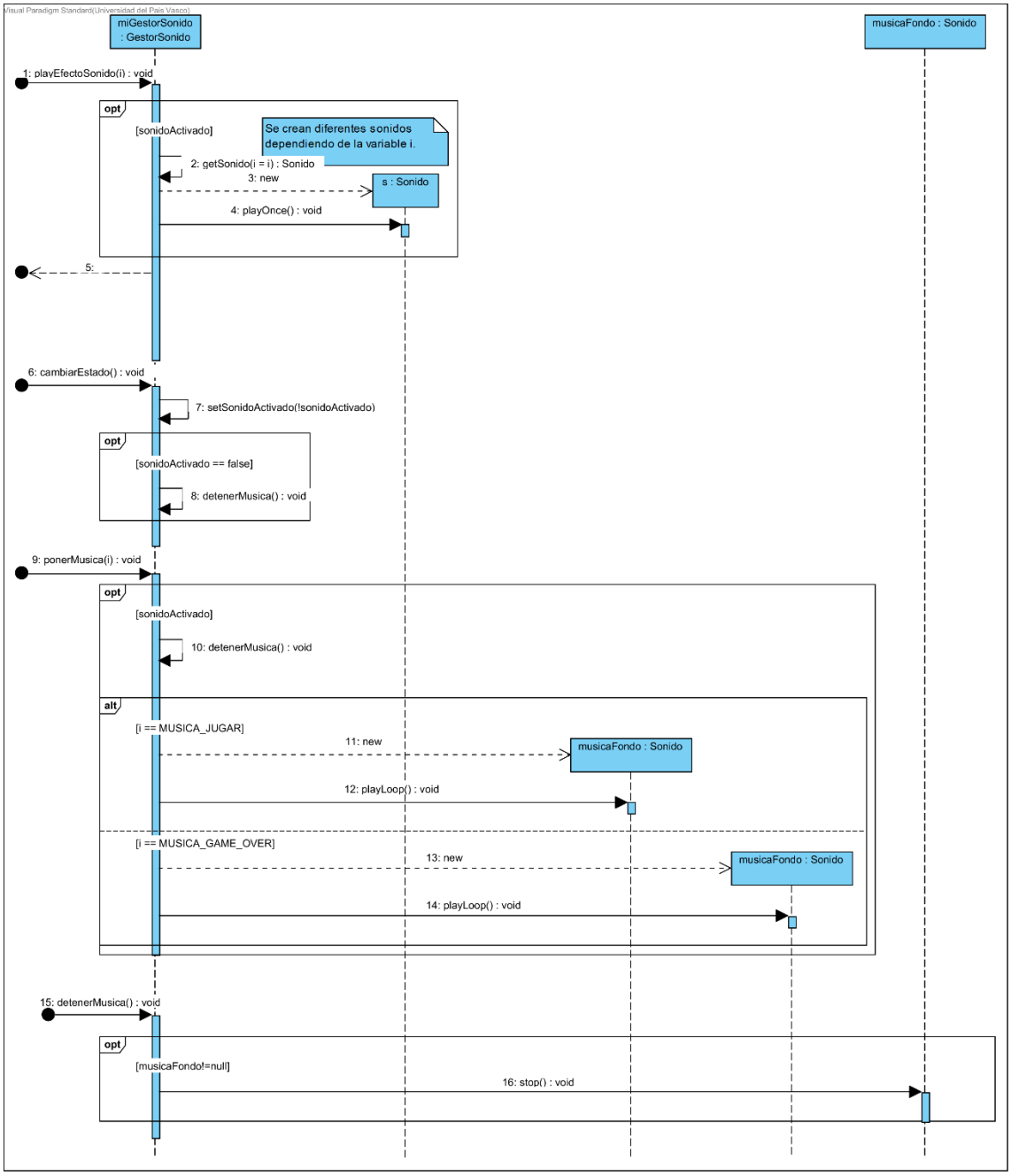



Ilustración 155: SECUENCIA GESTOR SONIDO

## IconoSonido

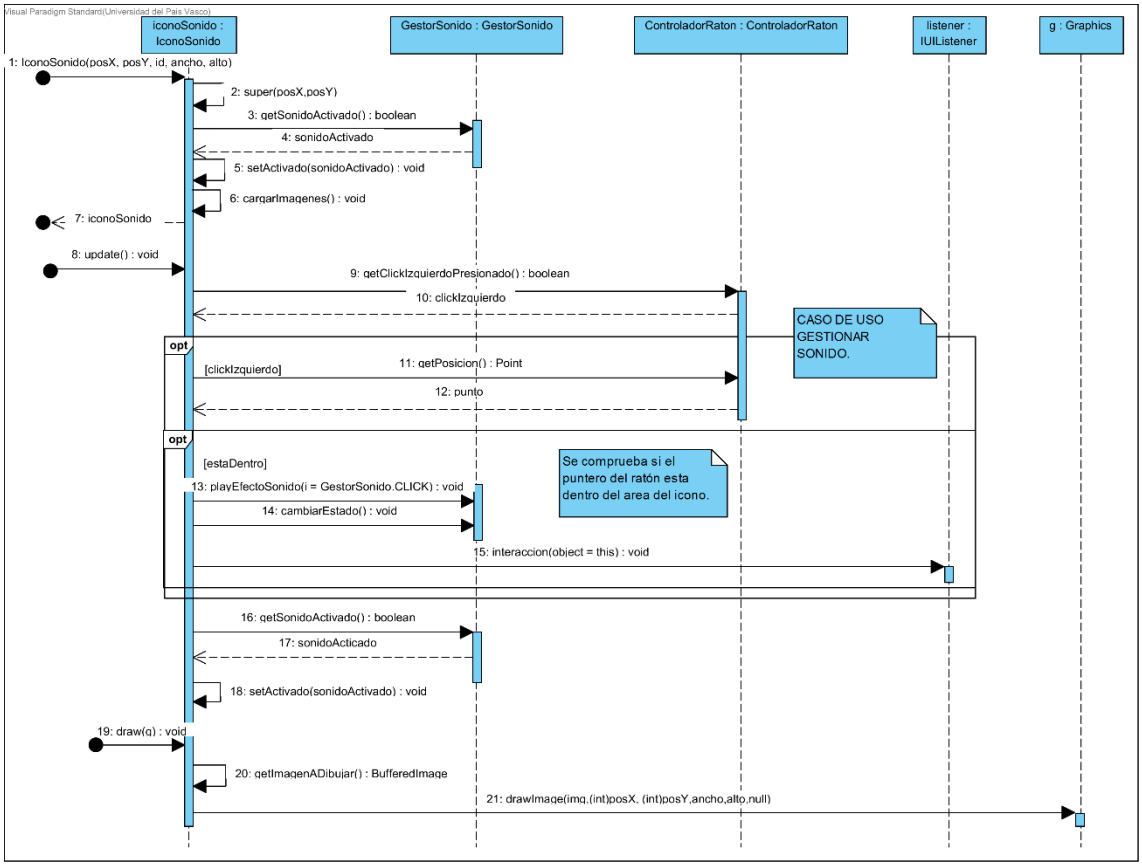



Ilustración 156: SECUENCIA ICONO SONIDO

## StateMachine

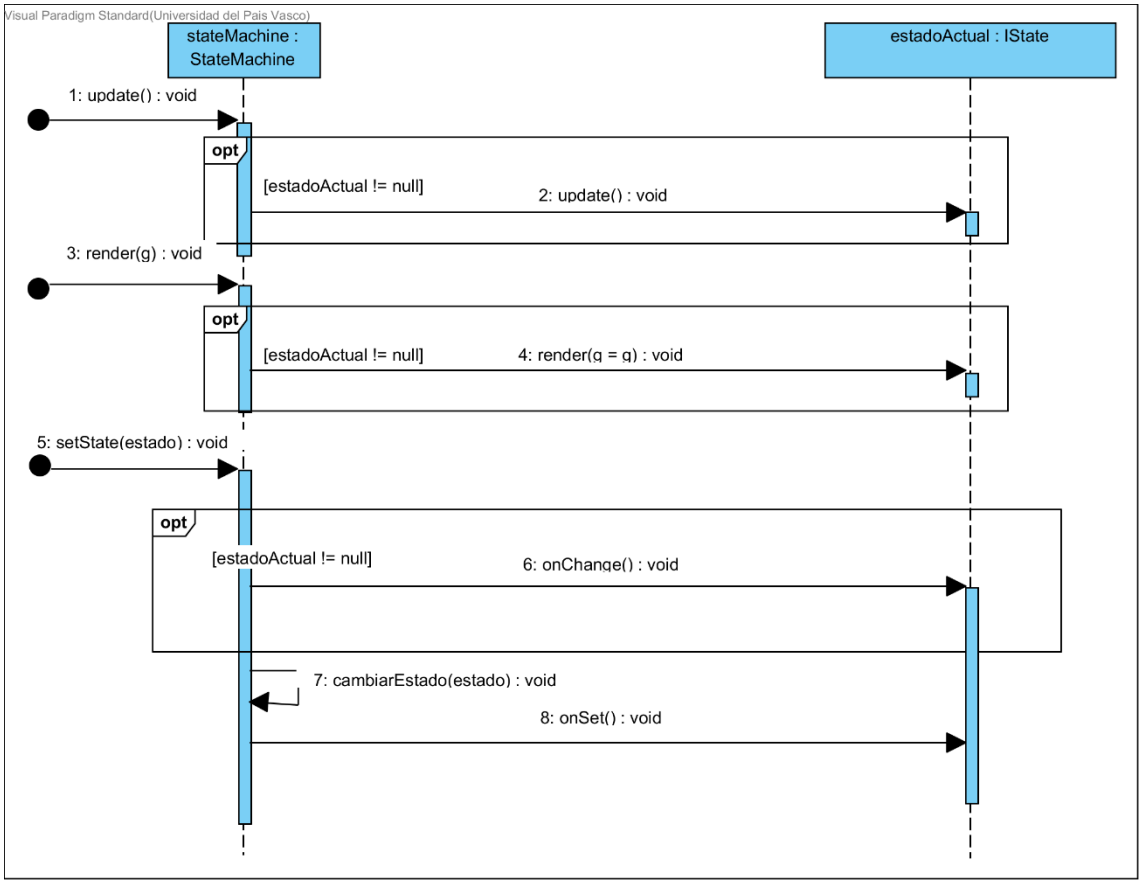



Ilustración 157: SECUENCIA STATE MACHINE

## Imagen

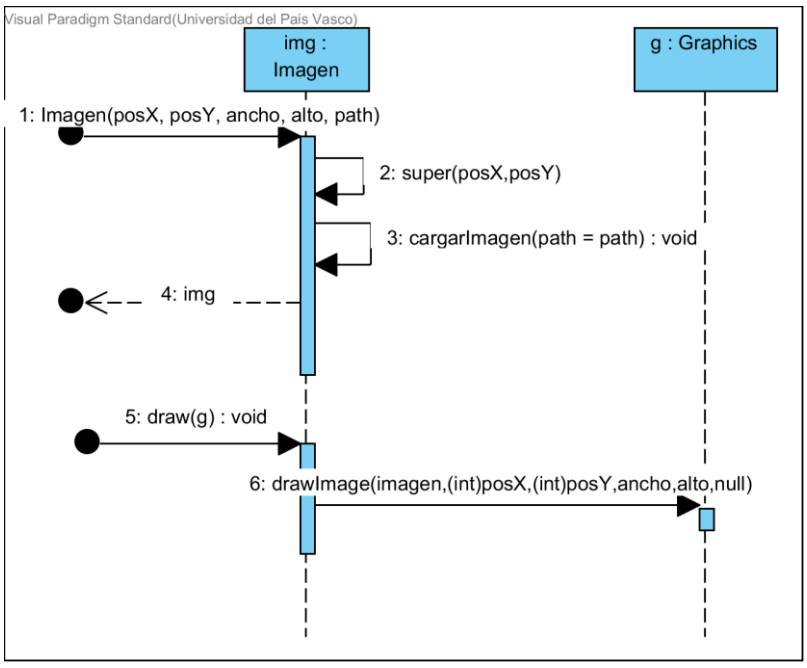



Ilustración 158: SECUENCIA IMAGEN

## BotonInterfaz

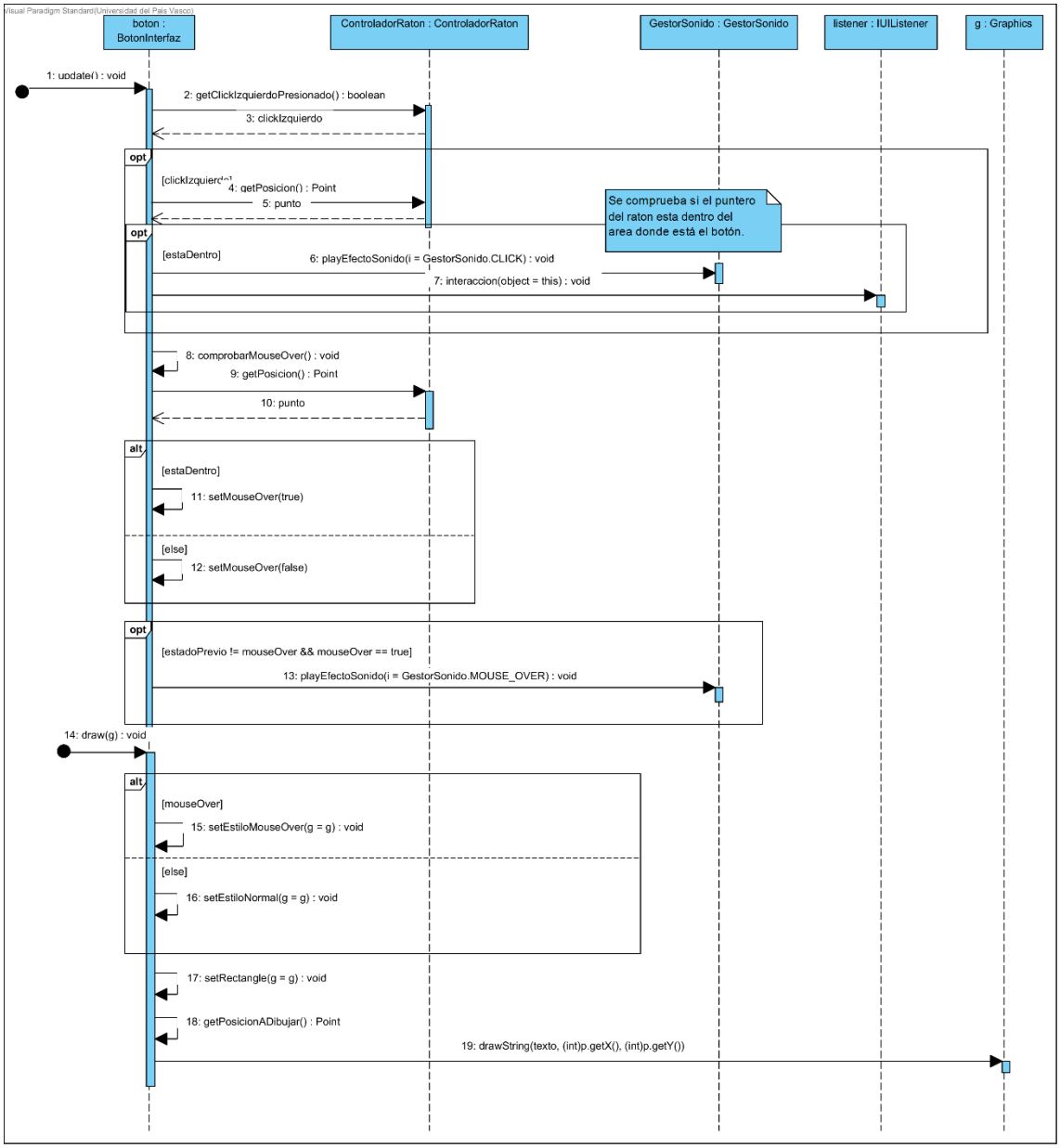



Ilustración 159: SECUENCIA BOTÓN INTERFAZ

## Texto

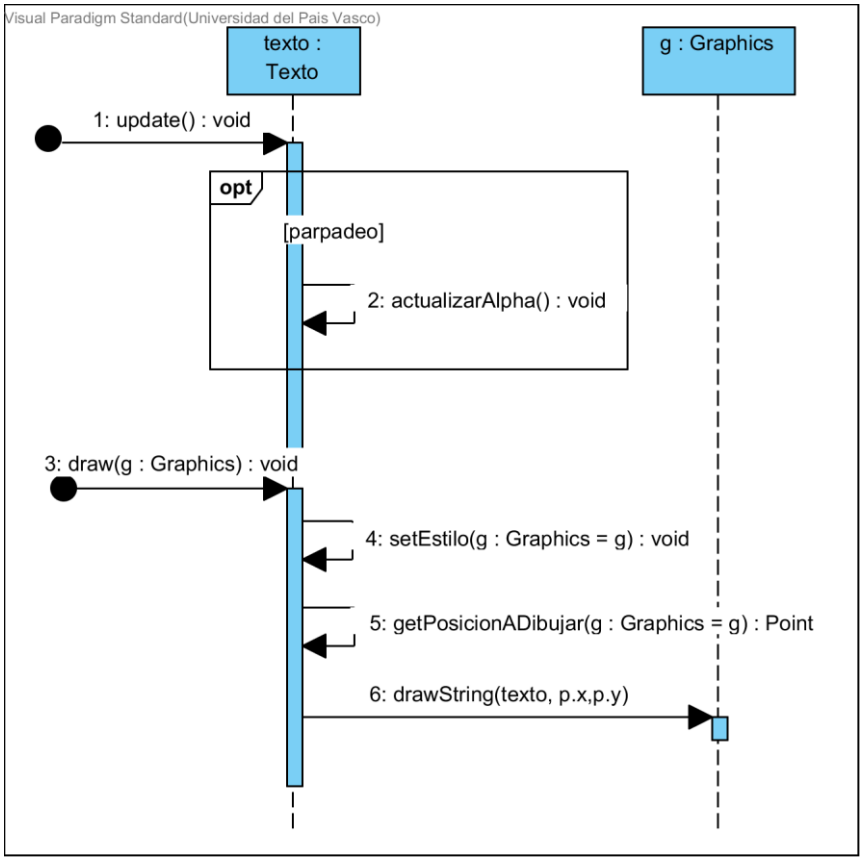



Ilustración 160: SECUENCIA TEXTO

## 11.2. Space-Defense

En este apartado se muestran los diagramas de secuencia correspondientes al juego Space-Defense.

En los diagramas correspondientes a las clases utilizadas del motor gráfico se mostrarán las funcionalidades añadidas o modificadas en caso de que se haya añadido o modificado algo.

Los casos de uso se identificarán mediante notas en el diagrama. El caso de uso Gestionar Sonido se gestiona en la clase IconoSonido del motor gráfico.

