*Universidad Autónoma Metropolitana*

*Unidad Azcapotzalco*

División de Ciencias Básicas e Ingeniería

**Reporte Final del Proyecto de Integración**

Licenciatura en Ingeniería en Computación

*Modalidad de Proyecto Tecnológico*

---

# Juego multiusuario para dispositivos móviles con sistema operativo Android

---

Edith Hernández Navarrete

205361429

Asesores:

Dra. María Lizbeth Gallardo López

Dra. Beatriz Adriana González Beltrán

Trimestre Lectivo 2014 Primavera

Agosto de 2014

*Yo, EDITH HERNÁNDEZ NAVARRETE, doy mi autorización* a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

*Yo,* Dra. MARÍA LIZBETH GALLARDO LÓPEZ, *declaro que aprobé el contenido del presente Reporte de Proyecto de Integración* y *doy mi autorización* para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

*Yo,* Dra. BEATRIZ ADRIANA GONZÁLEZ BELTRÁN, *declaro que aprobé el contenido del presente Reporte de Proyecto de Integración* y *doy mi autorización* para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

# Resumen

En este documento se presenta el desarrollo del proyecto que lleva por título “Juego multiusuario para dispositivos móviles con sistema operativo Android”.

Hoy día, la gran mayoría de los niños de tercer grado de primaria en adelante, disponen de dispositivos móviles y es en este periodo escolar que los alumnos aprenden las tablas de multiplicar. Sin embargo, el aprendizaje de las tablas de multiplicar presenta dificultades en un número importante de alumnos. Dados estos factores, se desarrolló una aplicación móvil que sirve como un medio más de apoyo para el aprendizaje de las tablas de multiplicar.

La aplicación está basada en el juego de mesa Dominó, en su versión “Tablas de Multiplicar”. El juego hace uso de la comunicación bluetooth para poder interactuar con otros dispositivos. La tecnología empleada para su desarrollo es Android y Java.

# Tabla de contenido

# Índice de Figuras

# Índice de Tablas

# Índice de diagramas

# 1. Introducción

En los últimos años, el desarrollo de aplicaciones para dispositivos móviles ha experimentado un gran crecimiento en la industria de la telefonía móvil. Uno de los factores de este crecimiento ha sido la aparición de los teléfonos inteligentes (smartphones[1]) con Android[2]. Se considera que el sistema operativo Android está en auge, gracias a que es software libre, lo que facilita el desarrollo de aplicaciones por una gran comunidad de usuarios.

Por otra parte, actualmente la gran mayoría de los niños de tercer año de primaria en adelante disponen de dispositivos móviles y es en este periodo donde se aprenden las tablas de multiplicar. Sin embargo, el aprendizaje de las tablas de multiplicar presenta dificultades en un número importante de alumnos. Dados estos factores, en este proyecto se desarrolló una aplicación móvil que sirve como un medio más de apoyo para el aprendizaje de las tablas de multiplicar.

La aplicación es una adaptación de las tablas de multiplicar al tradicional juego de mesa Dominó. Para jugar es necesario que al menos dos usuarios se unan al juego, a través del establecimiento previo de una comunicación vía bluetooth[3]. La finalidad del proyecto es jugar y aprender al mismo tiempo mediante el uso de dispositivos móviles.

El objetivo de este documento es presentar a grandes rasgos el desarrollo de este proyecto. La sección 1 contiene la introducción de nuestro proyecto, la sección 2 incluye los antecedentes que se investigaron en la propuesta del proyecto, en la sección 3 se da una justificación del proyecto. El objetivo general y los objetivos particulares se describen en la sección 4. La sección 5 muestra el marco teórico del proyecto. En la sección 6 se encuentra del desarrollo del proyecto que incluye la metodología empleada y el diseño del sistema con sus respectivos diagramas. La sección 7 muestra los resultados obtenidos en base a los objetivos particulares del proyecto. La sección 8 muestra un análisis y discusión de resultados. Las conclusiones del proyecto se muestran en la sección 9. Las perspectivas del proyecto se muestran en la sección 10. Por último, se muestran las referencias bibliográficas y, como apéndice, el API del código fuente de la aplicación.

[1] Teléfonos Inteligentes.

[2] *Android* es una plataforma completa de código abierto diseñado para dispositivos móviles. Es promovido por Google y es propiedad de la Open Handset Alliance. [1]

[3] *Bluetooth* es la norma que define un estándar global de comunicación inalámbrica que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace de radiofrecuencias.

# 2. Antecedentes

En esta sección se hace una breve descripción de aquellos proyectos que se investigaron y que están relacionados con este proyecto.

## 2.1 Referencias externas

A continuación se mencionan los proyectos propietarios o de libre distribución, desarrollados fuera de la Universidad Autónoma Metropolitana, disponibles al público en general y que tienen alguna relación con el proyecto.

- ***ChessAsin, servidor de ajedrez por correspondencia.*** Este proyecto de tesis fue propuesto en diciembre de 2011 en la Universidad Carlos III de Madrid [6] y se centra en el desarrollo del juego ajedrez para teléfonos inteligentes. Este proyecto fue desarrollado para la plataforma móvil Windows Phone 7 con el lenguaje de programación C#. La aplicación del proyecto que se presenta en este documento está realizada para la plataforma Android y el lenguaje de programación Java.
- ***POKERAN: Juego de poker***. Propuesto en junio de 2012, en la Universidad Autonóma de Barcelona [7], este proyecto desarrolla un juego de poker para un máximo de cuatro personas, pero utilizando un solo dispositivo. Está desarrollado en lenguaje Java y utiliza el sistema operativo Android. En nuestro proyecto la aplicación utiliza un mínimo de dos dispositivos, los cuales se conectan vía bluetooth, se utiliza el lenguaje de programación Java y está realizado para la plataforma Android.

- ***Desarrollo de un Sistema de Juego ubicuo bajo Plataforma Android.*** Este proyecto desarrolla un juego ubicuo de realidad aumentada para entornos de exterior bajo la plataforma Android [8]. Esta aplicación hace uso de la realidad aumentada[4] y emplea un solo dispositivo móvil, mientras que nuestro proyecto emplea la comunicación bluetooth, requiere de la interacción de dos dispositivos móviles y hace uso del sistema operativo Android.

## 2.2 Referencias internas

En esta sección se mencionan los proyectos realizados en la Universidad Autónoma Metropolitana (UAM), Unidad Azcapotzalco y que tienen alguna similitud con nuestra aplicación.

- ***Implementación de un sistema de noticias con dualidad en la obtención de la información: vía aérea y vía bluetooth***. Propuesto en el trimestre 2007 primavera [3], este proyecto se enfoca en el envío de información de noticias desde un servidor ubicado en una computadora de escritorio hacia dispositivos móviles que estén a su alcance. Este proyecto difiere con el

[4] La realidad aumentada (RA) es el término que se usa para definir una visión a través de un dispositivo tecnológico, directa o indirecta, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real.

nuestro porque no existe interacción directa entre los dispositivos móviles, además, los móviles utilizados no son inteligentes..

- ***Sistema de transmisión y recepción vía bluetooth para votación.*** Propuesto en el trimestre 2008 otoño [4]. Este proyecto está enfocado en el envío y recepción de información a través de una conexión bluetooth, y toma como caso de estudio la votación en juntas ejecutivas. En nuestro proyecto se diseñó un juego educativo.

- ***Aplicación Android para la sincronización de las tareas y el material didáctico de sistemas Moodle.*** Propuesto en el trimestre 2011 primavera [5], aquí se propone una aplicación Android para tener acceso a los contenidos (cursos, tareas y material didáctico) de un sistema de gestión de cursos Moodle. En este proyecto, los dispositivos móviles solo se comunican con el servidor y no existe interacción entre los dispositivos móviles ni se requiere de la comunicación bluetooth.

# 3. Justificación

En la actualidad, los dispositivos móviles se han convertido en una herramienta útil con la que el usuario disfruta de una amplia variedad de funcionalidades. Más allá de las aplicaciones básicas y de las llamadas telefónicas, estos dispositivos se han convertido en un elemento de apoyo en las actividades de las personas, ya sea para trabajar, para estudiar o para divertirse. El uso extendido de estos aparatos ha provocado no sólo la necesidad de crear nuevas aplicaciones para los usuarios, sino la necesidad también de migrar y adaptar a dichos dispositivos móviles, programas ya existentes para una computadora de escritorio.