### StateMachine

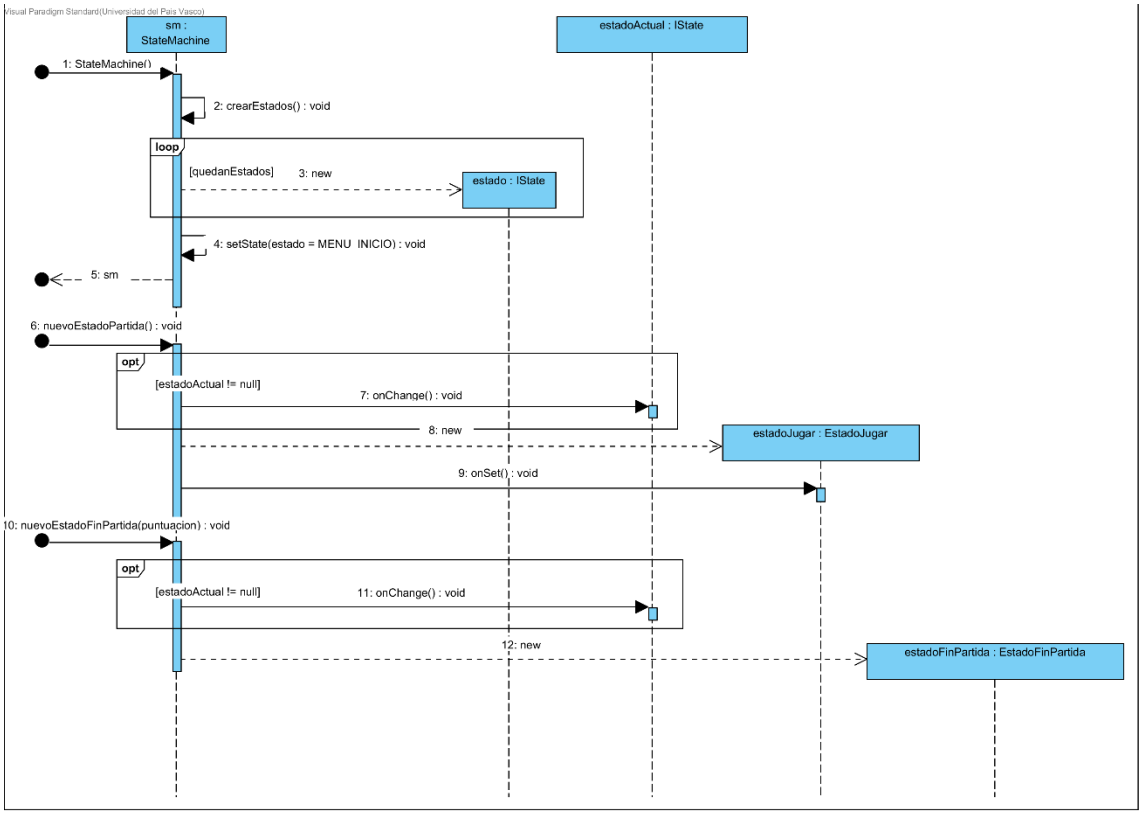



Ilustración 161: SECUENCIA STATE MACHINE

## GestorPrincipal

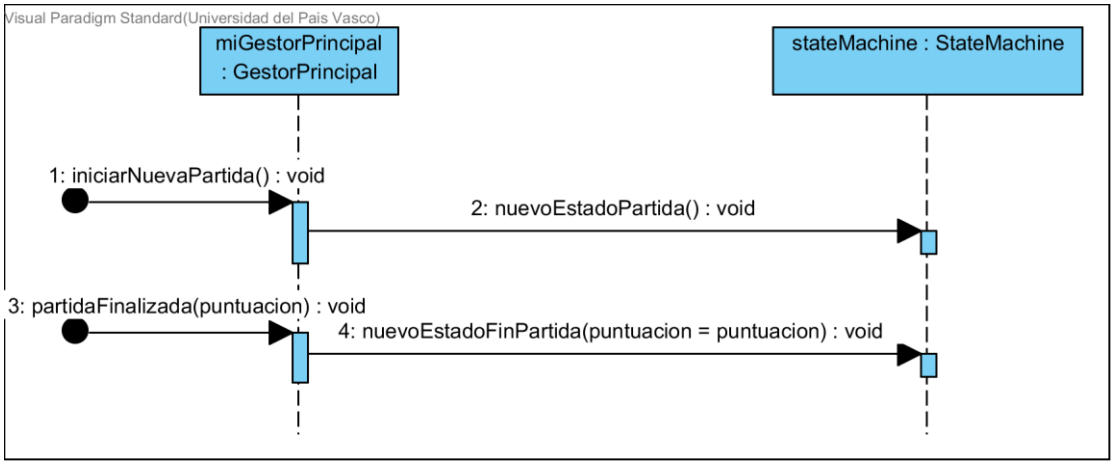



Ilustración 162: SECUENCIA GESTOR PRINCIPAL

## GestorUsuarios

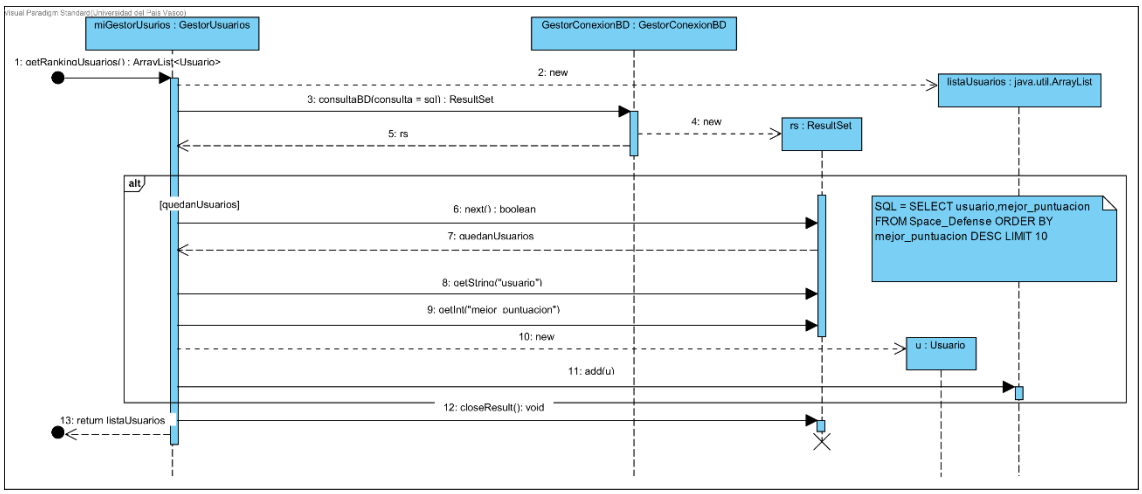



Ilustración 163: SECUENCIA GESTOR USUARIOS

## Usuario

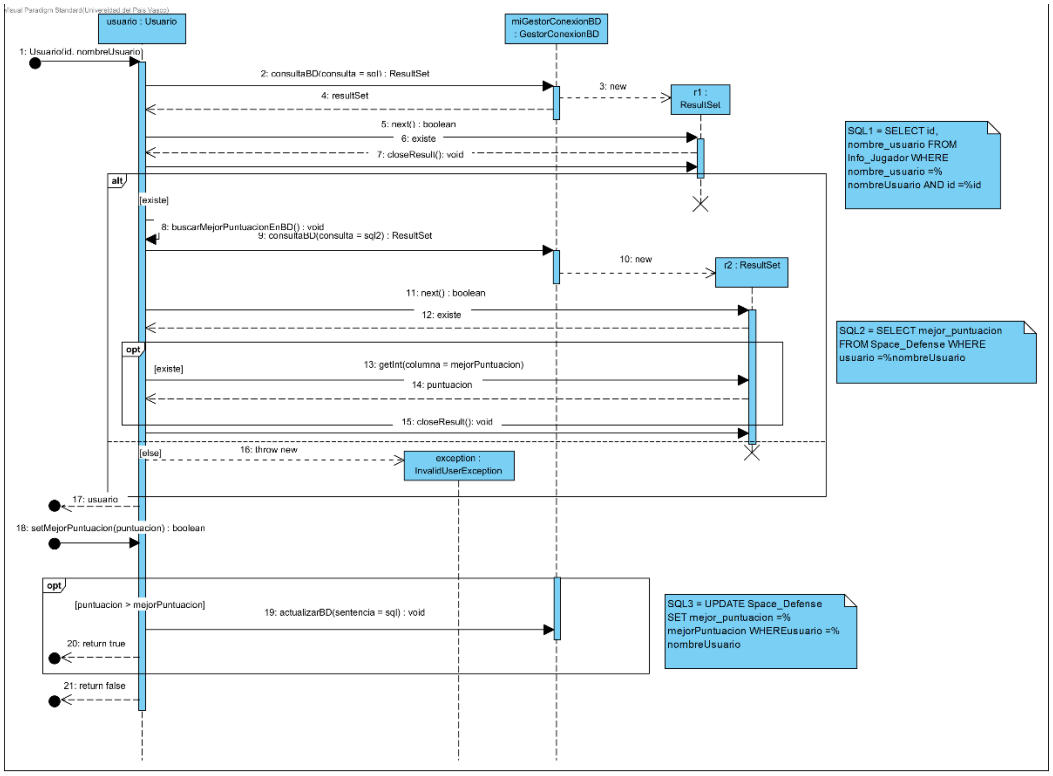



Ilustración 164: SECUENCIA USUARIO

## EstadoMenuInicial

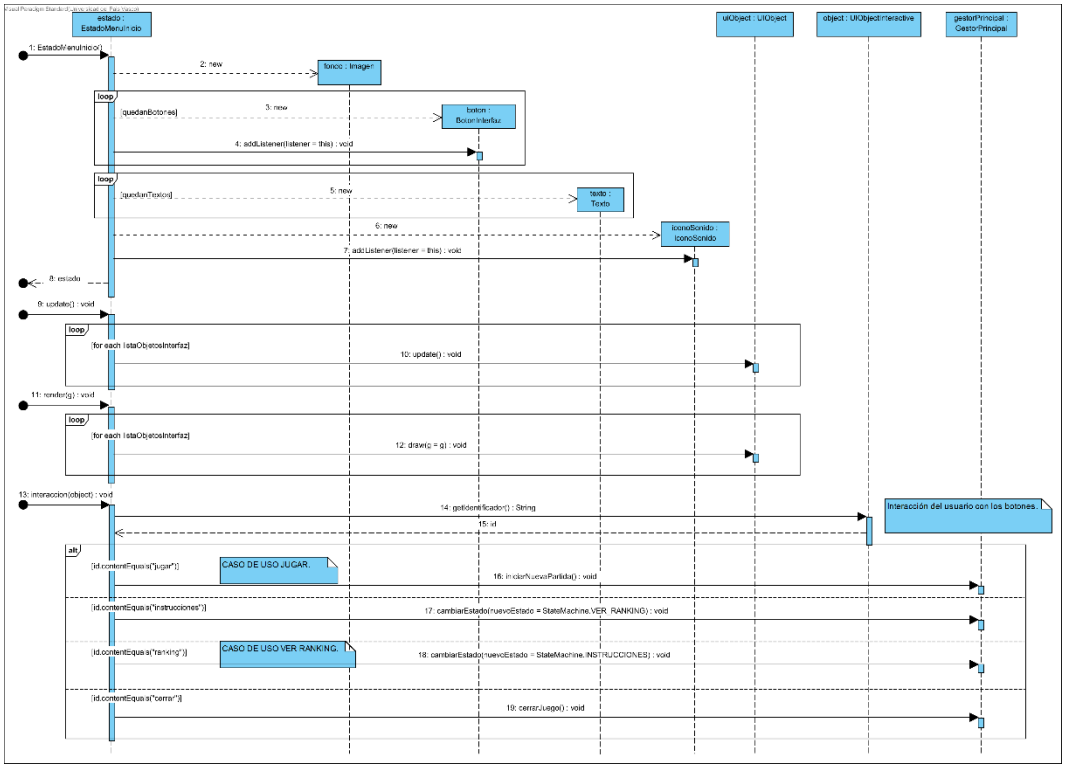



Ilustración 165: SECUENCIA ESTADO MENÚ INICIAL

## EstadoVerRanking

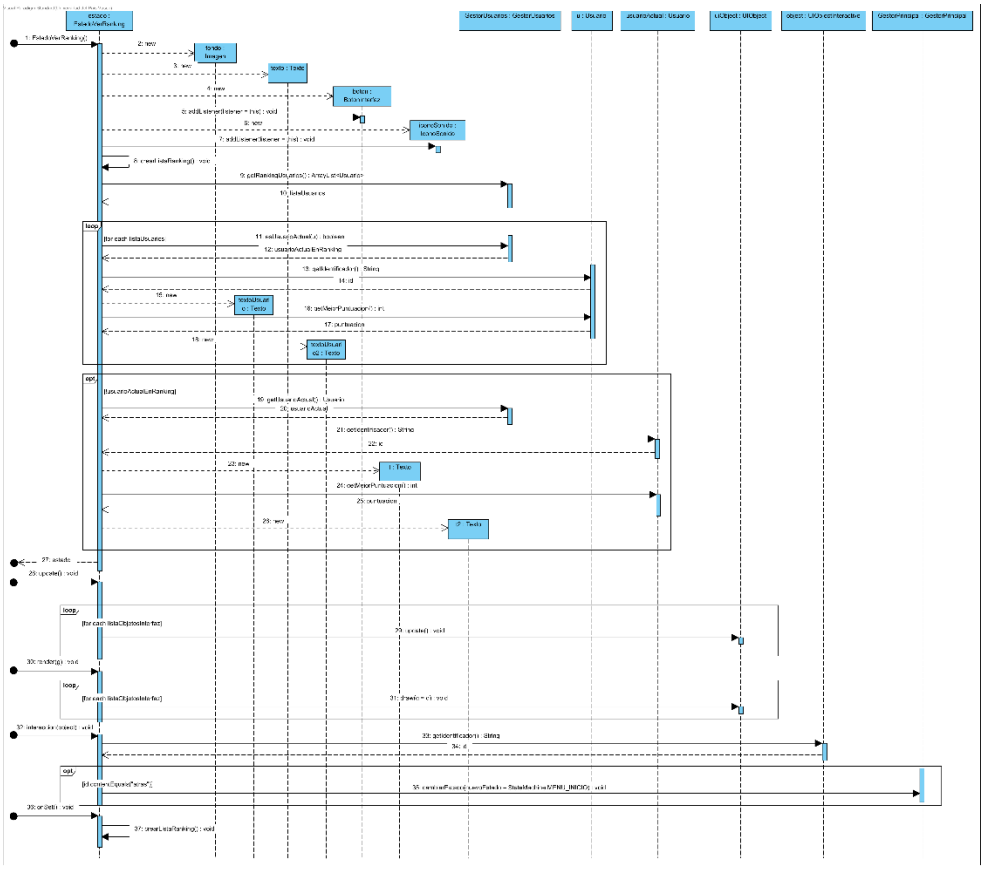


Ilustración 166: SECUENCIA ESTADO VER RANKING

## EstadoInstrucciones

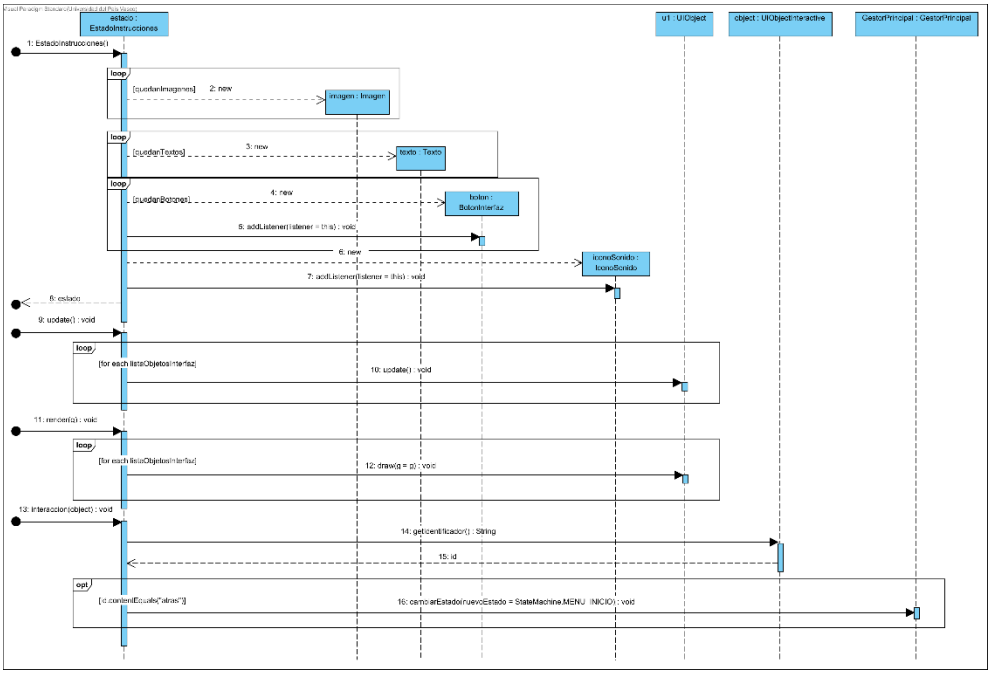



Ilustración 167: SECUENCIA ESTADO INSTRUCCIONES

## EstadoJugar

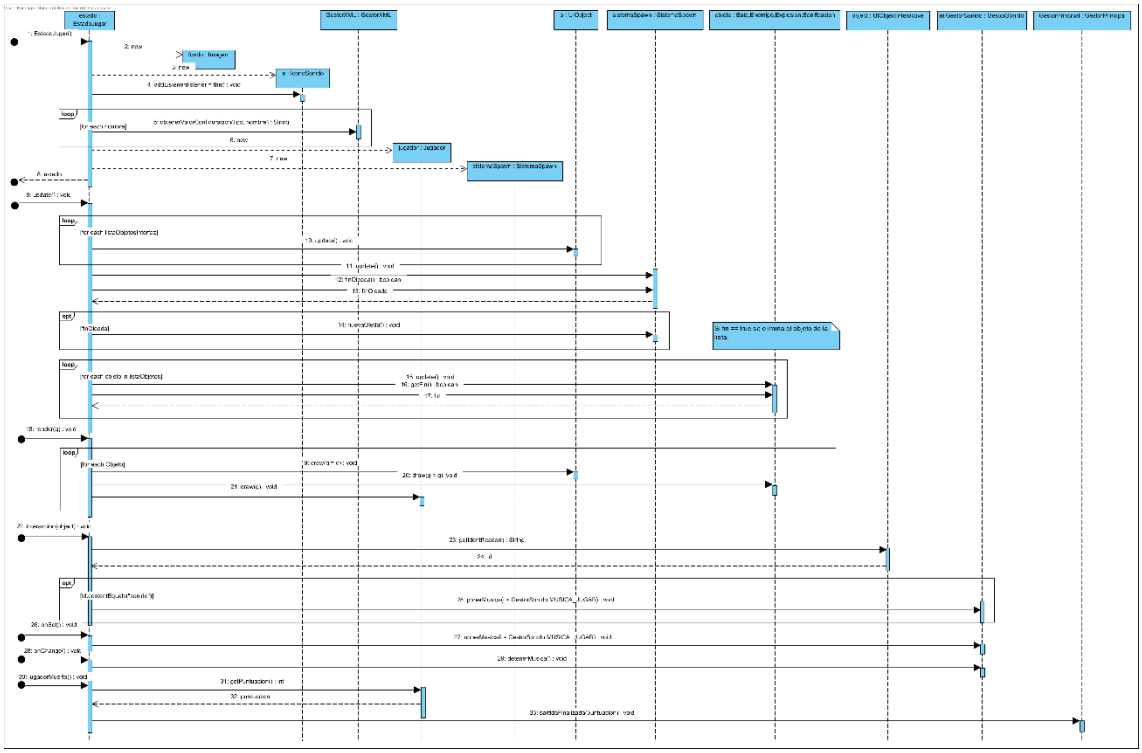


Ilustración 168: SECUENCIA ESTADO JUGAR

## EstadoPausa

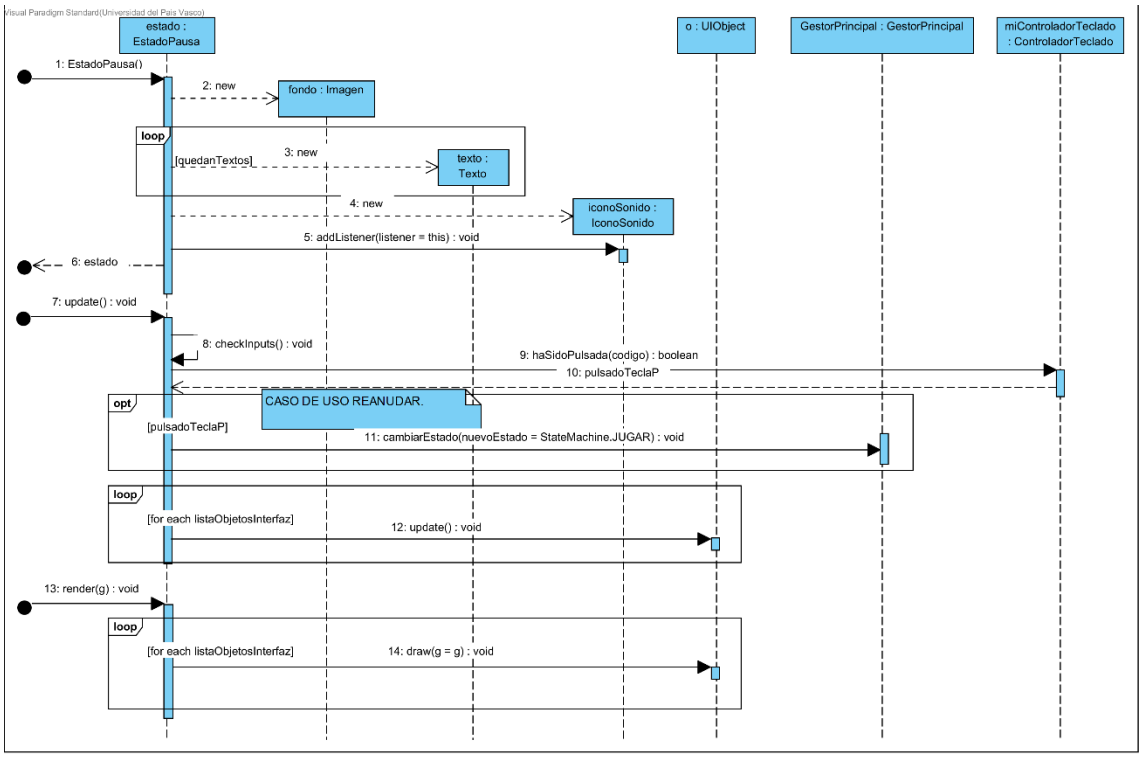



Ilustración 169: SECUENCIA ESTADO PAUSA

## EstadoFinPartida

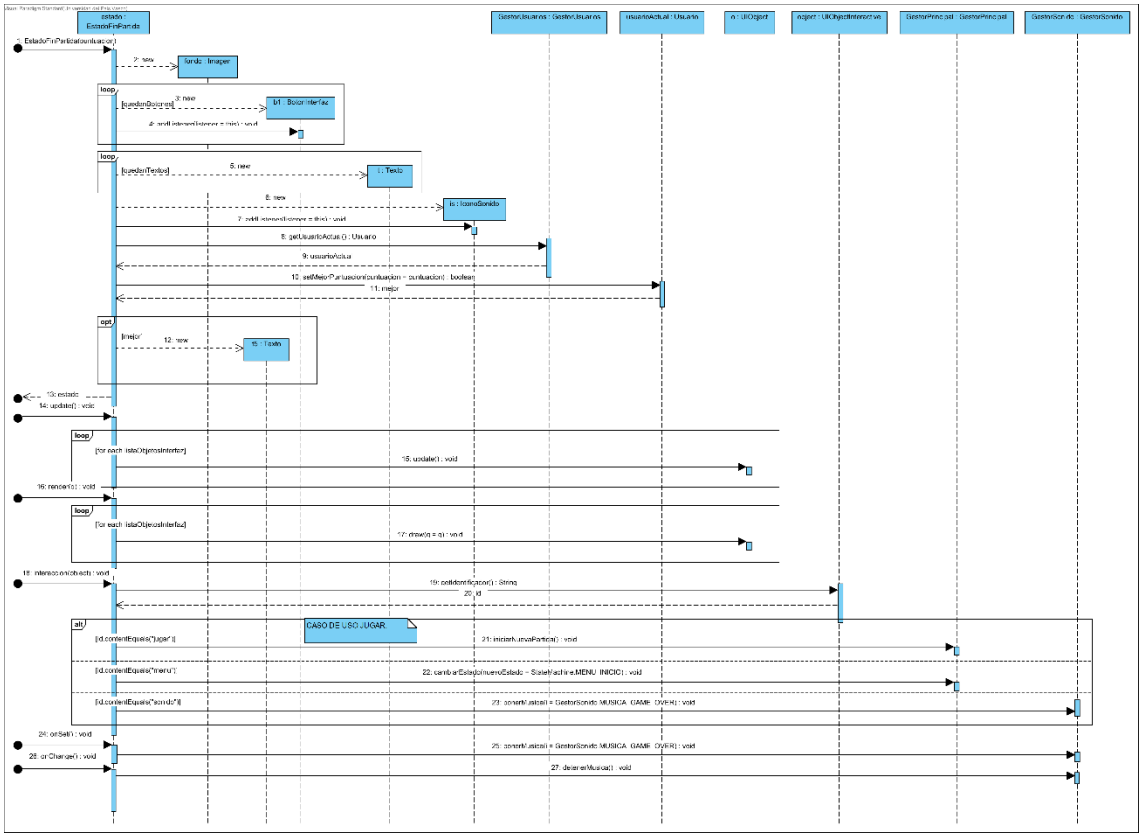