Para muchos alumnos de primaria, aprender las tablas de multiplicar es todo un reto, se emplean múltiples medios audio-visuales para que los niños las practiquen. Aprovechando que hoy en día los niños entre 8 y 11 años disponen de un teléfono celular, el propósito de este proyecto es desarrollar una aplicación multiusuario que va a ser un medio adicional para que los niños de tercero a quinto grado de primaria practiquen las tablas de multiplicar. De manera concreta, se desarrolló el juego de mesa Dominó – Tablas de multiplicar  para dispositivos móviles con sistema operativo Android.

# 4. Objetivos

Los objetivos planteados para la realización de este proyecto terminal son los siguientes:

## 4.1 Objetivo General:

Diseñar e implementar un juego multiusuario para dispositivos móviles con sistema operativo Android, tomando como caso de estudio una versión del juego de mesa Dominó – tablas de multiplicar.

## 4.2 Objetivos Particulares:

- Diseñar e implementar un módulo gestión del juego.
- Diseñar e implementar un módulo gestión de mensajes.
- Diseñar e implementar un módulo gestión de comunicación entre dispositivos.
- Diseñar e implementar un módulo de interfaz gráfica del juego.

# 5. Marco Teórico

En esta sección se presentan algunos conceptos que nos sirvieron como base para el desarrollo de nuestra aplicación.

## 5.1 *Aplicación multiusuario*

Una aplicación multiusuario es un programa que permite proveer servicio y procesamiento a múltiples usuarios simultáneamente. Este tipo de aplicaciones difiere de las aplicaciones monousuario que proveen servicio y procesamiento a un solo usuario.

Las características de las aplicaciones multiusuarios se pueden resumir como sigue:

- ✓ Compartir recursos, lo cual no se puede efectuar con una aplicación monousuario.
- ✓ Envío y recepción de mensajes o datos entre distintos dispositivos.
- ✓ Soportan conexión con más de un dispositivo, lo cual difiere con las aplicaciones monousuarios.
- ✓ Manejo de eventos entre dispositivos.

## 5.2 *Aplicaciones Android*

A continuación se describen algunos de los conceptos mínimos que el usuario debe conocer para poder desarrollar una aplicación Android.

***Activity.*** Una clase *Activity* crea una interfaz de usuario interactiva para una aplicación que va a funcionar en *Android*. Está conformada por dos partes: la parte lógica y la parte gráfica.

La parte lógica es un archivo Java que especifica la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad. La parte gráfica es un archivo XML que tiene todos los elementos de una pantalla declarados con etiquetas parecidas a las de HTML.

Una actividad puede encontrarse en tres estados:

- ***Running*** (ejecutándose). Permite la interacción con el usuario.
- ***Paused*** (pausado). Cuando la *Activity* está todavía en la pantalla, pero en estado latente y pausada.
- ***Stopped*** (detenido). Cuando la *Activity* está completamente oscurecida por otra *Activity*.

***View.*** Esta clase representa el bloque básico de construcción para los componentes de una interfaz de usuario.

***Layout.*** Es un contenedor de una o más vistas y controla su comportamiento y posición. Hay que destacar que un Layout puede contener a otro Layout y que es un descendiente de la clase View. A continuación se muestran los tipos de Layout más utilizados en Android.

- ***LinearLayout***. Coloca los elementos en una fila o en una columna.
- ***TableLayout***. Posiciona los elementos de forma tabular.
- ***RelativeLayout***. Distribuye los elementos en relación a otro Layout.
- ***AbsoluteLayout***. Posiciona los elementos en forma absoluta.

- ***FrameLayout***. Permite el cambio dinámico de los elementos que contiene.

***ImageView.*** La clase *ImageView* hereda los métodos y atributos de la clase *View. ImageView* permite desplegar una imagen a partir de distintos recursos, a saber: imágenes, *bitmaps, drawables*[5].

***Button.*** Esta clase representa a un botón, sobre el cual se programa una acción específica, a petición del usuario.

***Context.*** *Context* es una clase abstracta del sistema *Android* que permite el acceso a recursos específicos de la aplicación; permite además que la aplicación realice solicitudes al sistema Android tales como: lanzadores de actividades, etc.

***HorizontalScrollView.*** Layouts de un contenedor para una jerarquía de vistas (views) que se pueden desplazar por el usuario, permitiendo que sea más grande que la pantalla física.