Ilustración 170: SECUENCIA ESTADO FIN PARTIDA

# Jugador

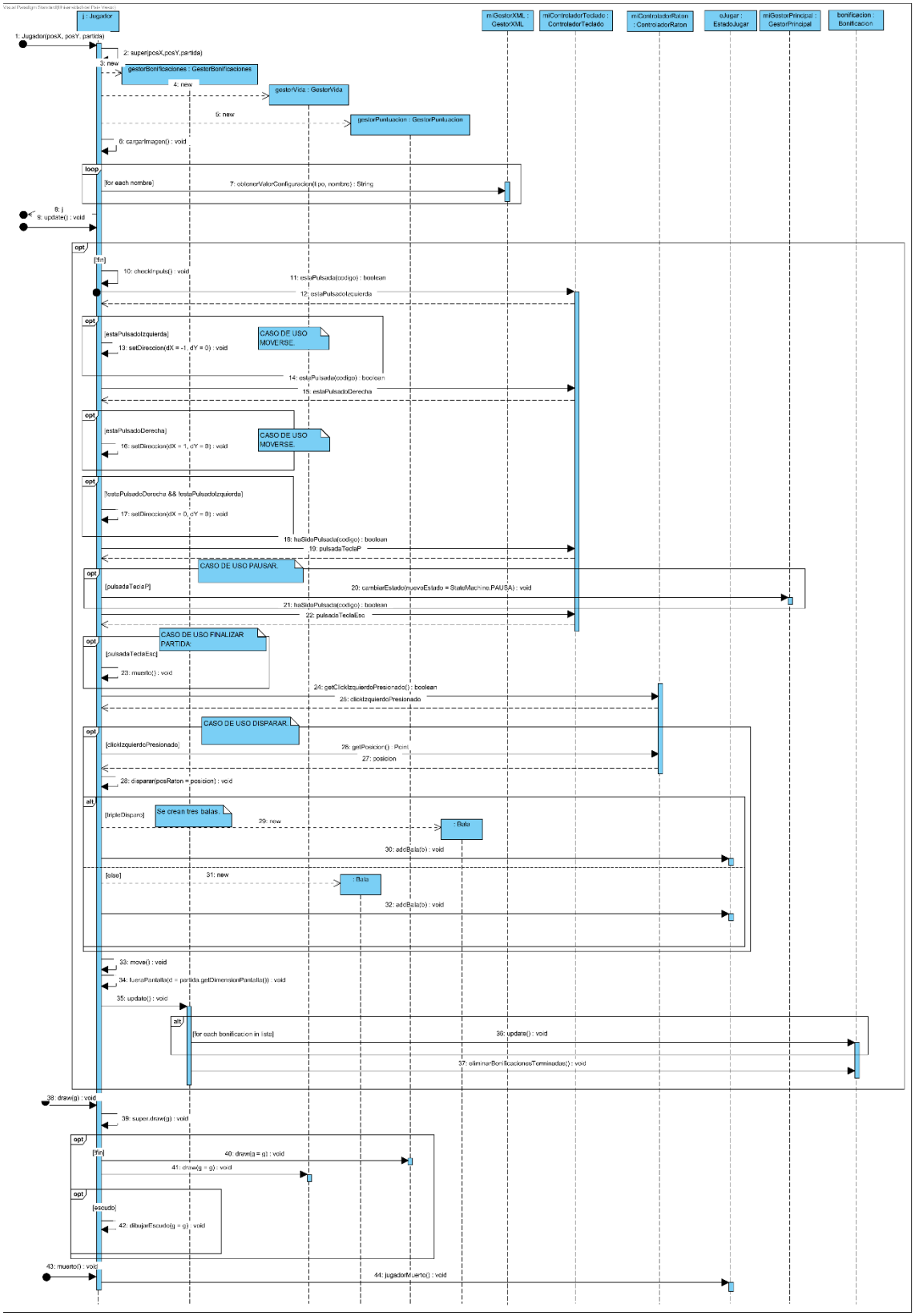



Ilustración 171: SECUENCIA JUGADOR

## SistemaSpawn

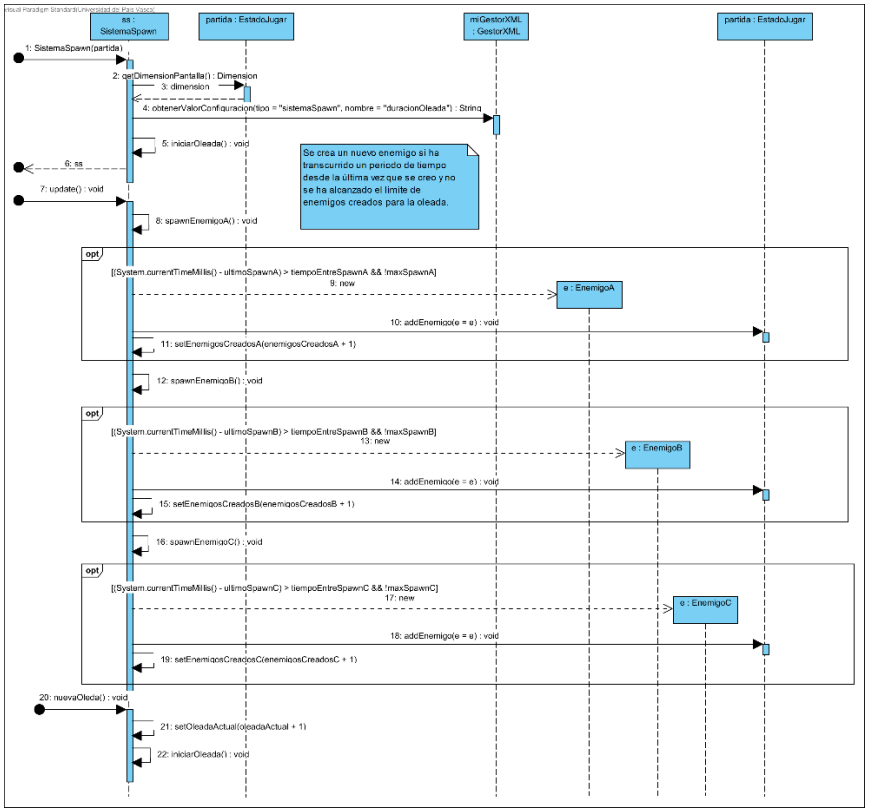



Ilustración 172: SECUENCIA SISTEMA SPAWN

## Explosion

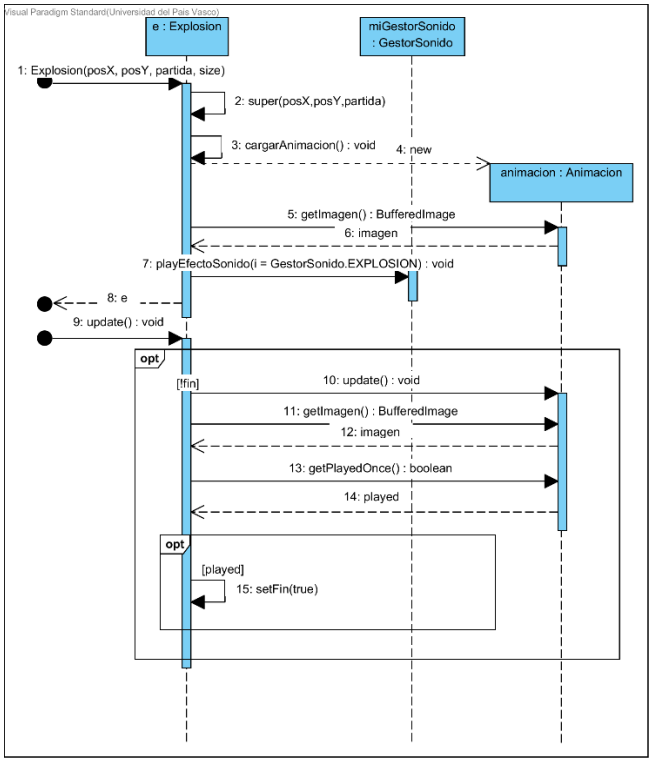



Ilustración 173: SECUENCIA EXPLOSIÓN

## Enemigo

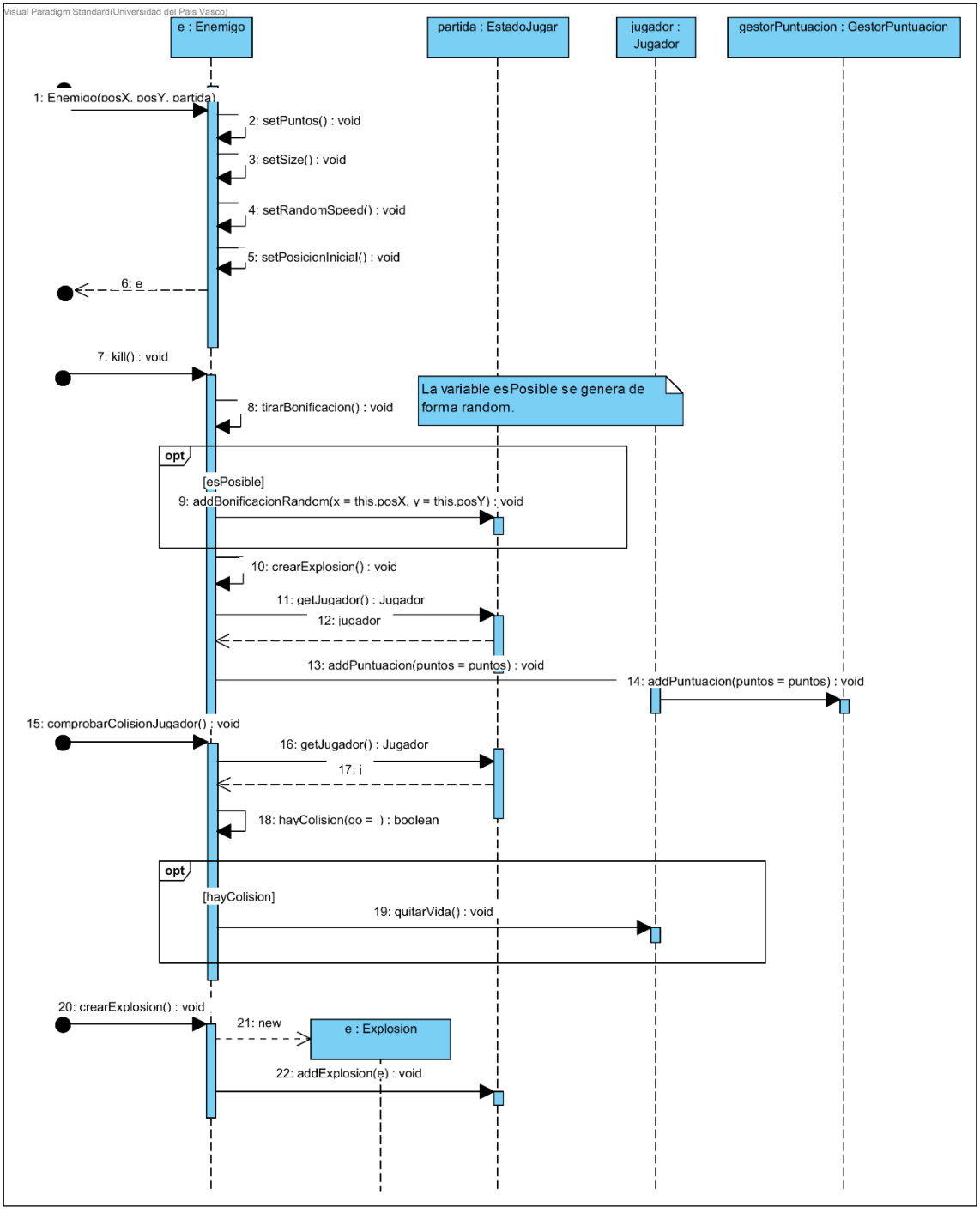



Ilustración 174: SECUENCIA ENEMIGO

## EnemigoA

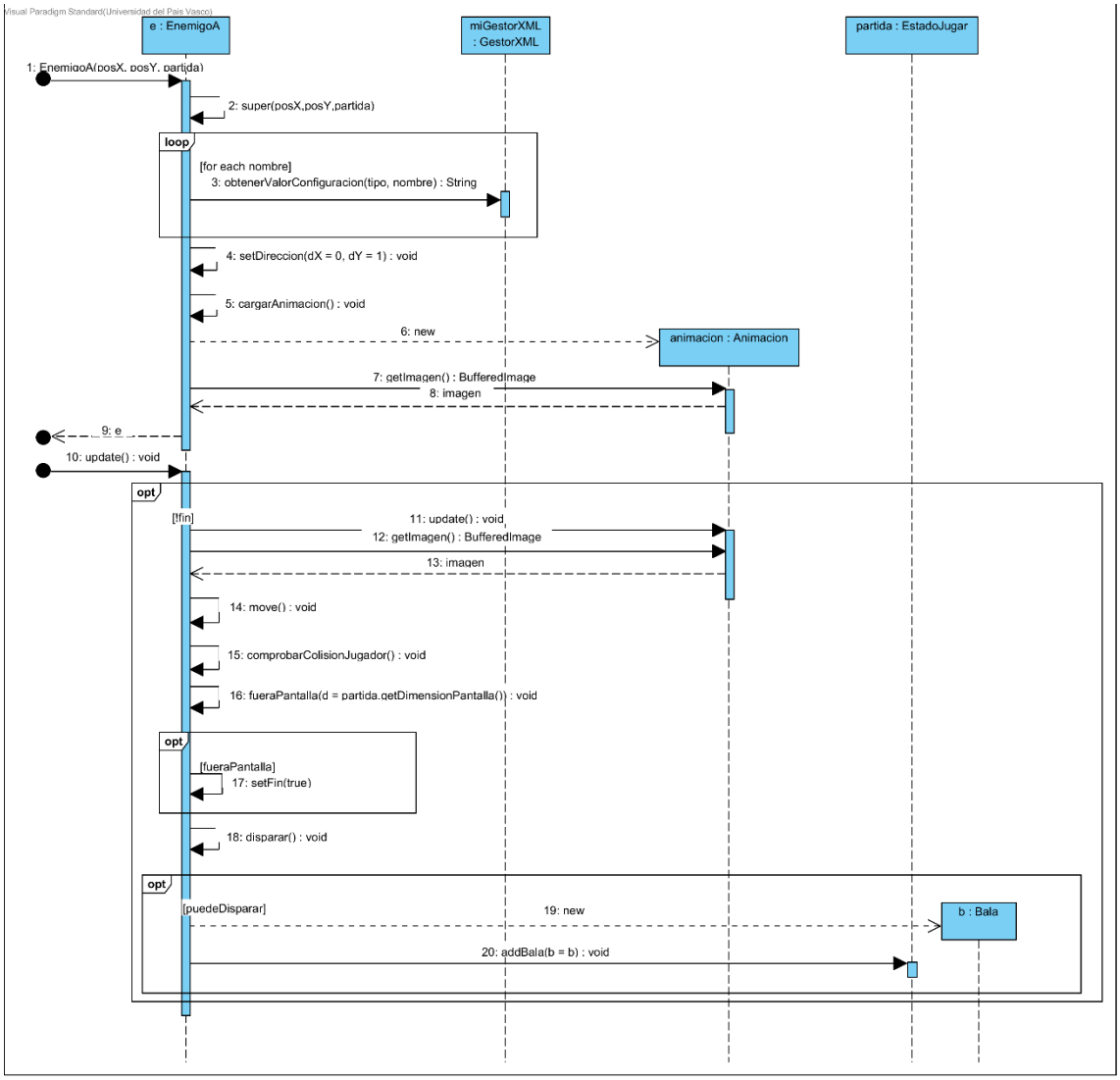



Ilustración 175: SECUENCIA ENEMIGO A

## EnemigoB

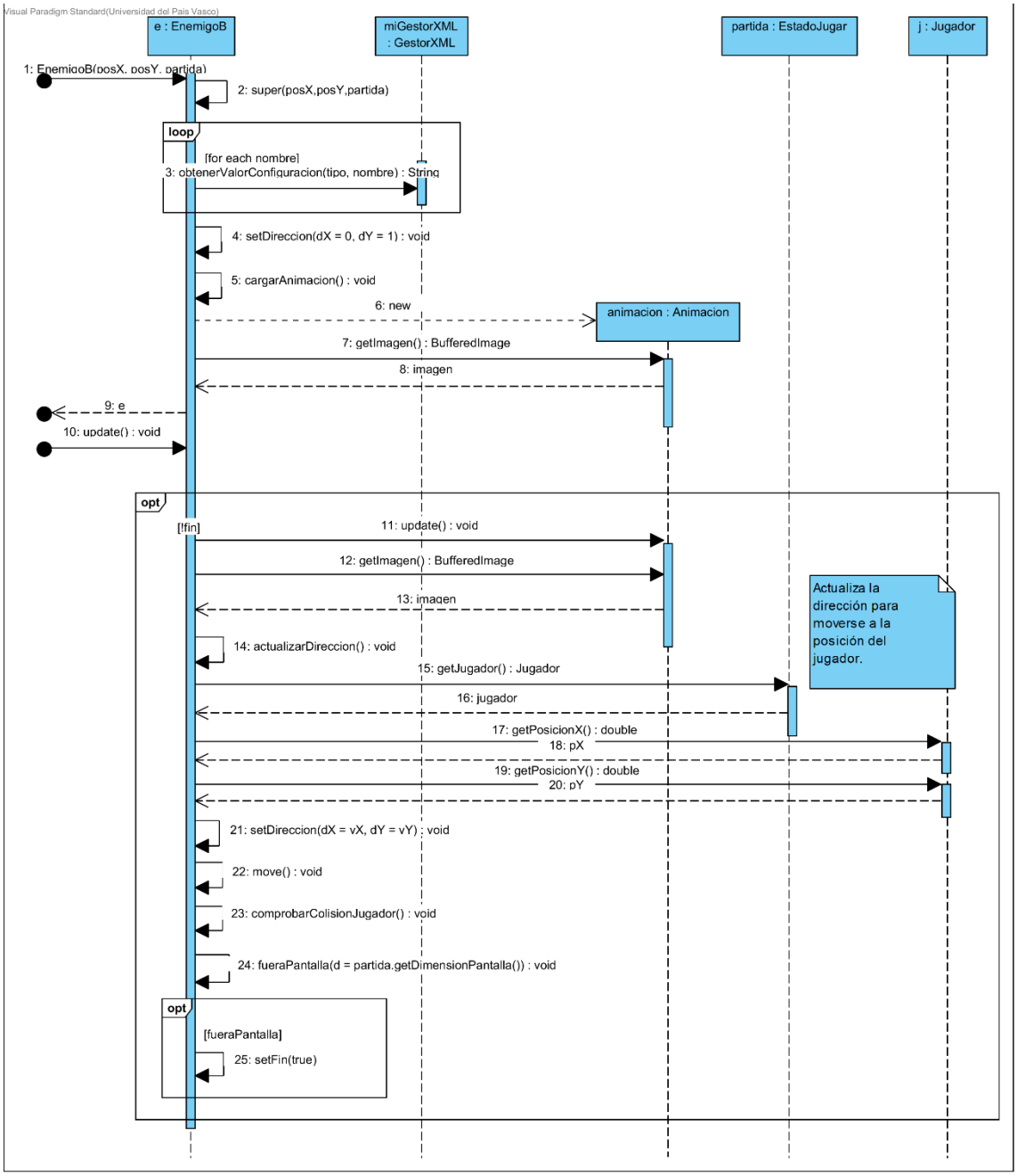



Ilustración 176: SECUENCIA ENEMIGO B

## EnemigoC

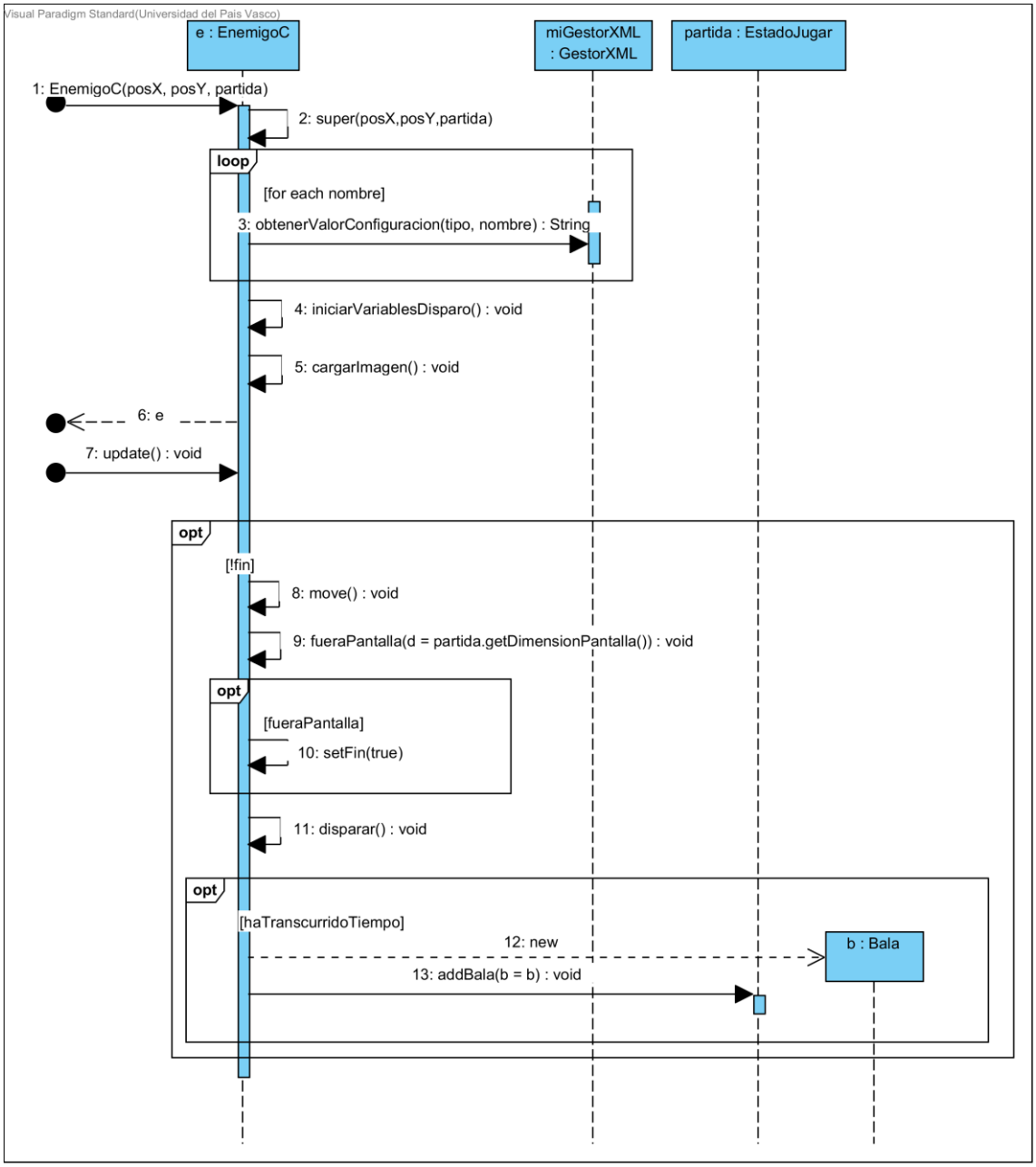



Ilustración 177: SECUENCIA ENEMIGO C

# Bala

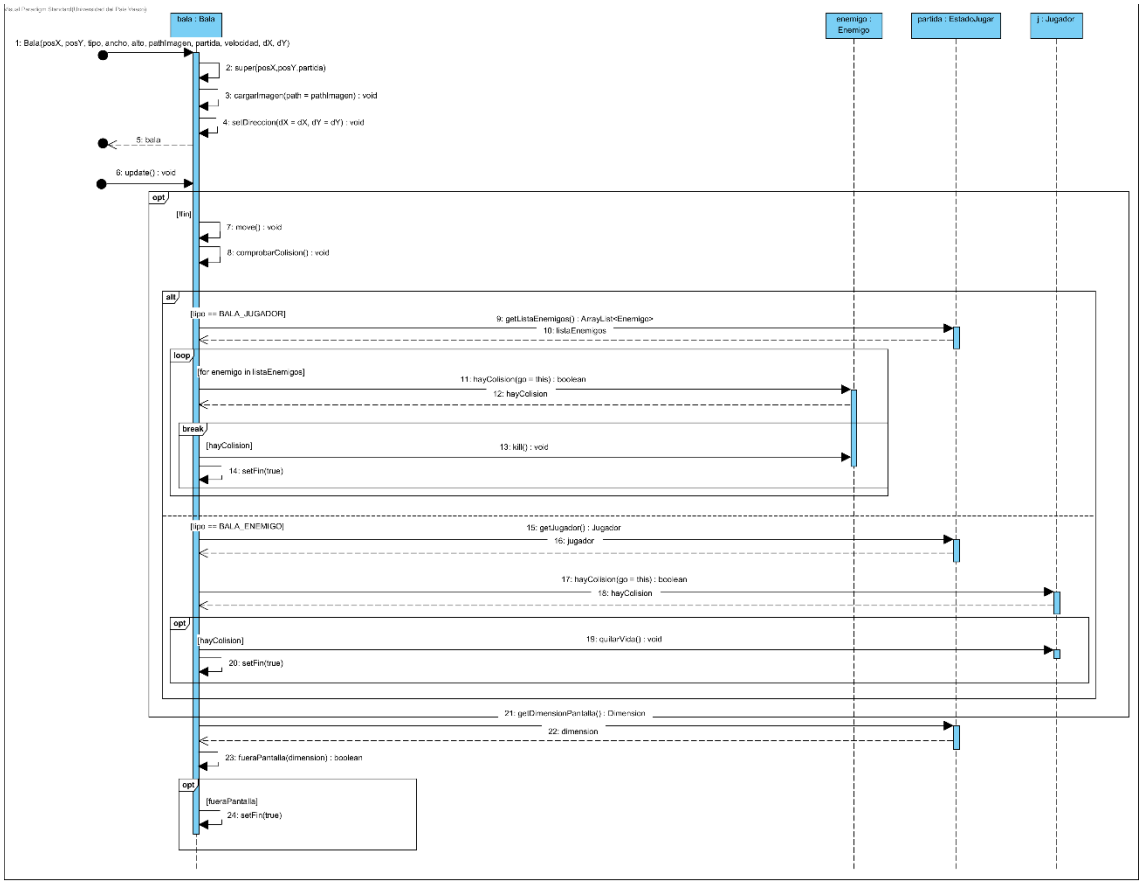



Ilustración 178: SECUENCIA BALA

## Bonificacion

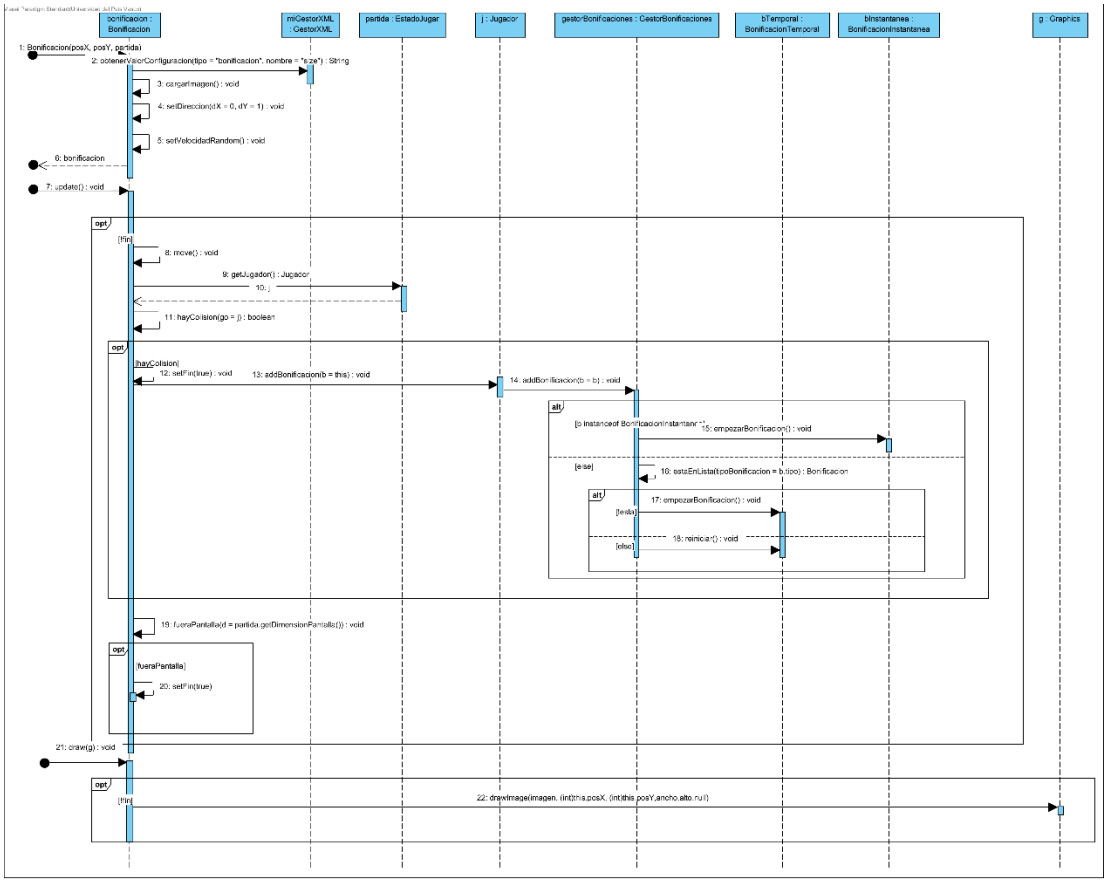



Ilustración 179: SECUENCIA BONIFICACIÓN

## BonificacionTemporal

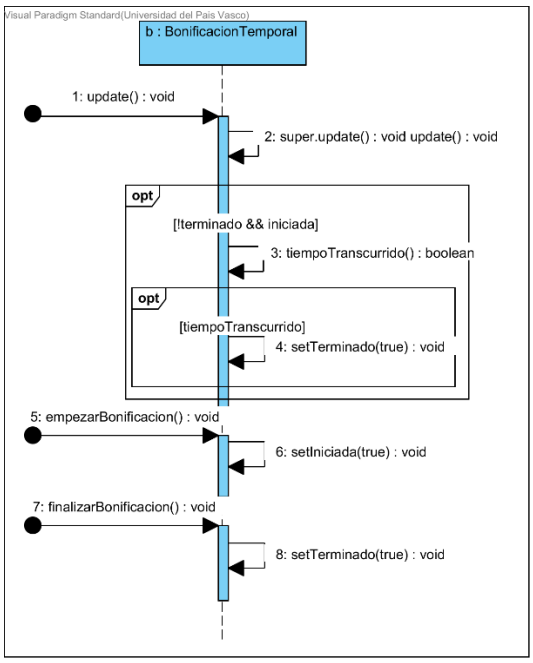



Ilustración 180: SECUENCIA BONIFICACIÓN TEMPORAL