**AndroidManifest.xml.** Este es un archivo XML requerido para cualquier aplicación de Android, está localizado en la raíz del directorio del proyecto y su función es describir los valores globales del proyecto, incluyendo los componentes de la aplicación (actividades, servicios, etc.). Es importante mencionar que al declarar un componente, se declaran al mismo tiempo algunos de sus atributos como son: el momento de iniciar el componente, la posición o el tema que adoptará una vez iniciado el componente.

## 5.3 *Juego de mesa dominó*

A continuación se describe el juego de mesa dominó en su versión original.

El dominó es un juego de mesa en el que se emplean unas fichas rectangulares, generalmente blancas por la cara y negras por el revés, divididas en dos cuadrados, cada uno de los cuales lleva marcado de cero a seis puntos. El juego completo de fichas de dominó consta normalmente de 28 piezas siendo la ficha más grande la de doble seis.

***Jugadores****:* El juego generalmente se juega con cuatro jugadores en solitario. El juego de dominó se puede jugar con 2, 4 o 6 personas y se tienen que dividir las fichas según la cantidad de jugadores.

***Objetivo****:* El objetivo del juego es alcanzar una determinada puntuación previamente fijada, jugando para ello las manos o rondas que sean precisas.

El jugador que gana una ronda, suma los puntos de las fichas de sus adversarios. El primer jugador que alcanza la puntuación fijada al principio de la partida, gana.

***Inicio del juego****:* El primer turno es para el jugador que tenga la ficha con mayor puntuación, y la ronda, empezará el jugador a la derecha del que empezó la ronda anterior.

***Desarrollo del juego****:* En su turno, cada jugador colocará una de sus piezas con la restricción de que dos piezas sólo puedan colocarse juntas cuando los cuadrados adyacentes sean del mismo valor (ej, el 1 con el 1, el 2 con el 2, etc.).

[5] *Drawable* es una abstracción de Android para cualquier cosa que pueda ser dibujado. En este caso se refiere a recursos de la aplicación como son las imágenes de cada uno de las fichas.

***Fin del juego**:* La mano continúa hasta que se da alguna de las dos situaciones.

- Alguno de los jugadores se queda sin fichas válidas para colocar en la mesa de juego.
- En caso de cierre, cuando a pesar de que hay fichas disponibles en el juego ninguna puede colocarse, se atribuye el cierre. El jugador que tenga la ficha de menor valor gana la partida.

## 5.4 *Juego de mesa Dominó – Tablas de multiplicar*

El juego de mesa Dominó – Tablas de Multiplicar, emplea 43 fichas rectangulares, divididas en dos cuadrados, en el primer cuadro lleva marcado la operación de una multiplicación y en el otro cuadro se muestra un número, que representa el resultado de una operación de multiplicación.

***Jugadores***: El juego emplea al menos a dos jugadores y al iniciar el juego se reparte a cada jugador 7 fichas.

***Objetivo***: El objetivo del juego es colocar la mayor cantidad de fichas posibles en el tablero, durante la ronda de juego.

El jugador que gana una ronda, es aquel que no tiene ninguna ficha, o si se fijo un tiempo, el que tiene el menor número de fichas en comparación con su adversario.

***Inicio del juego***: El jugador que coloque primero una ficha es el que inicia el juego.

***Desarrollo del juego***: Cada vez que a un jugador le toque su turno, éste colocará una de sus fichas en el tablero de juego, con la restricción de que dos fichas sólo pueden colocarse juntas cuando los cuadrados adyacentes sean del mismo valor. Por ejemplo, el 3x4 con el 12 y el 6x3 con el 18 (ver Figura 1).


Figura 1. Movimiento válido, durante el desarrollo del juego.

***Fin del juego***: La ronda continúa hasta que se da alguna de las dos situaciones:

- Alguno de los jugadores se queda sin fichas válidas para colocar en el tablero de juego.
- Si se acabó el tiempo, entonces el jugador que tenga el menor número de fichas gana la partida.

# 6. Descripción Técnica

En este apartado se describen las etapas generales para el análisis, diseño e implementación de nuestra aplicación.

## 6.1 Metodología empleada en el desarrollo del proyecto

Para el desarrollo e implementación del sistema se utilizó la metodología del Proceso Unificado, que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental.

**Dirigido por los casos de uso**: Los casos de uso se emplean para capturar los requisitos funcionales del sistema, las necesidades del usuario y para definir los contenidos de las iteraciones del sistema.