## BonificacionDisparoRapido

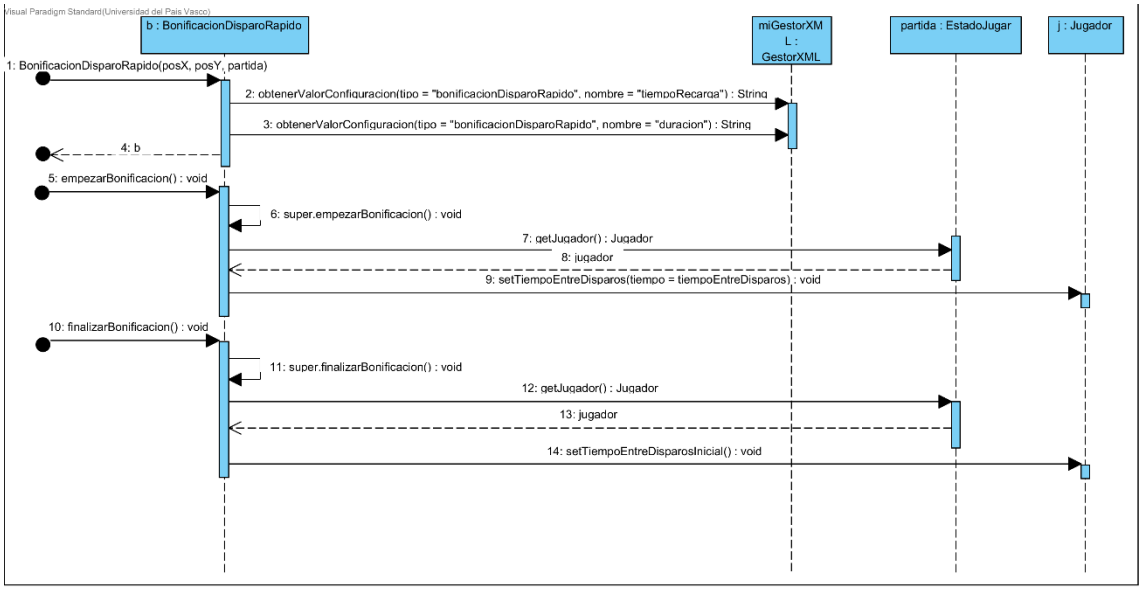



Ilustración 181: SECUENCIA BONIFICACIÓN DISPARO RÁPIDO

## BonificacionEscudo

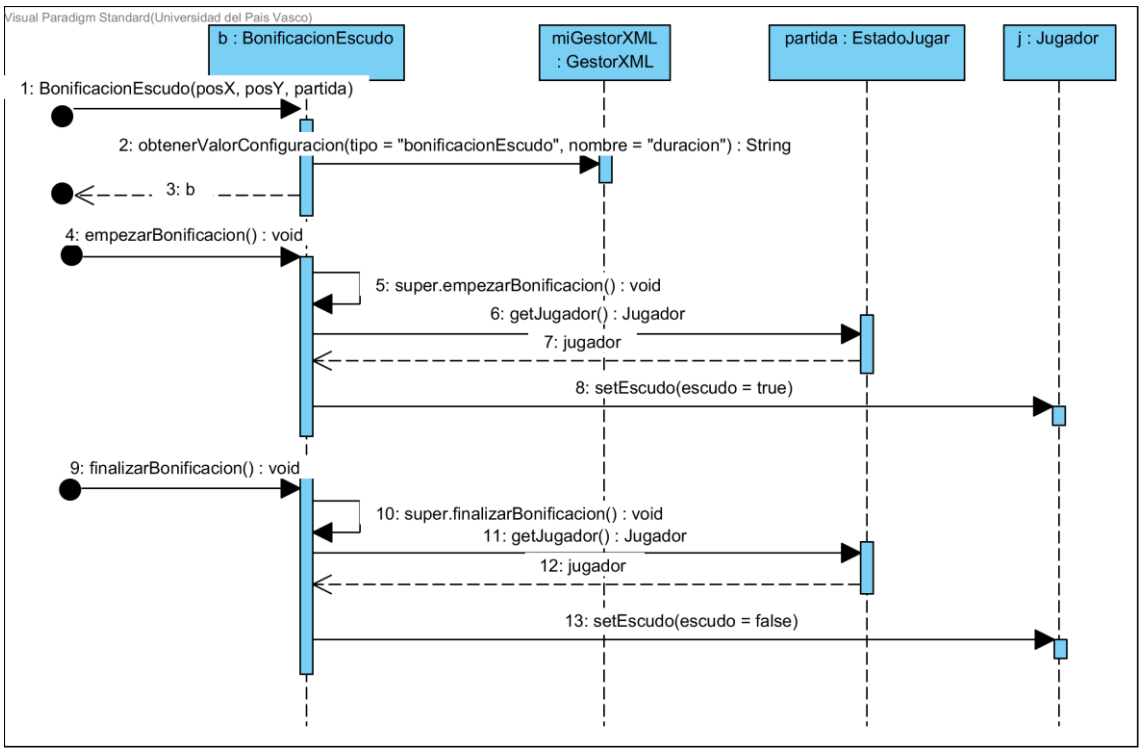



Ilustración 182: SECUENCIA BONIFICACIÓN ESCUDO

## BonificacionTripleDisparo

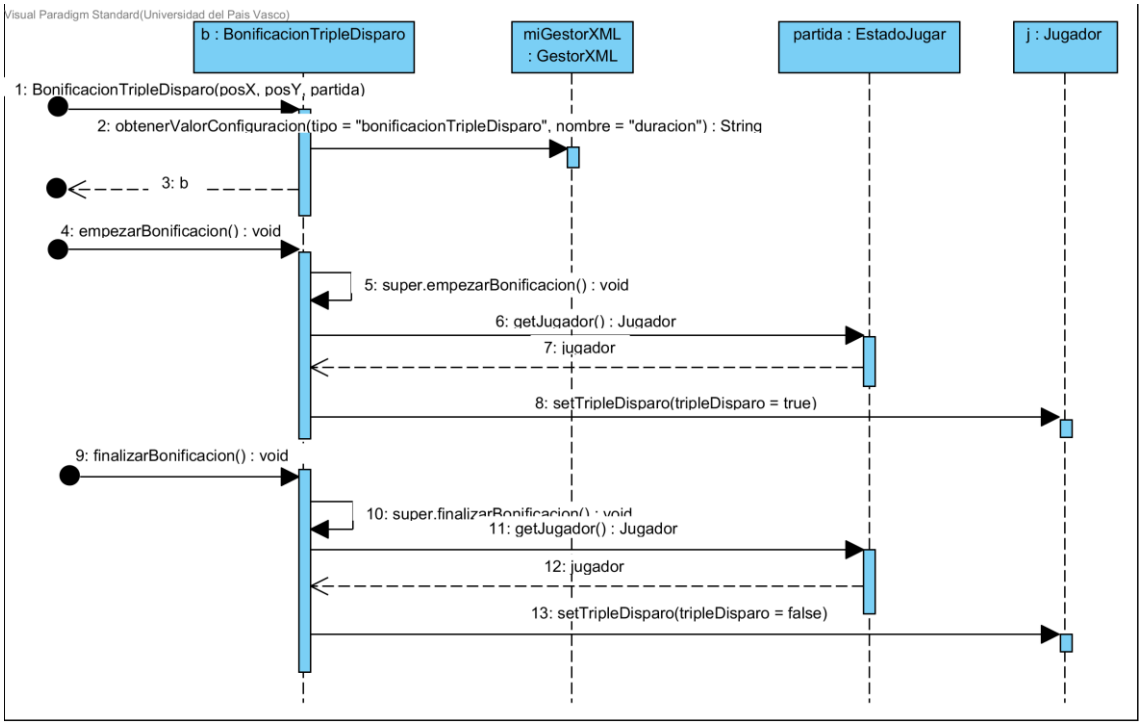



Ilustración 183: SECUENCIA BONIFICACIÓN TRIPLE DISPARO

## BonificacionVida

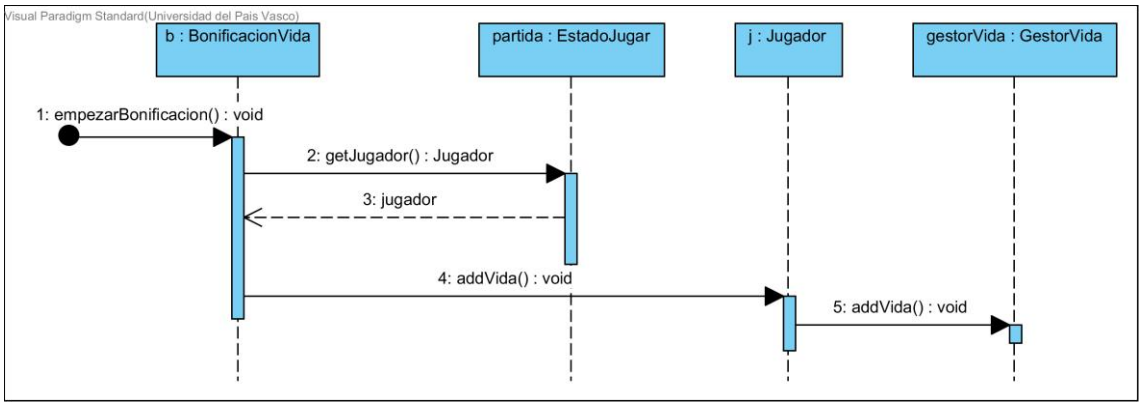



Ilustración 184: SECUENCIA BONIFICACIÓN VIDA

## 11.3. Ping-Ping

En este apartado se muestran los diagramas de secuencia correspondientes al juego Ping-Pong.

En los diagramas correspondientes a las clases utilizadas del motor gráfico se mostrarán las funcionalidades añadidas o modificadas en caso de que se haya añadido o modificado algo.

Los casos de uso se identificarán mediante notas en el diagrama. El caso de uso Gestionar Sonido se gestiona en la clase IconoSonido del motor gráfico.

Como el juego cuenta tanto con una aplicación cliente como servidor se han dividido los diagramas en dos apartados.

### 11.3.1. Cliente

#### StateMachine

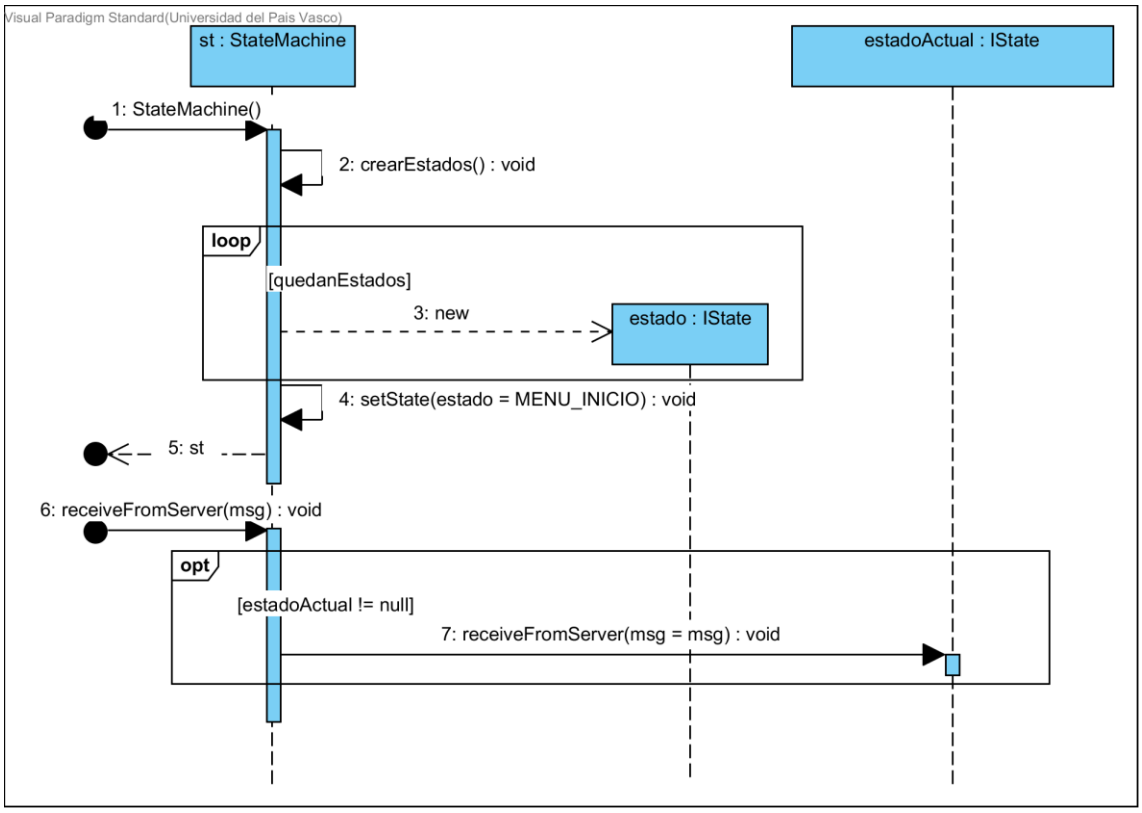



Ilustración 185: SECUENCIA STATE MACHINE

## GestorPrincipal

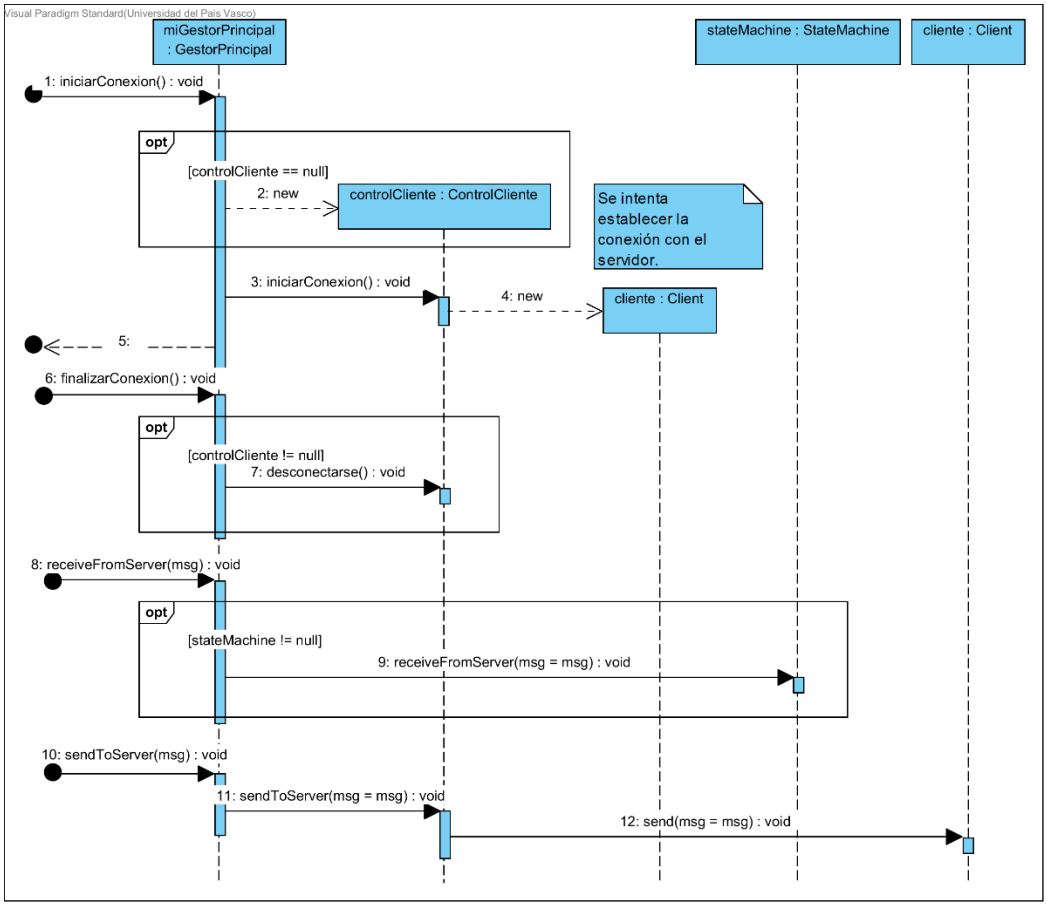



Ilustración 186: SECUENCIA GESTOR PRINCIPAL

## GestorUsuarios

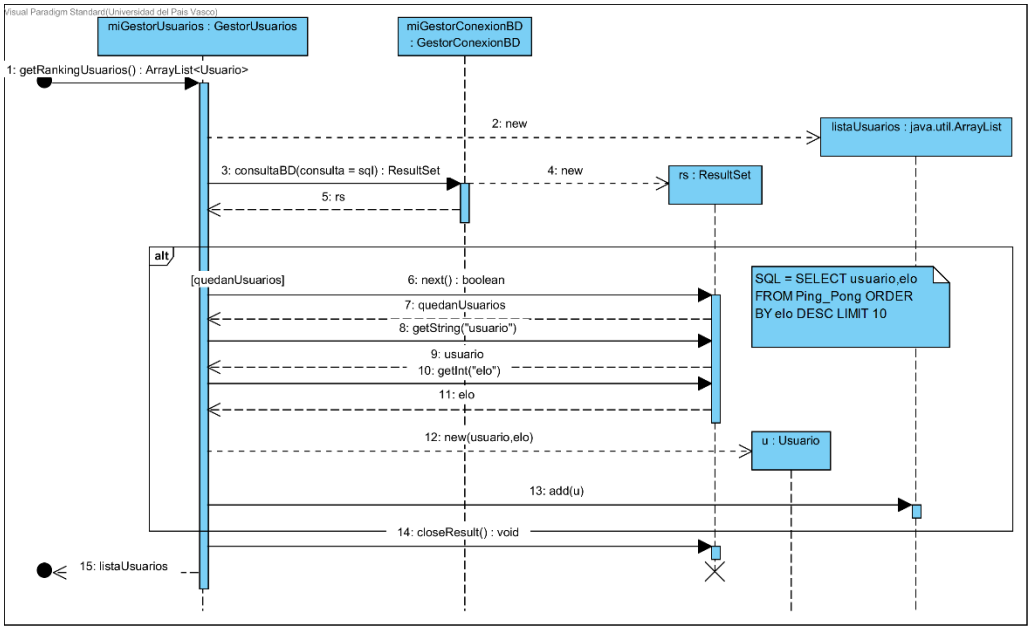



Ilustración 187: SECUENCIA GESTOR USUARIOS

## Usuario

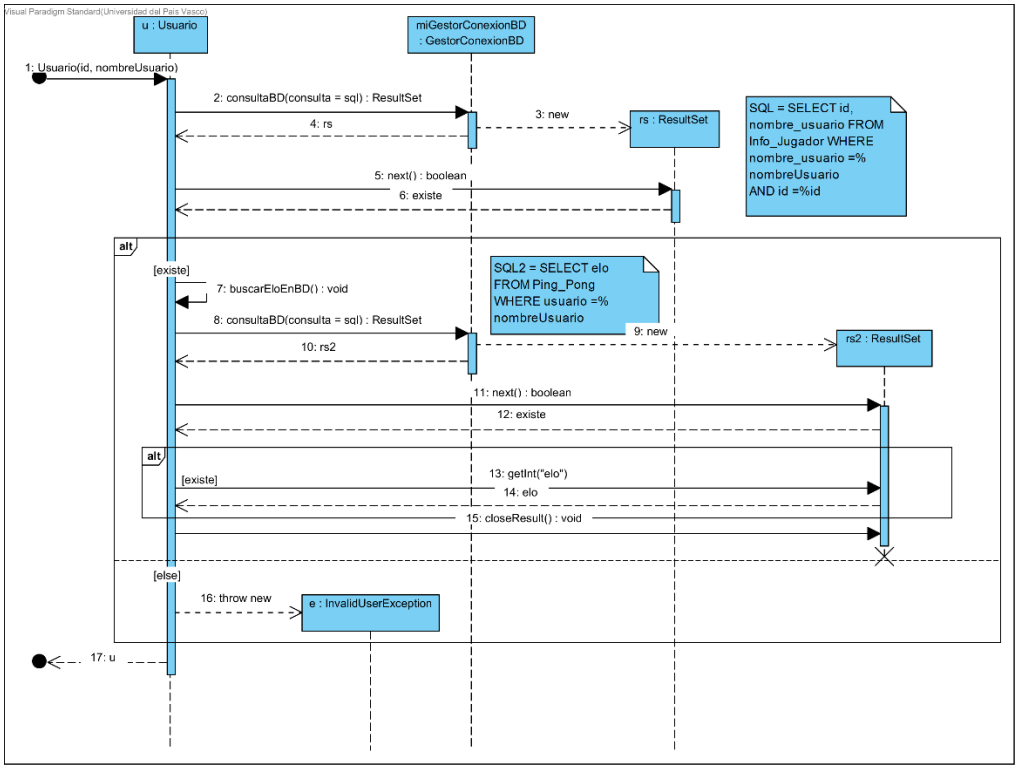



Ilustración 188: SECUENCIA USUARIO

## ControlCliente

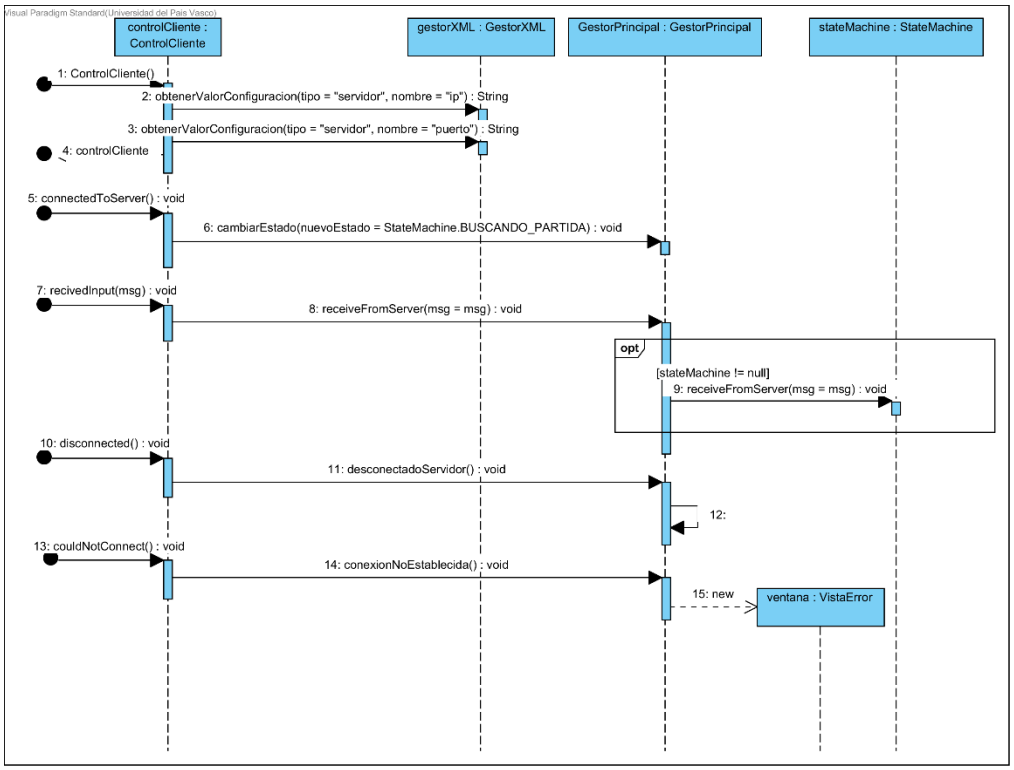



Ilustración 189: SECUENCIA CONTROL CLIENTE

## EstadoMenuInicio

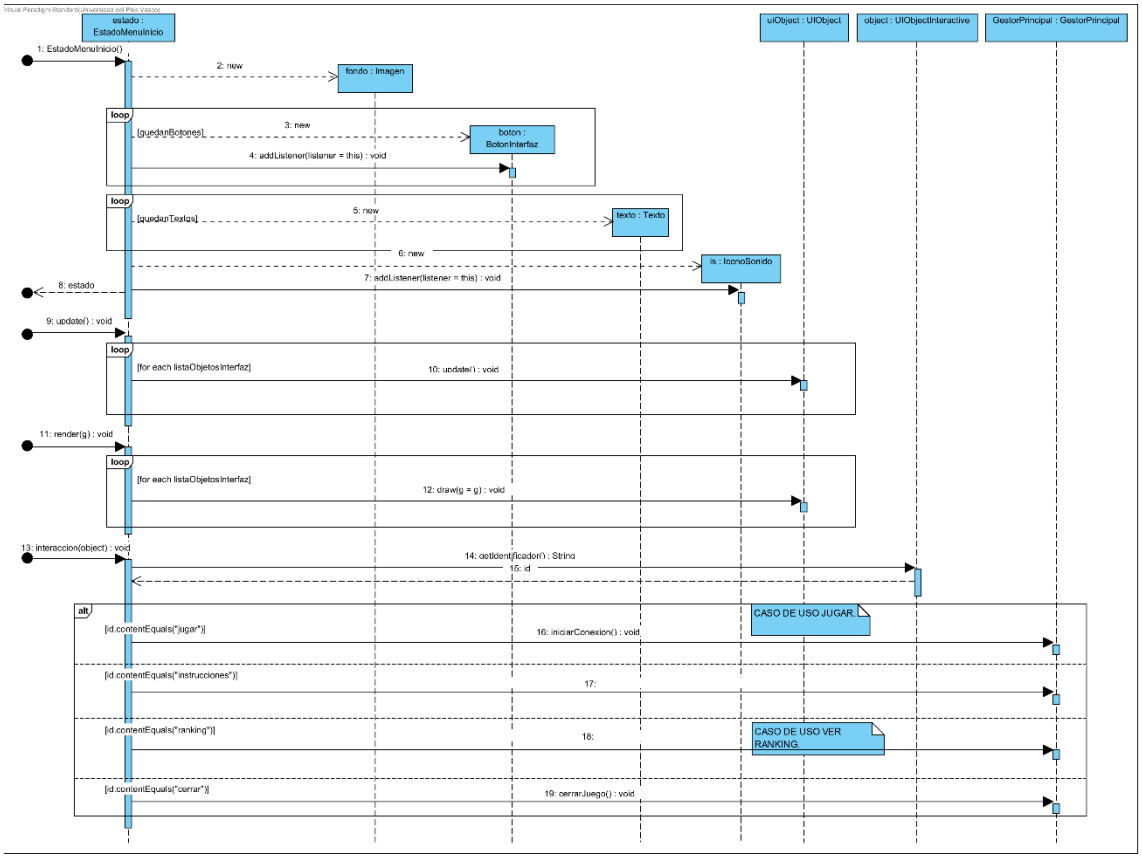



Ilustración 190: SECUENCIA ESTADO MENÚ INICIO

## EstadoVerRanking

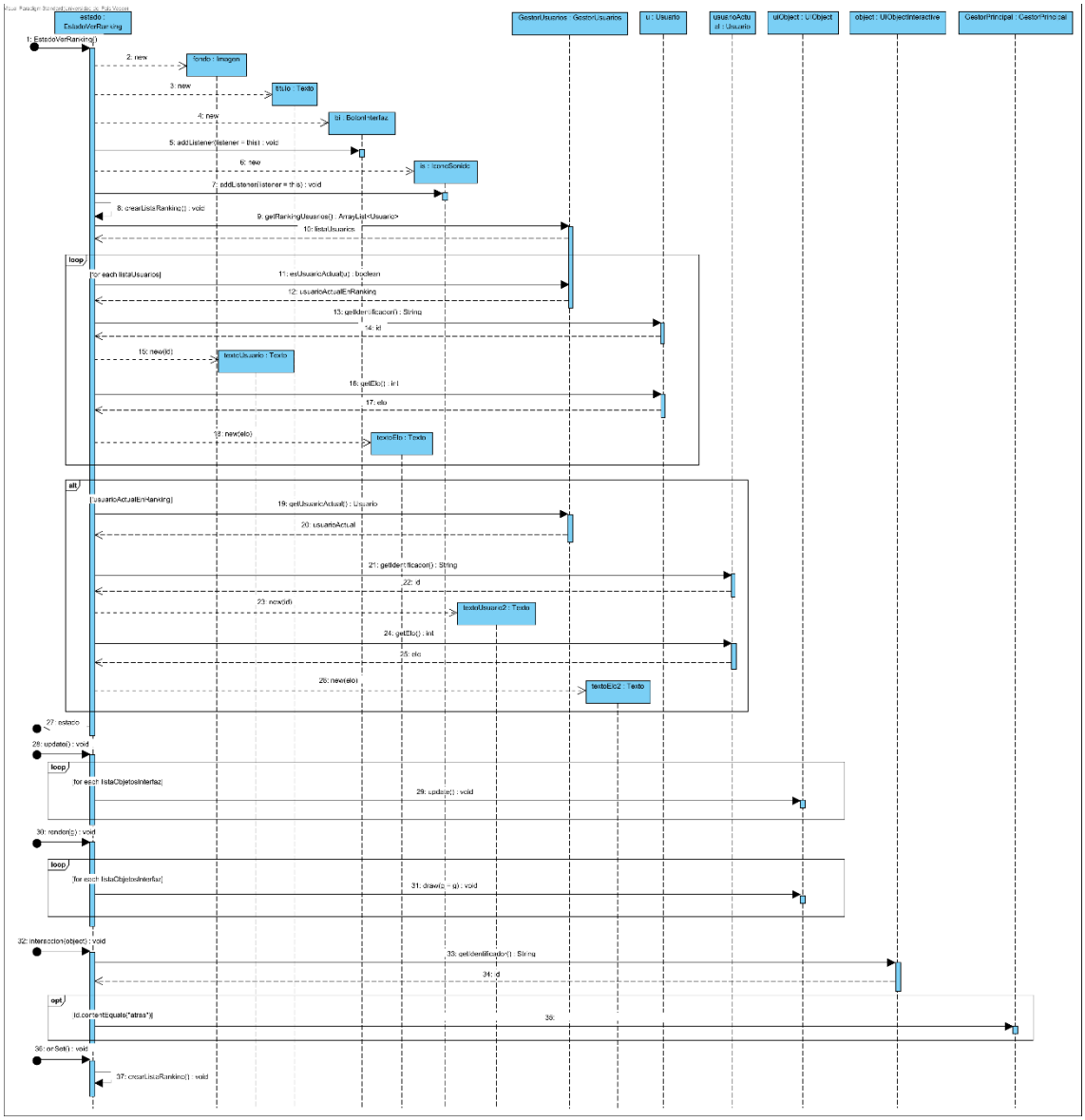



Ilustración 191: SECUENCIA ESTADO VER RANKING

## EstadoInstrucciones

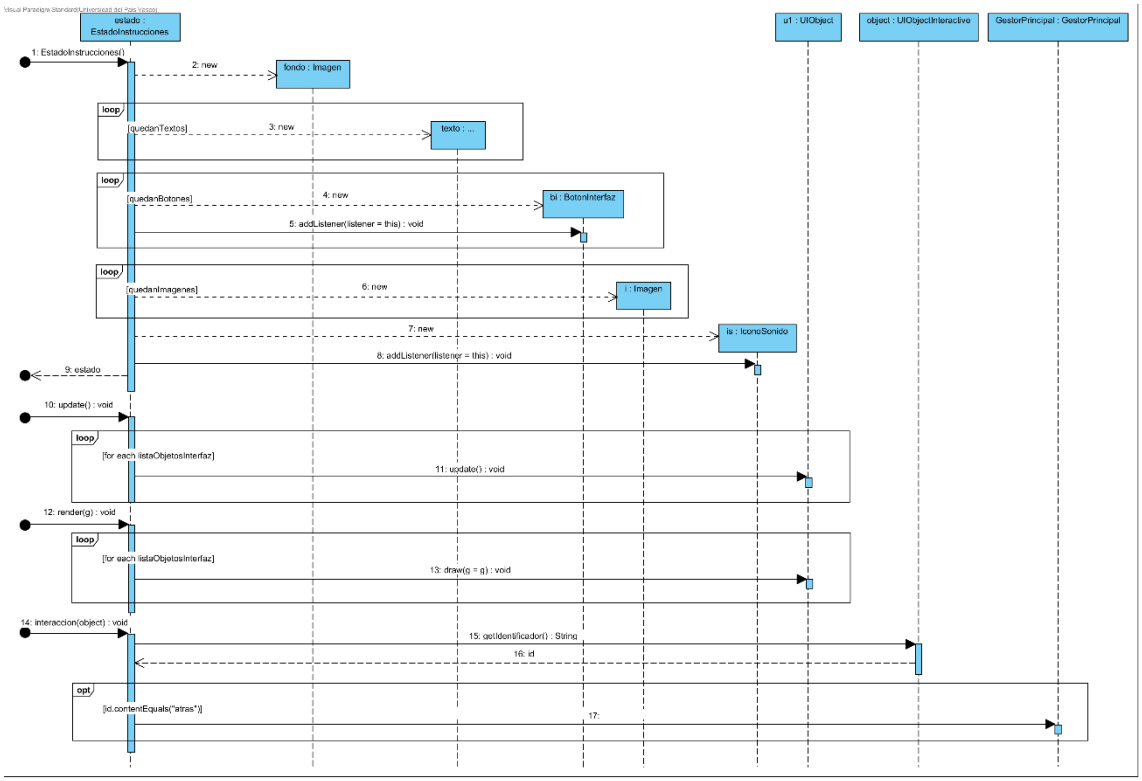



Ilustración 192: SECUENCIA ESTADO INSTRUCCIONES

## EstadoBuscandoPartida

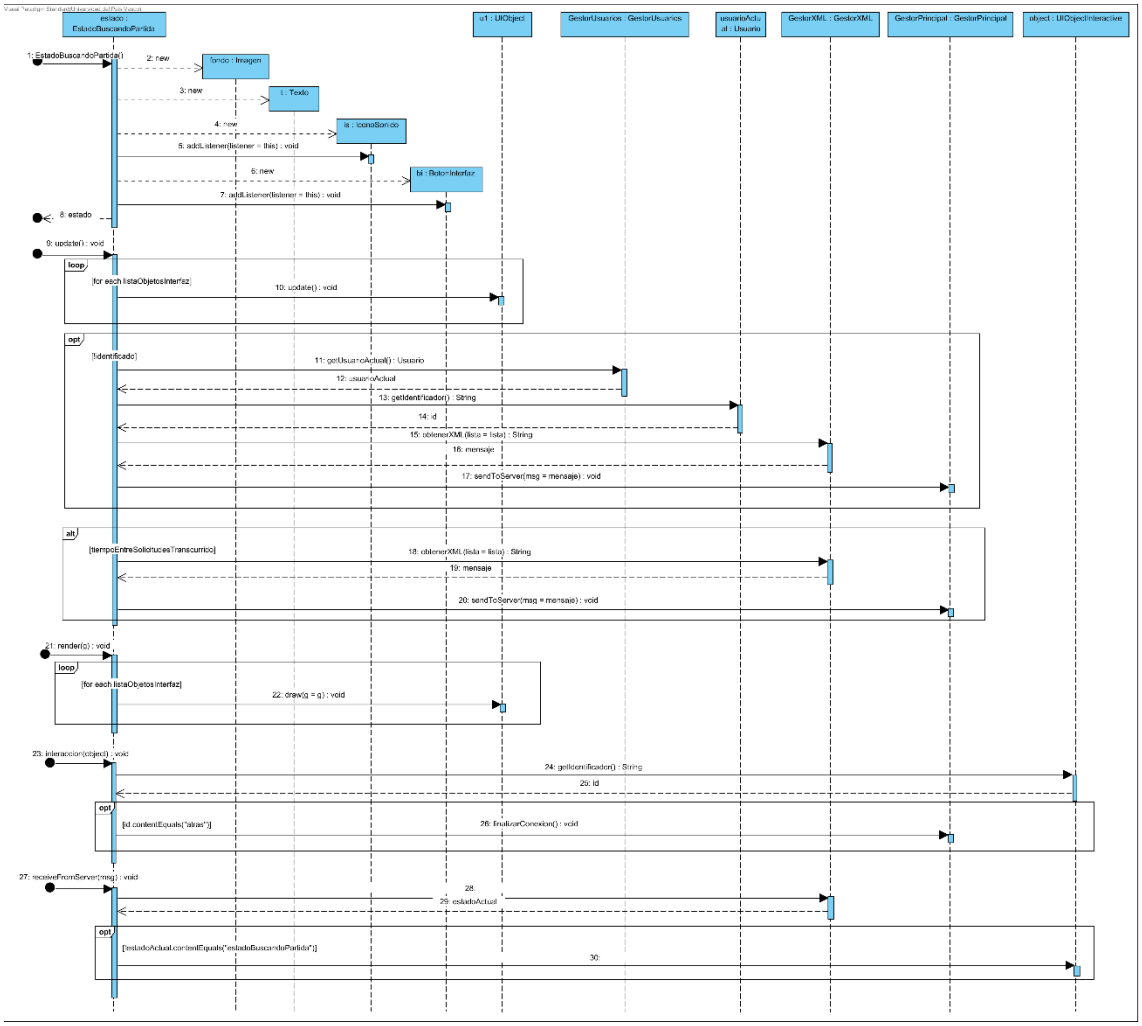



Ilustración 193: SECUENCIA ESTADO BUSCANDO PARTIDA

## EstadoiniciandoPartida

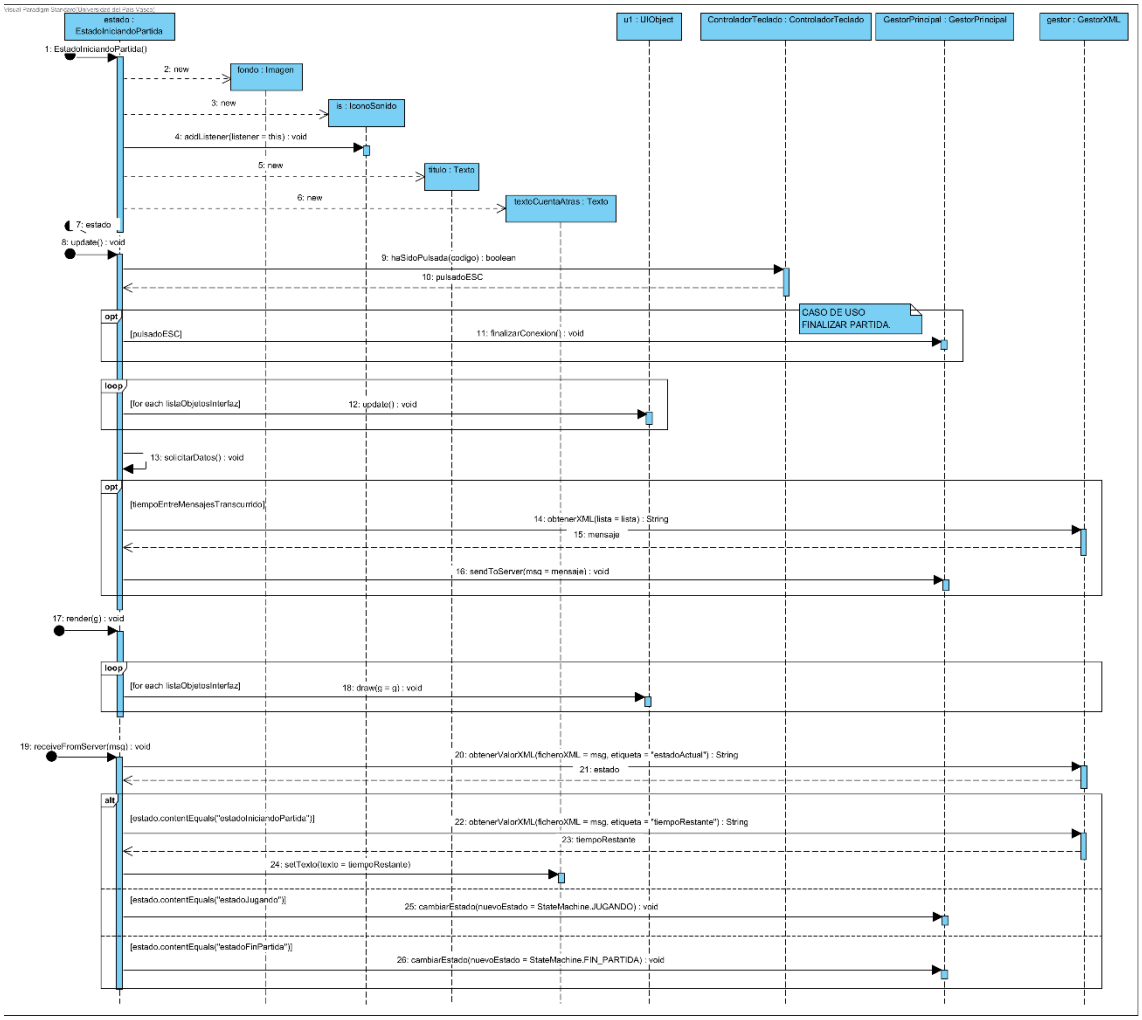



**Ilustración 194: SECUENCIA ESTADO INICIANDO PARTIDA**

# EstadoJugando

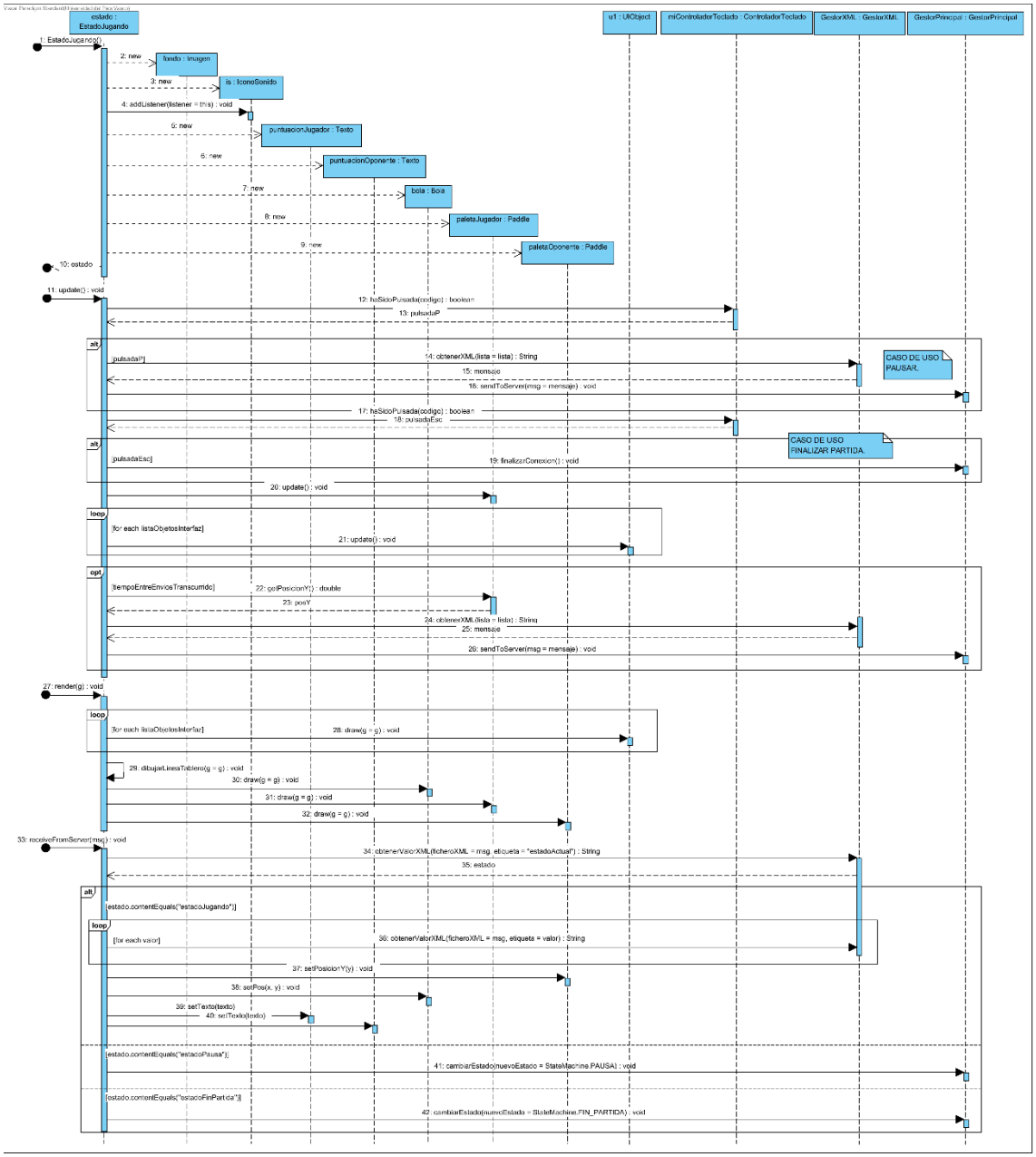



Ilustración 195: SECUENCIA ESTADO JUGANDO

## EstadoPausa

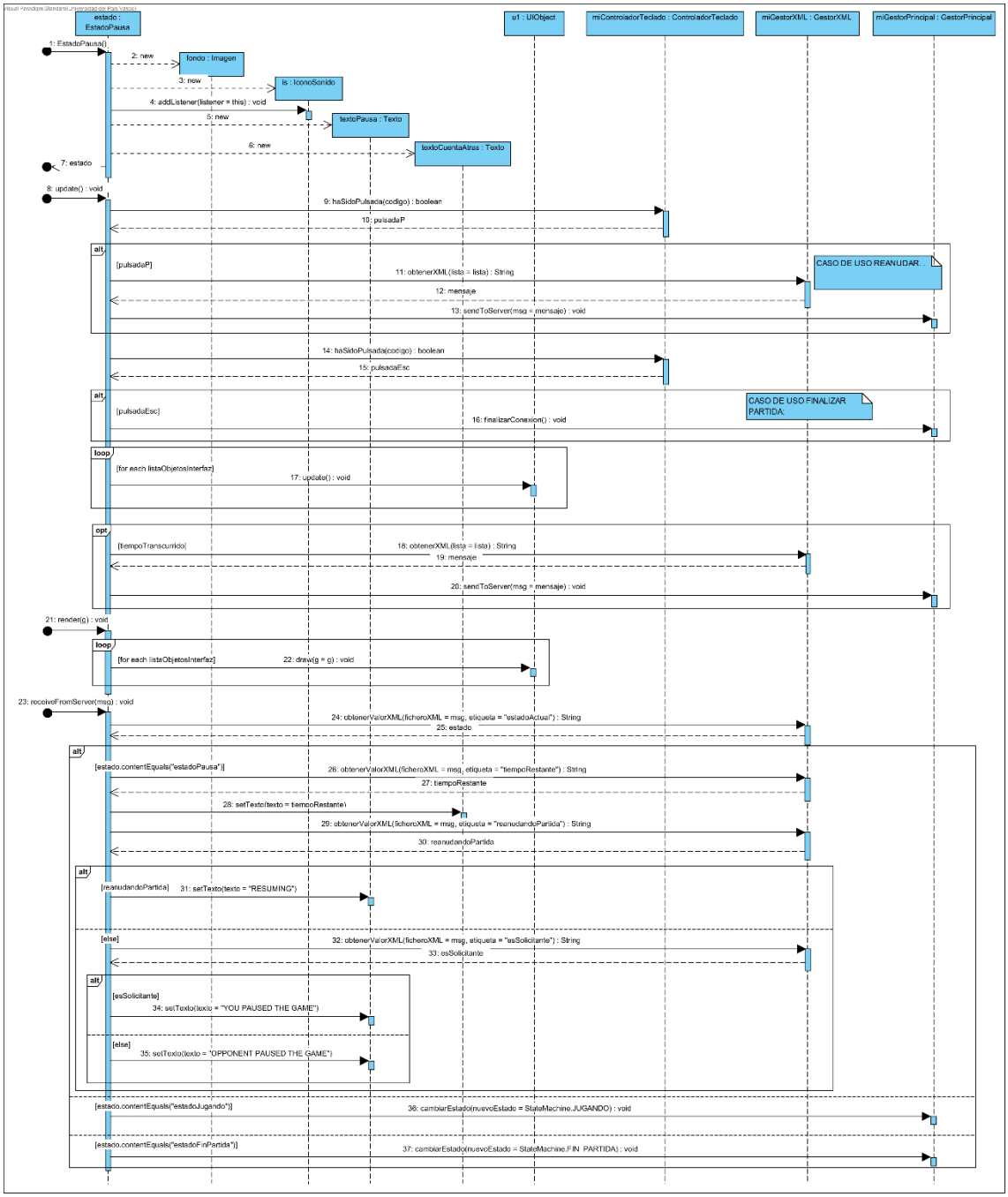



Ilustración 196: SECUENCIA ESTADO PAUSA

## EstadoFinPartida

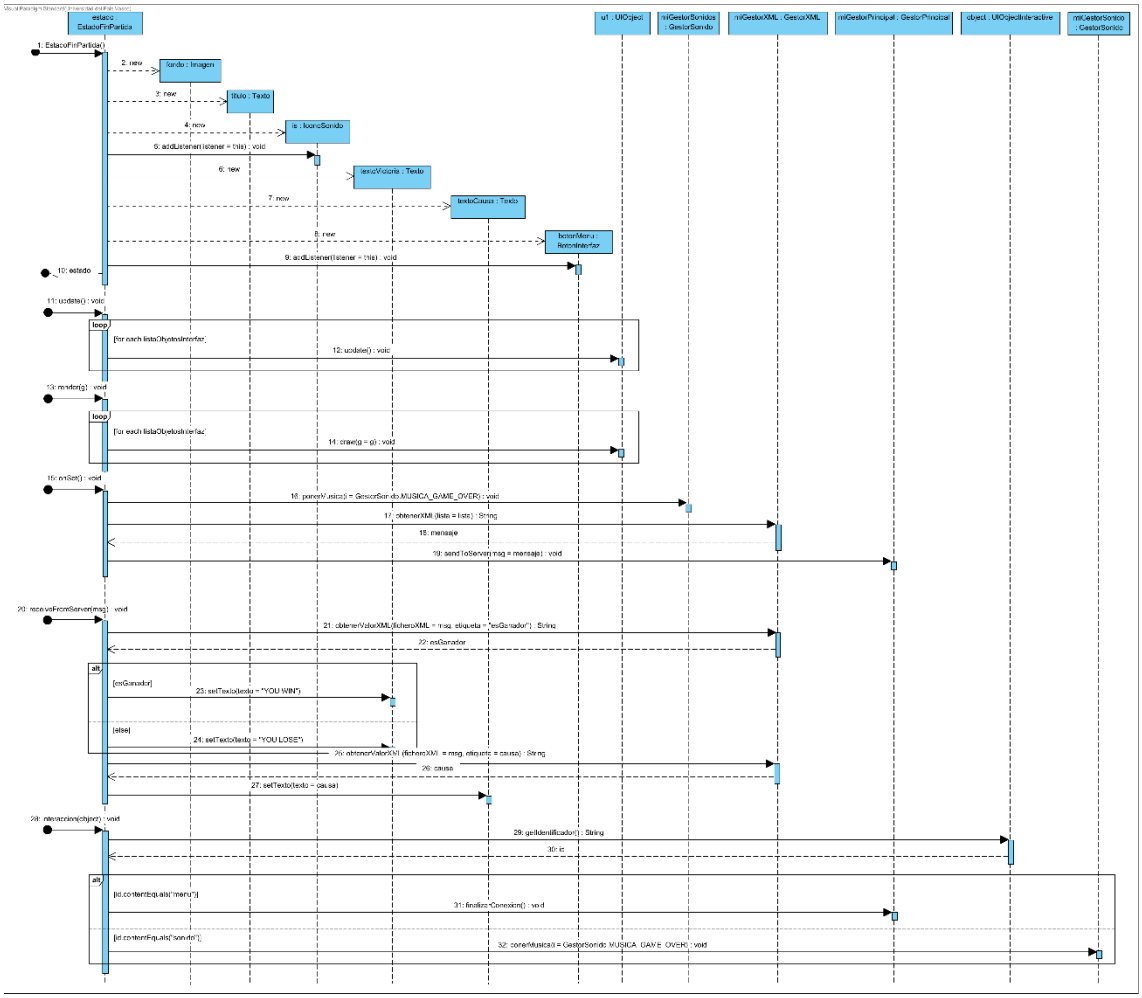


Ilustración 197: SECUENCIA ESTADO FIN PARTIDA

## Paddle

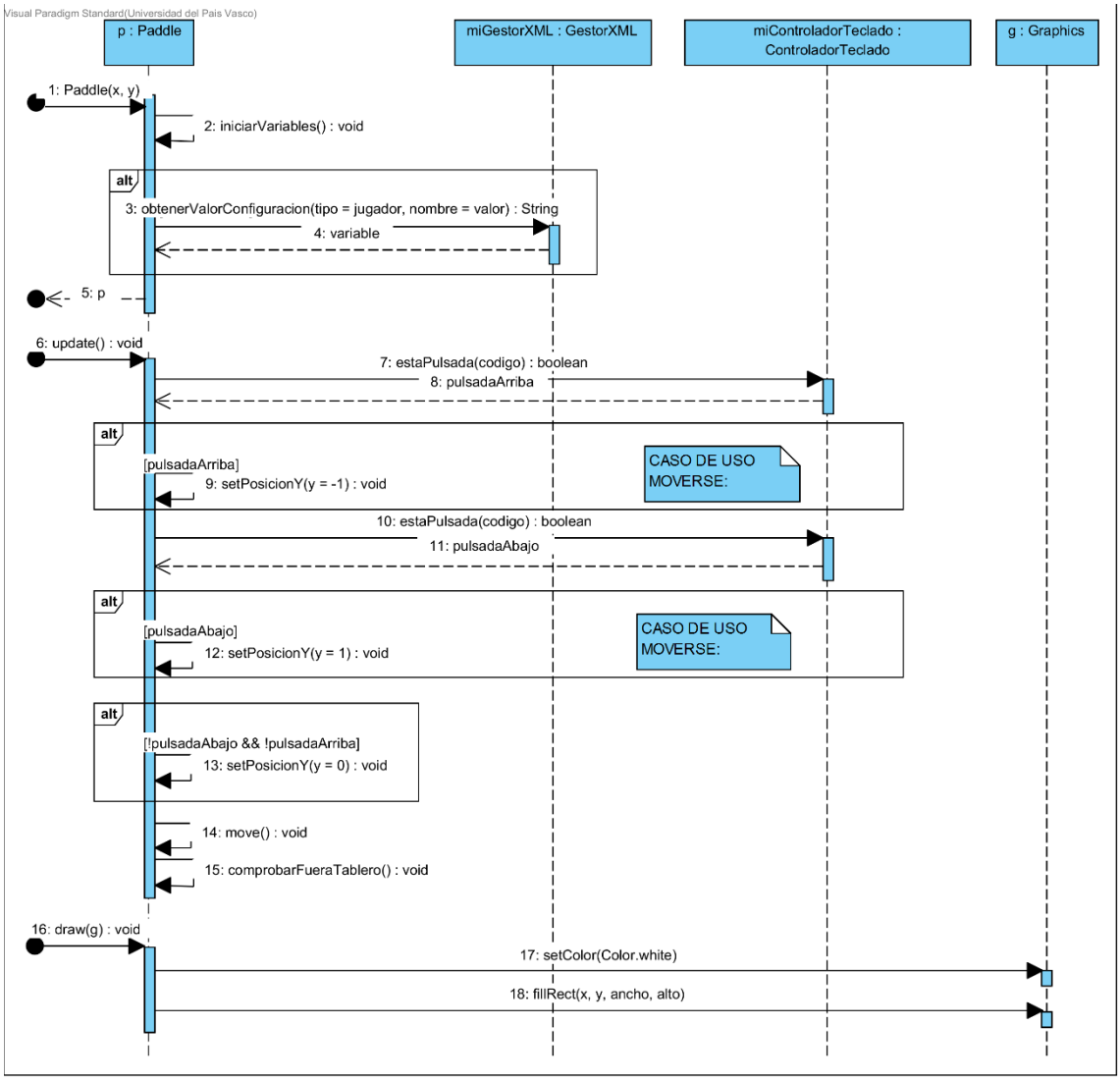



Ilustración 198: SECUENCIA PADDLE

### 11.3.2. Servidor

## Iniciar servidor

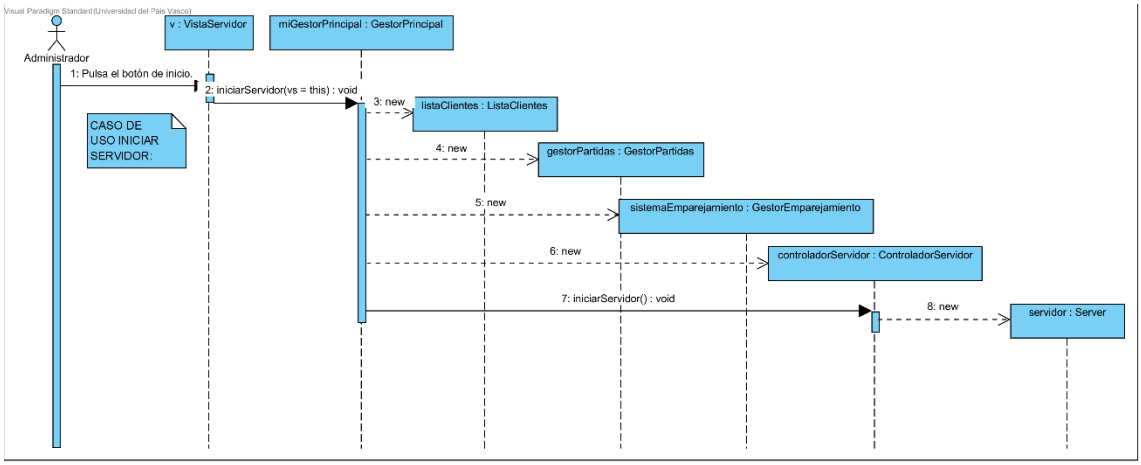



Ilustración 199: SECUENCIA INICIAR SERVIDOR

## Cliente conectado

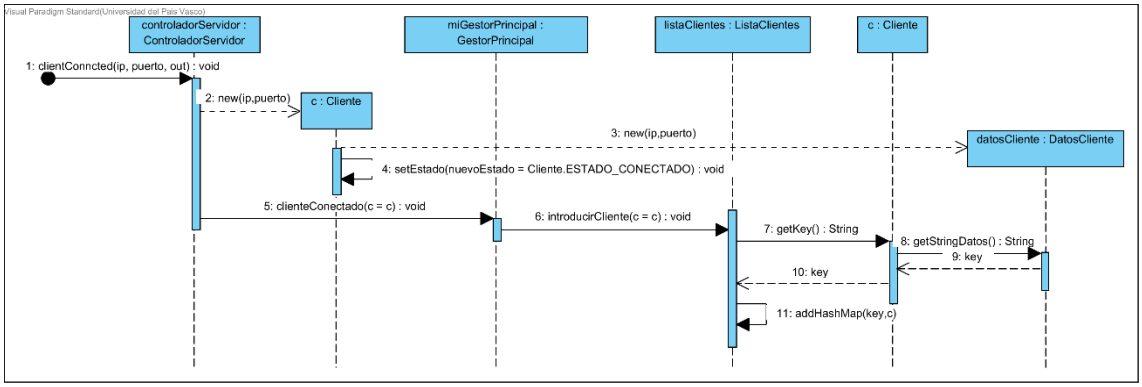



Ilustración 200: SECUENCIA CLIENTE CONECTADO

## Cliente desconectado

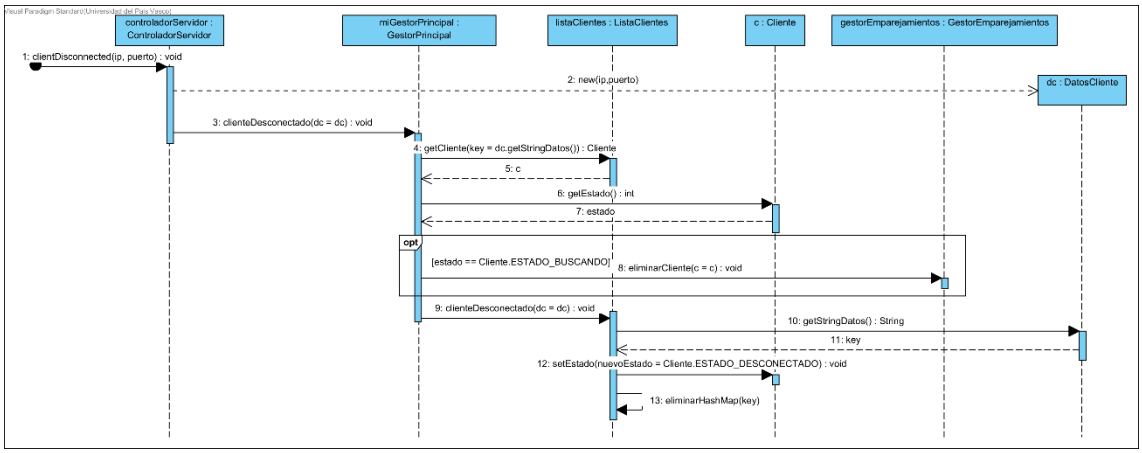



Ilustración 201: SECUENCIA CLIENTE DESCONECTADO

## Mensaje cliente

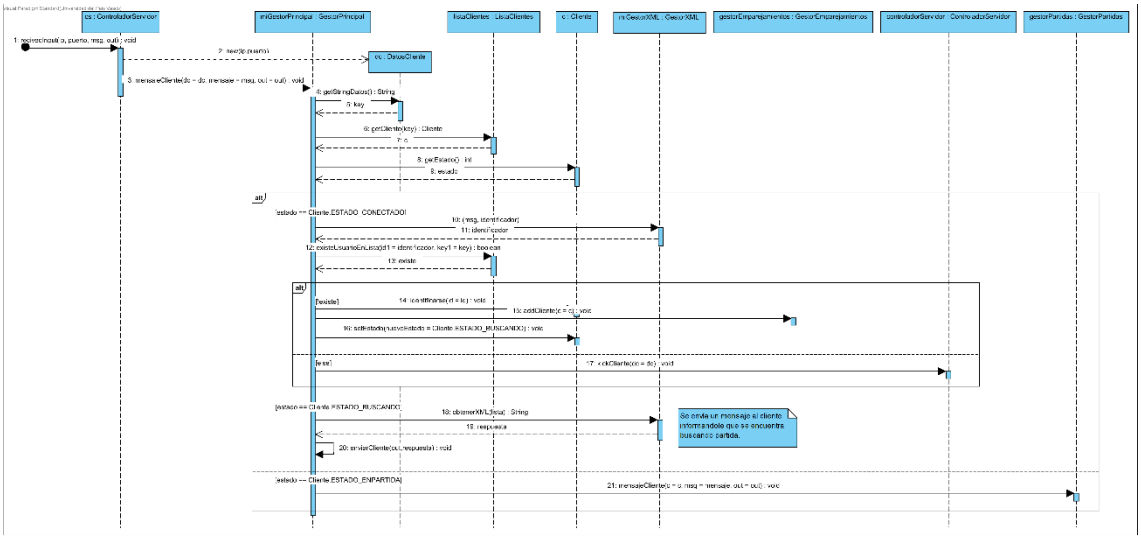


Ilustración 202: SECUENCIA MENSAJE CLIENTE

## Cliente

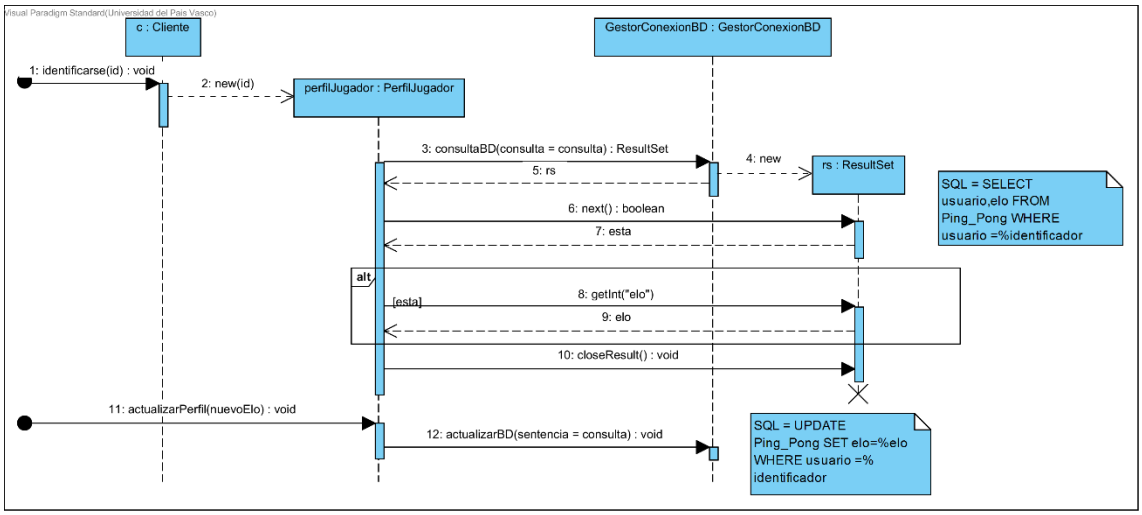



Ilustración 203: SECUENCIA CLIENTE

## GestorEmparejamientos

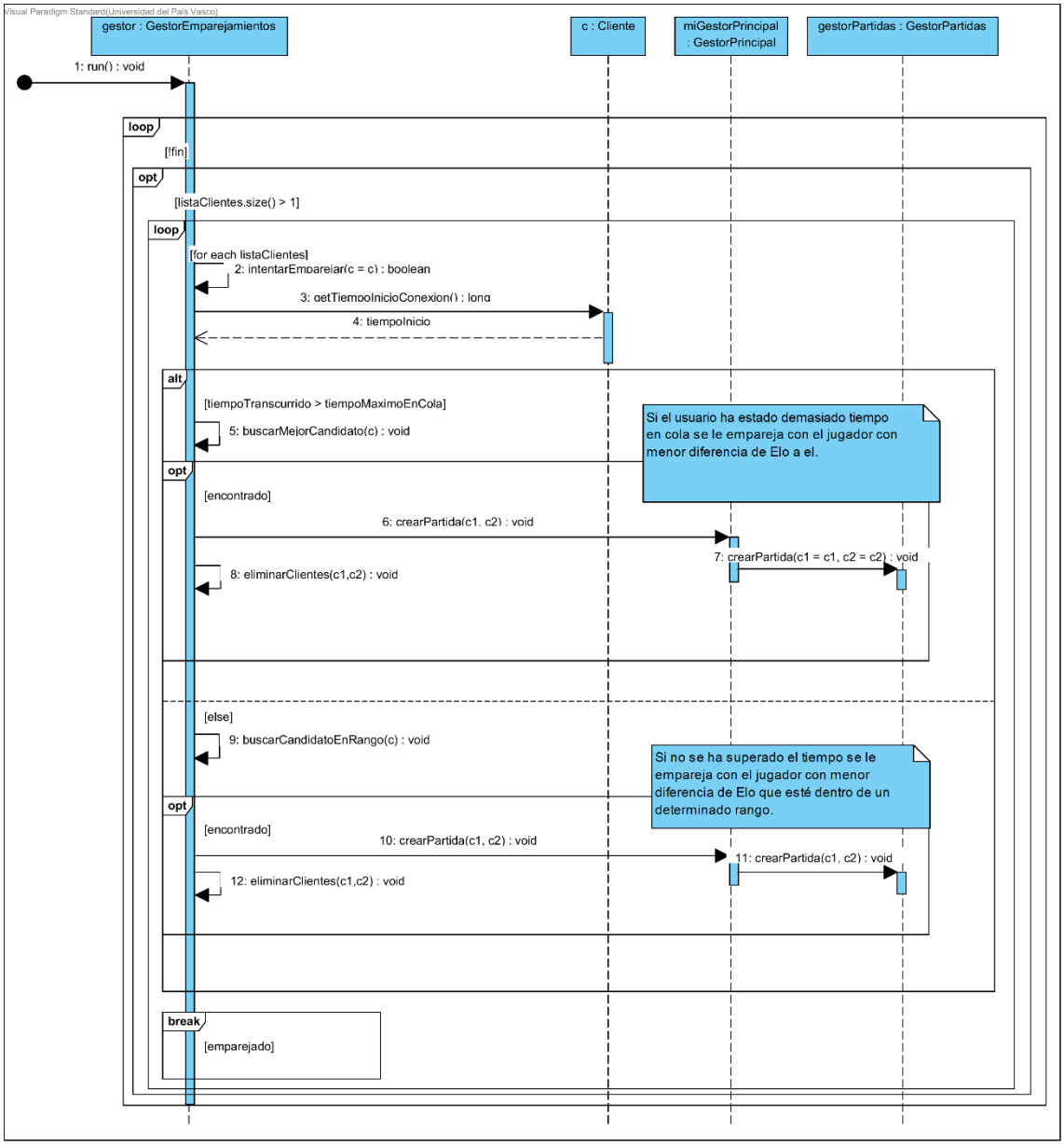



Ilustración 204: SECUENCIA GESTOR EMPAREJAMIENTOS

## GestorPartidas

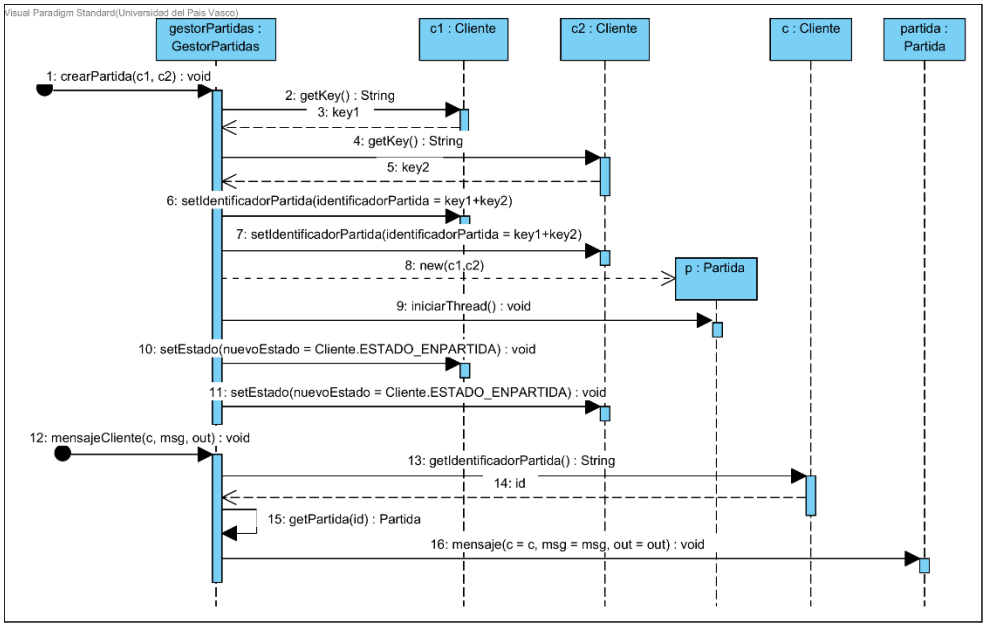



Ilustración 205: SECUENCIA GESTOR PARTIDAS

## Partida

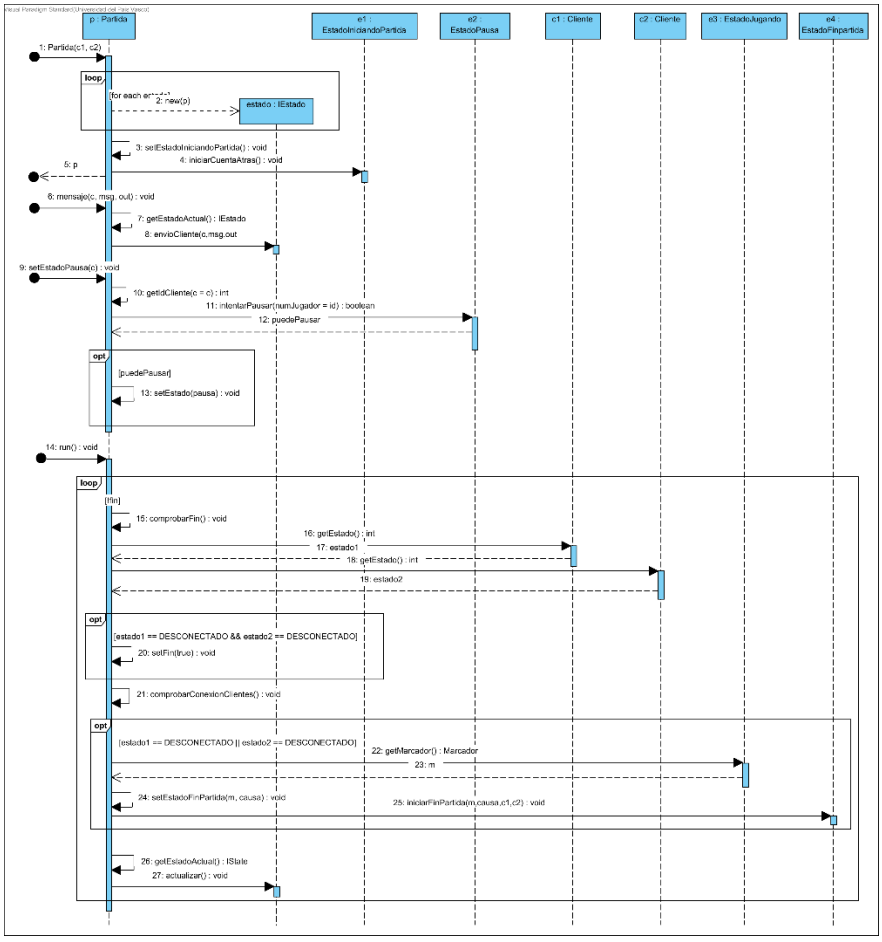



Ilustración 206: SECUENCIA PARTIDA

## EstadoIniciandoPartida

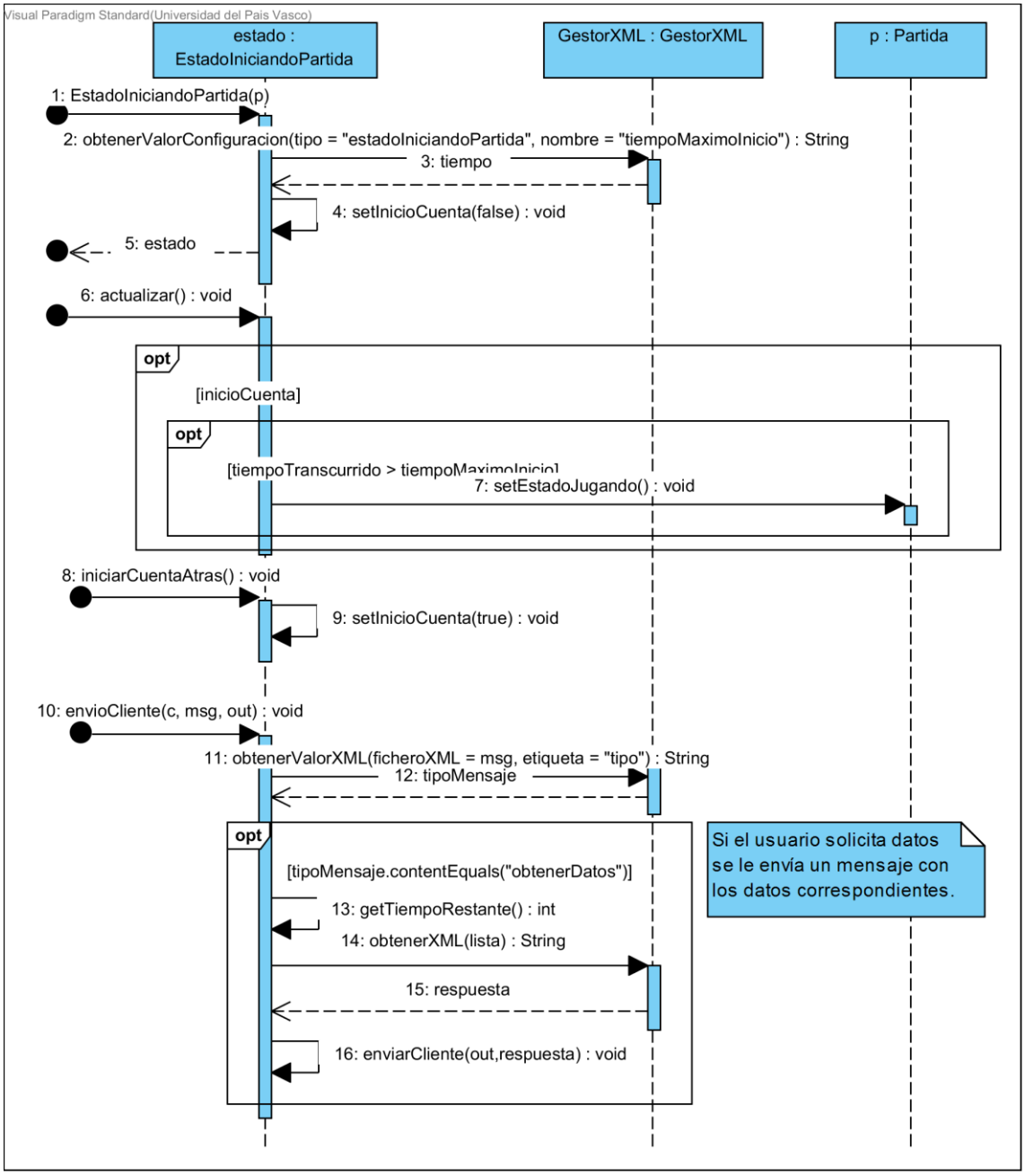



Ilustración 207: SECUENCIA ESTADO INICIANDO PARTIDA

## EstadoJugando

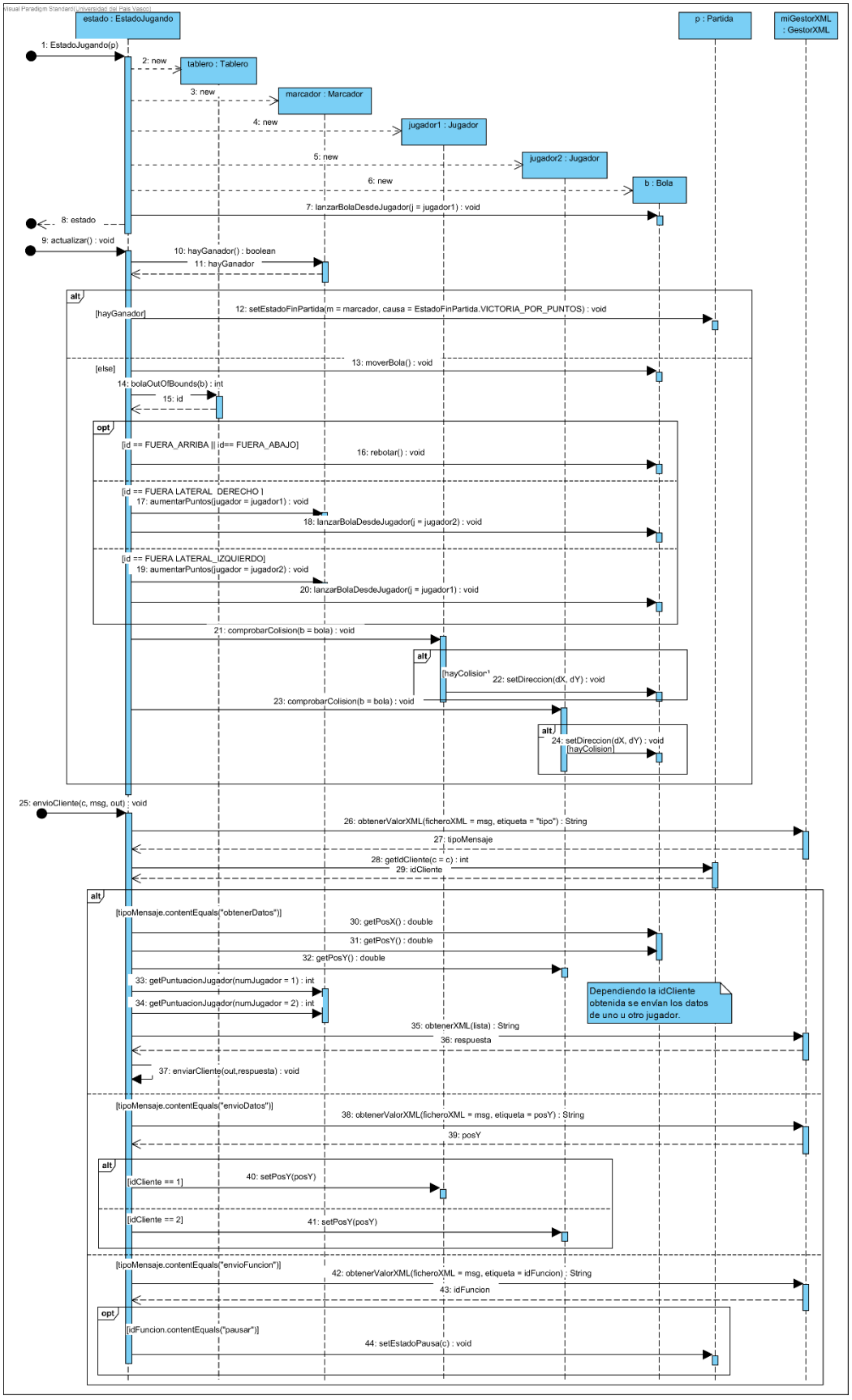



Ilustración 208: SECUENCIA ESTADO JUGANDO

## EstadoPausa

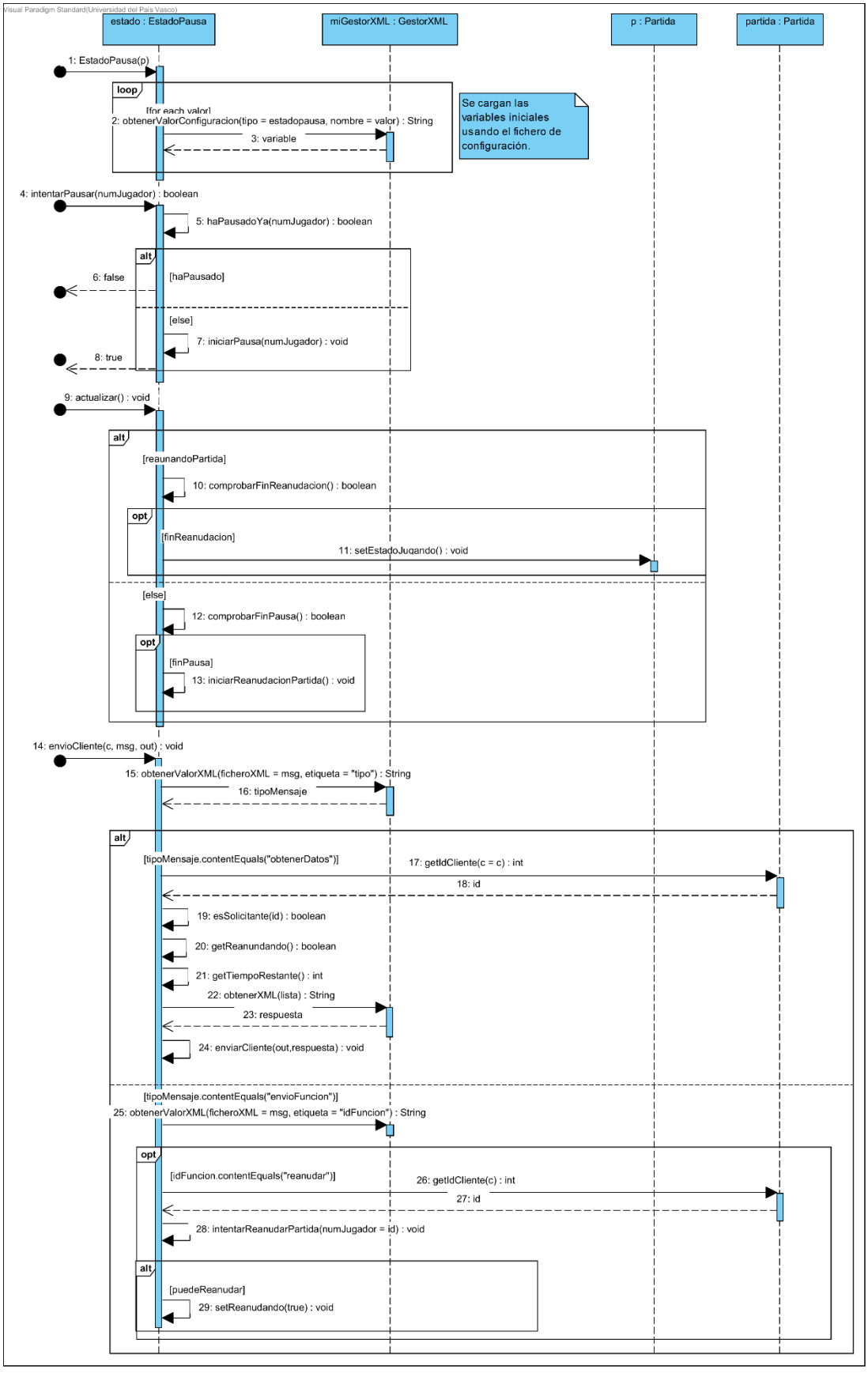



Ilustración 209: SECUENCIA ESTADO PAUSA

## EstadoFinPartida

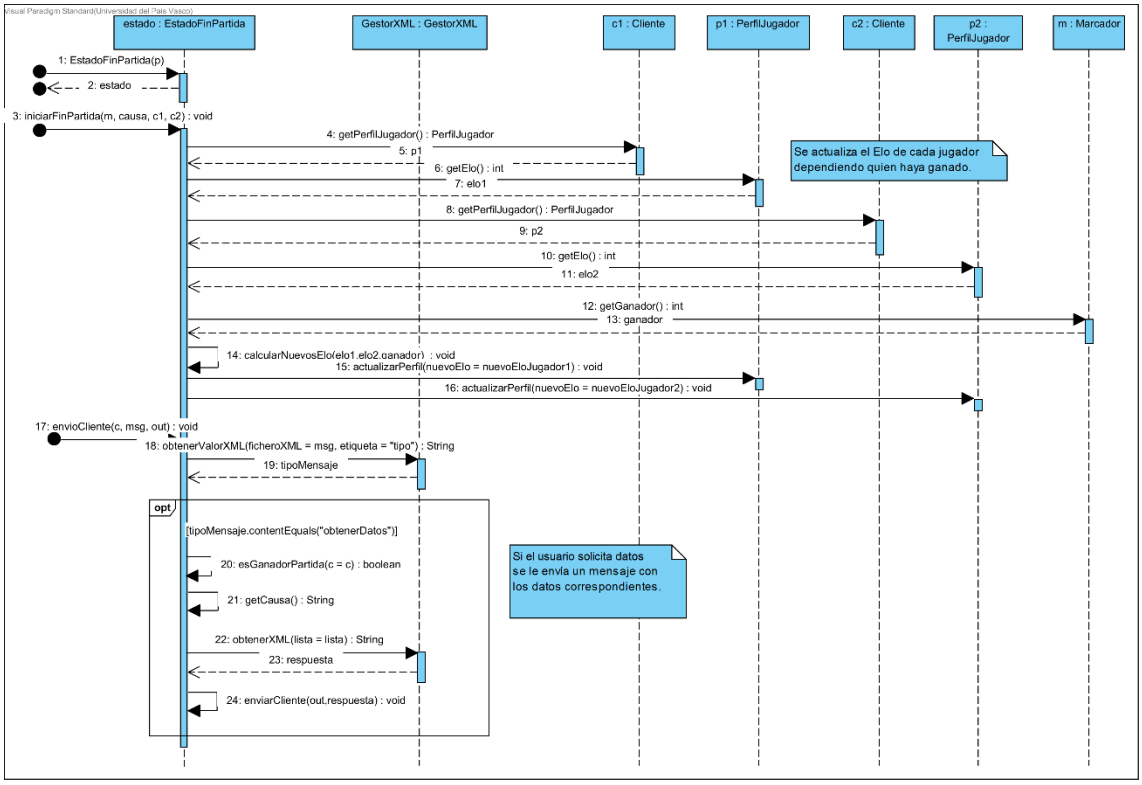



**Ilustración 210: SECUENCIA ESTADO FIN PARTIDA**

## 11.4. Plataforma de juego

En este apartado se mostrarán los diagramas de secuencia correspondientes a cada caso de uso.

### Registrarse

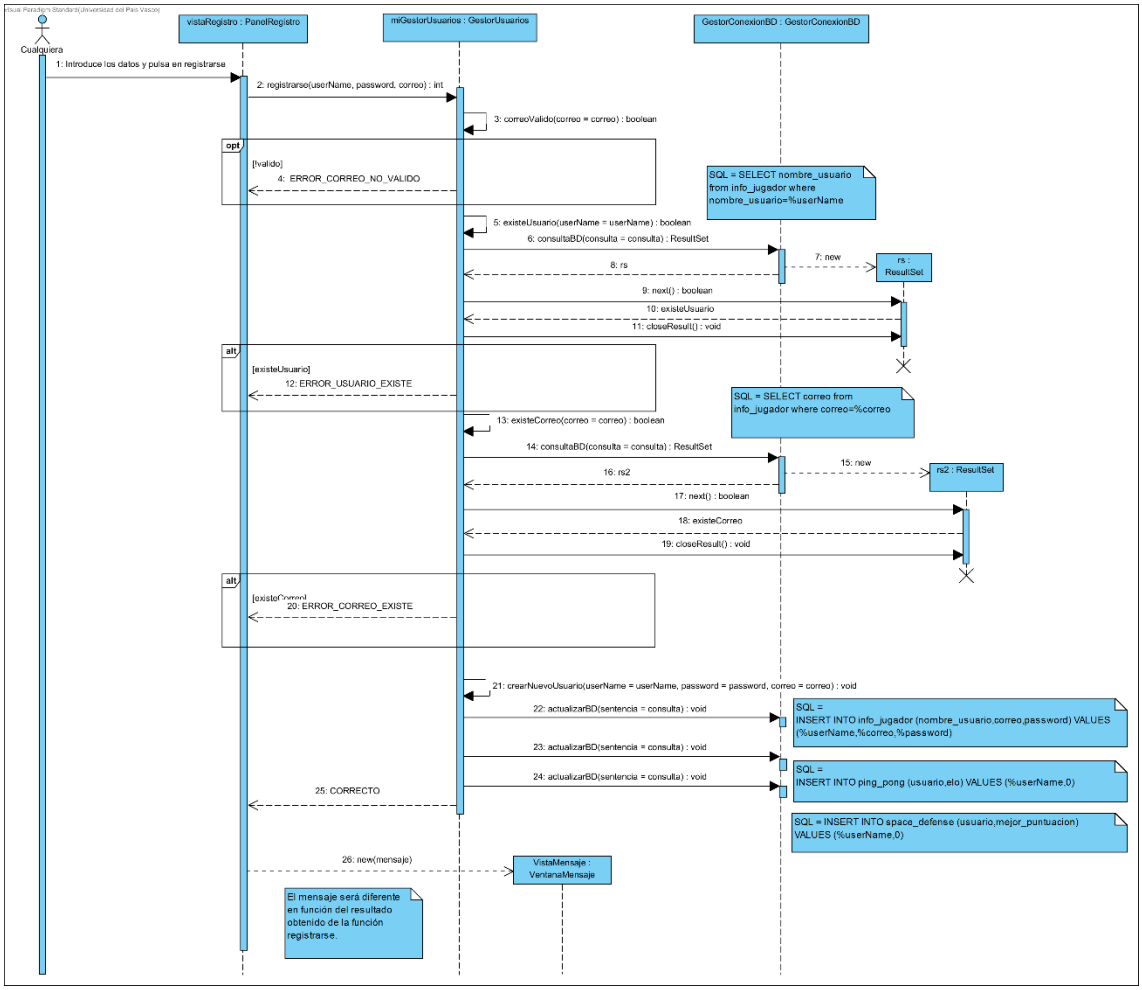



Ilustración 211: SECUENCIA REGISTRARSE

## Recuperar contraseña

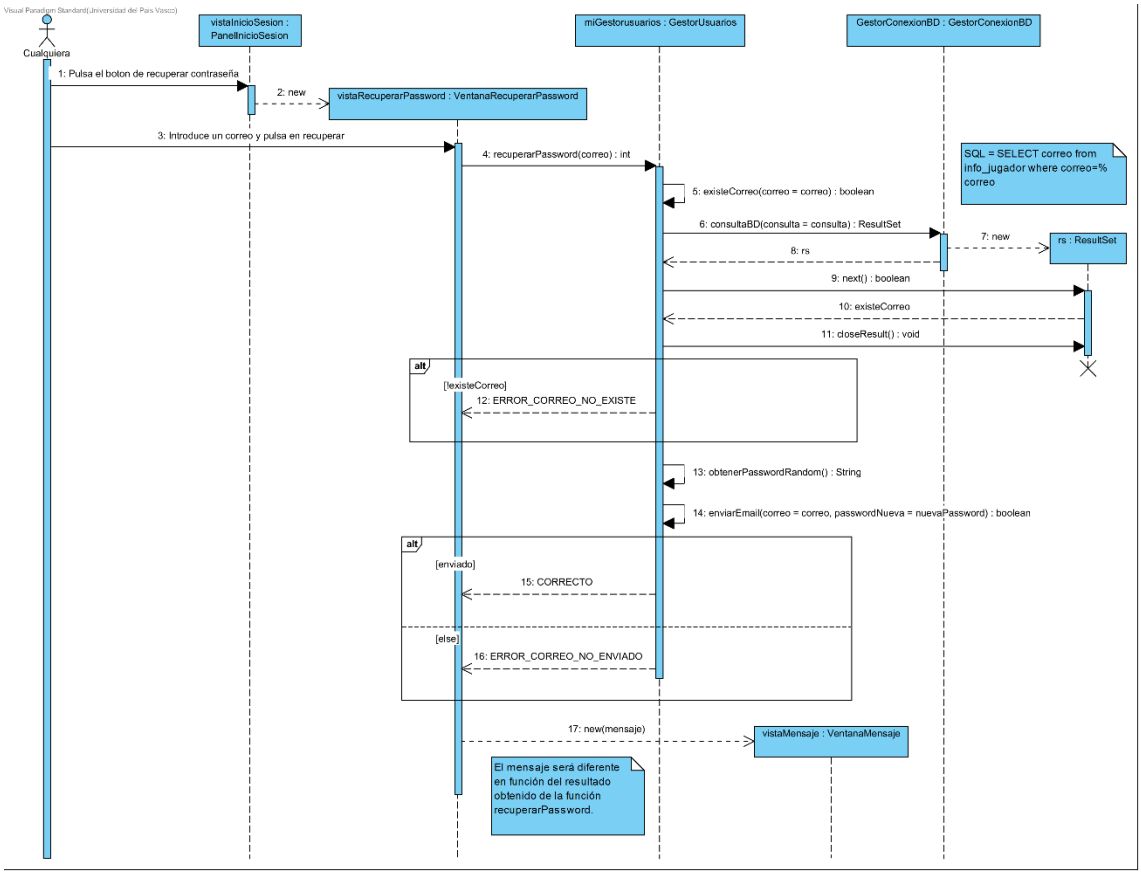



Ilustración 212: SECUENCIA RECUPERAR CONTRASEÑA

## Iniciar sesión

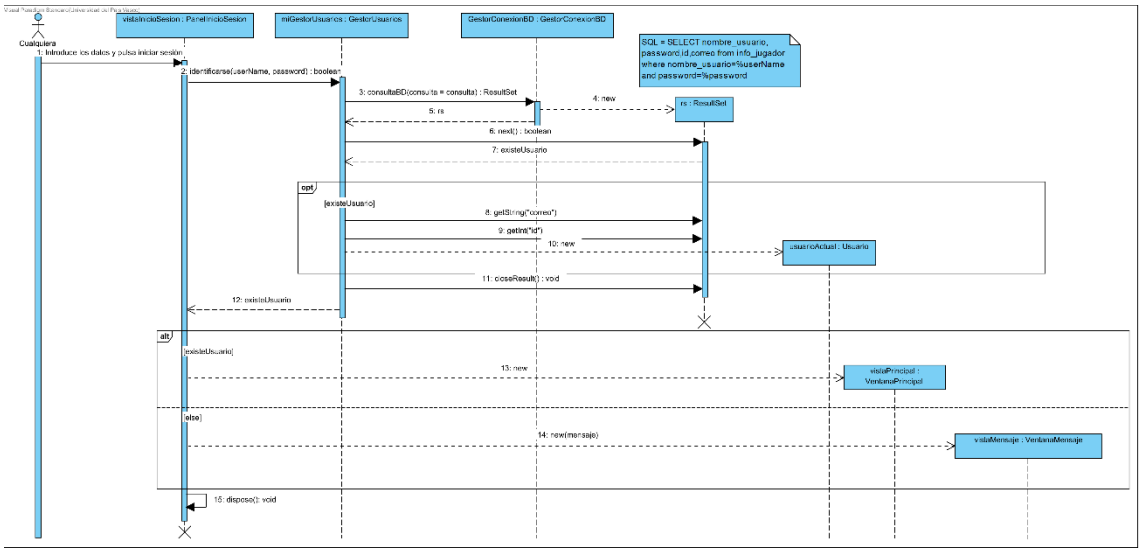



Ilustración 213: SECUENCIA INICIAR SESIÓN

## Constructora VistaPrincipal

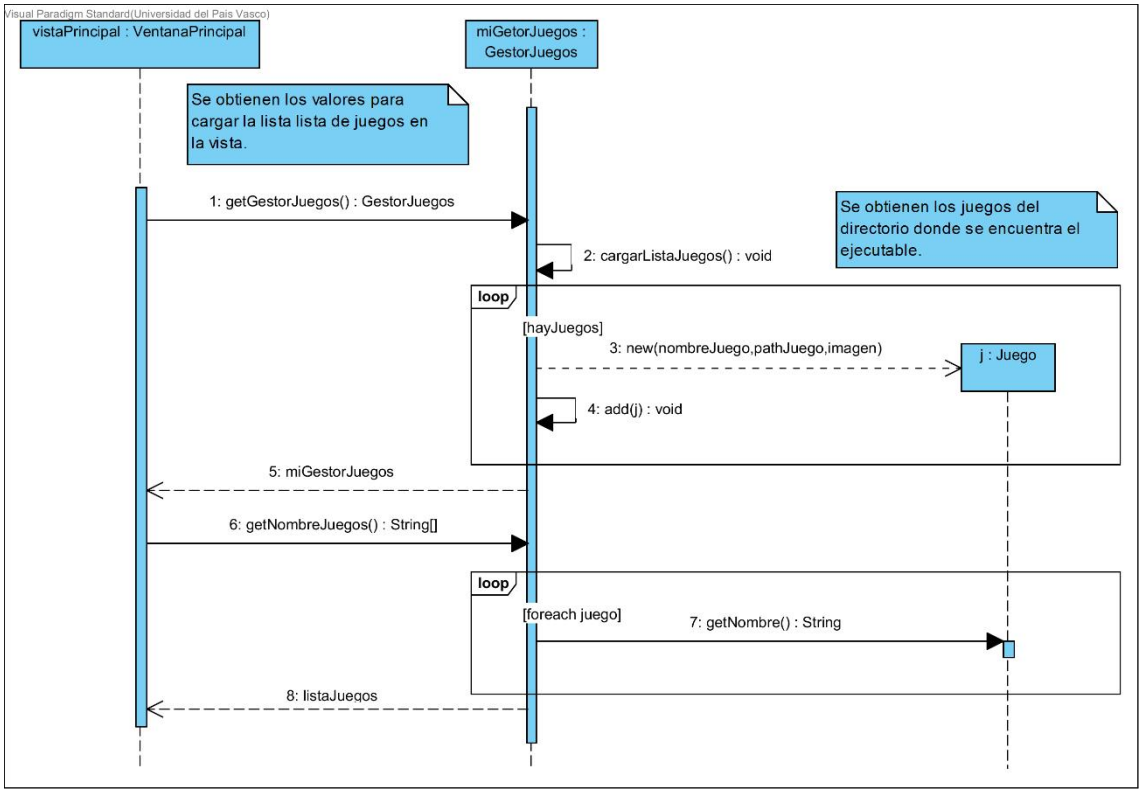



Ilustración 214: SECUENCIA CONSTRUCTORA VISTA PRINCIPAL

## Cambiar correo

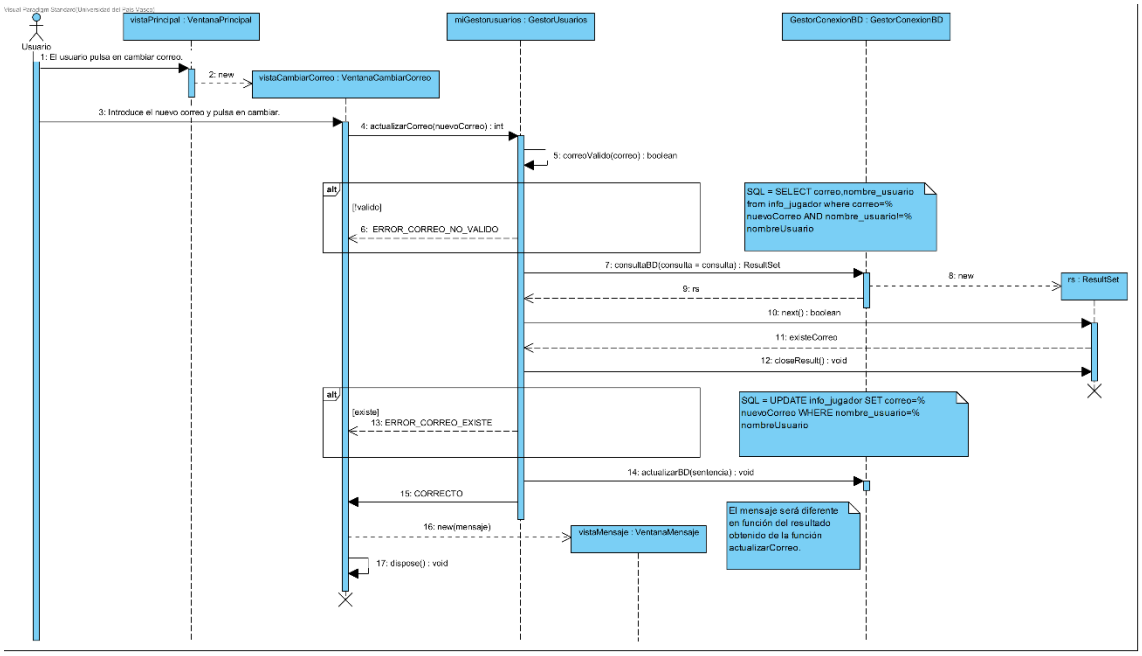



Ilustración 215: SECUENCIA CAMBIAR CORREO

## Cambiar contraseña

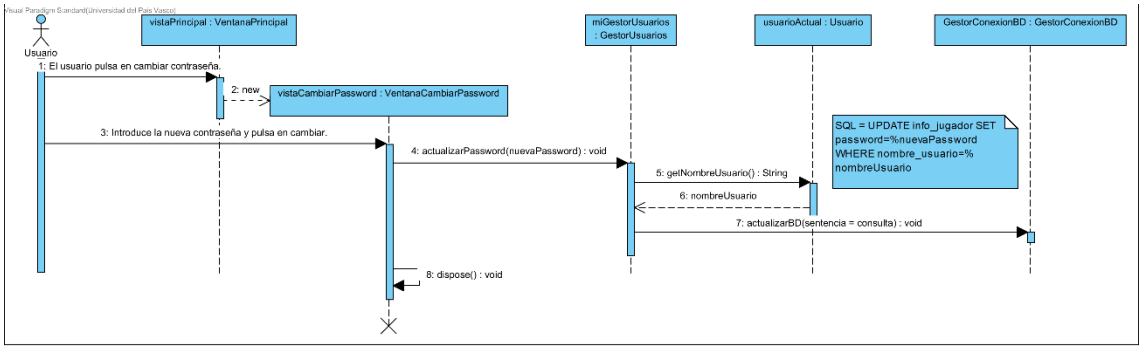



Ilustración 216: SECUENCIA CAMBIAR CONTRASEÑA

## Iniciar juego

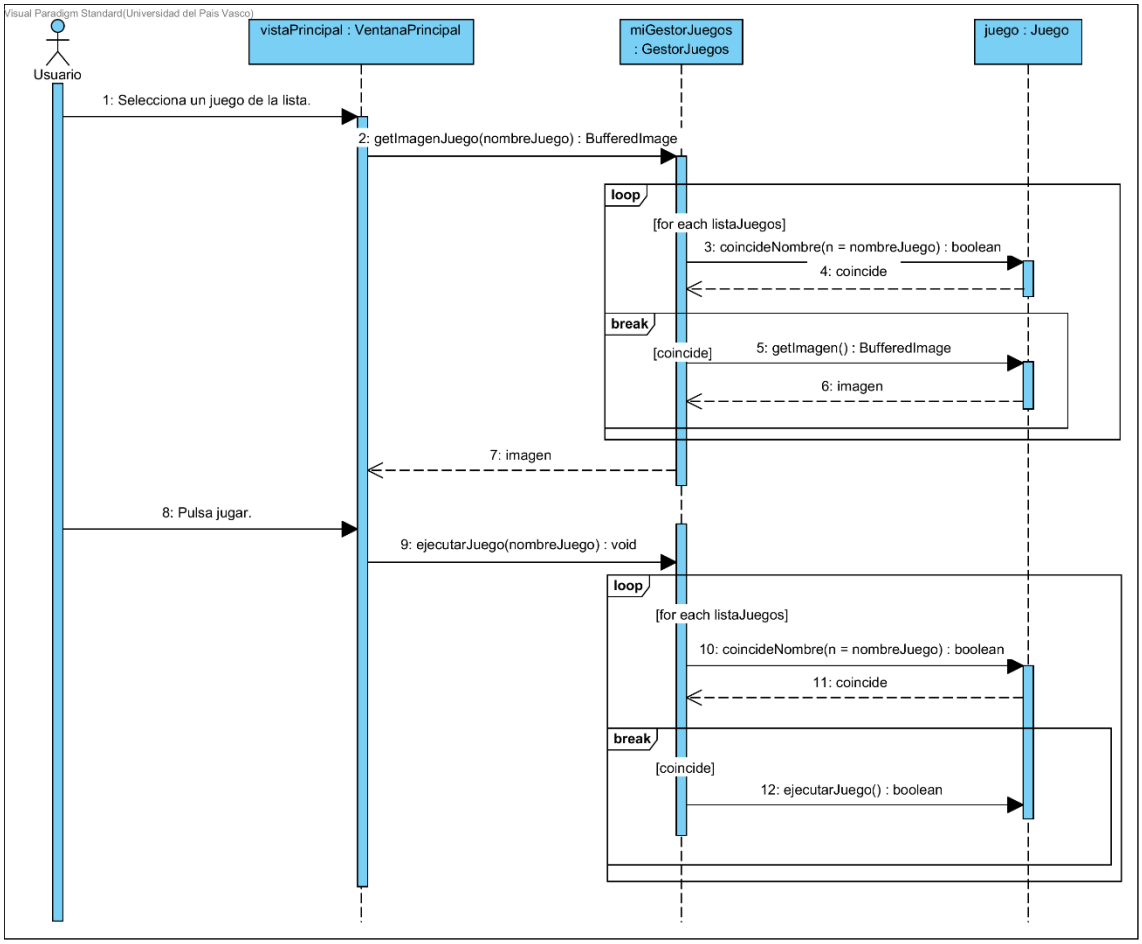



Ilustración 217: SECUENCIA INICIAR JUEGO