**Centrado en la arquitectura**: Dado que no existe un modelo único que cubra todos los aspectos del sistema, existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

**Iterativo e incremental**: El proceso unificado está compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez divida en una serie de iteraciones. Dichas iteraciones ofrecen como resultado un incremento del sistema que se añade o mejora las funcionalidades del sistema en desarrollo. Cada una de las iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: análisis de requisitos, diseño, implementación y pruebas.

## 6.2 Diseño del sistema

En este apartado se proporcionarán los diagramas *UML*[6] empleados para el diseño de los diagramas de casos de uso de la aplicación, así como su descripción.

### 6.2.1 Diagrama de estados

A continuación se presenta el diagrama de transición de estados (DTS) que describe al juego dominó las Tablas de Multiplicar (ver Diagrama 1).

[6] UML son las siglas de *Unified Modeling Language* (Lenguaje Unificado de Construcción de Modelos), notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos [10].

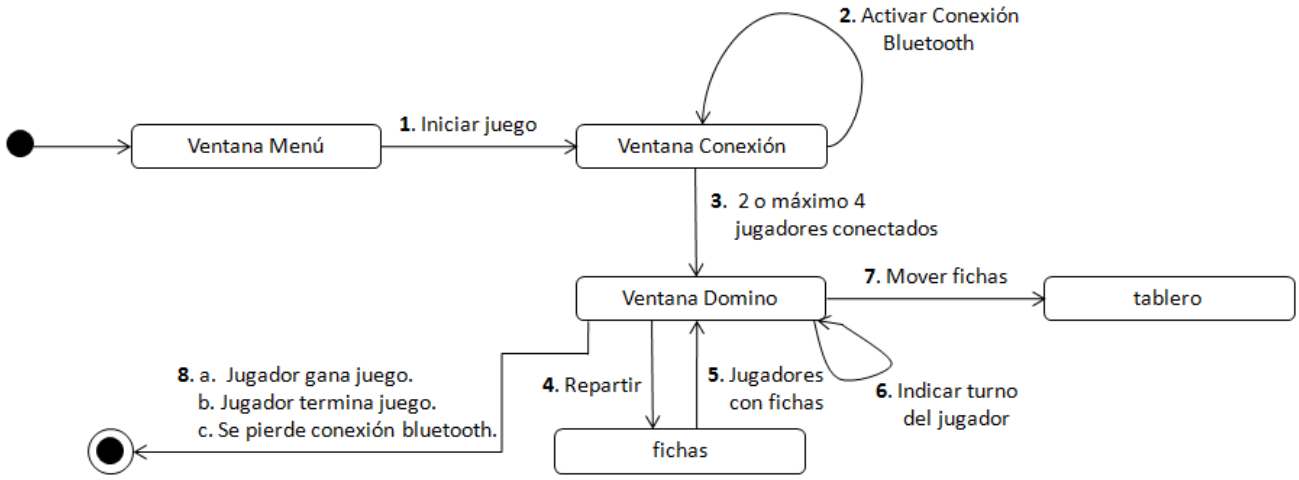



Diagrama 1. Diagrama de Estados de la aplicación.

### 6.2.2 Diagrama de casos de uso

El diagrama general de los casos de uso para nuestra aplicación Juego Multiusuario se muestra en el Diagrama 2.

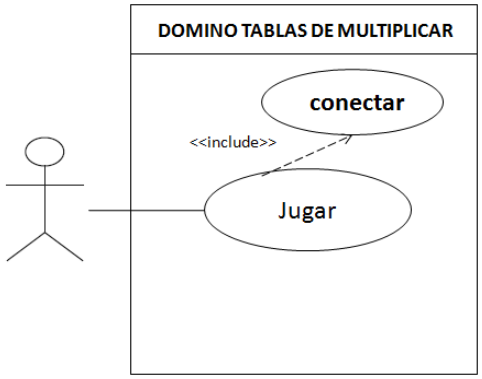



Diagrama 2. Diagrama de caso de uso general.

### 6.2.3 Diagrama de secuencias del sistema

El diagrama de secuencia muestra la iteración del usuario con la aplicación Dominó –Tablas de multiplicar. En el Diagrama 3 se pueden apreciar todas las actividades o acciones que pueden ocurrir al jugar.

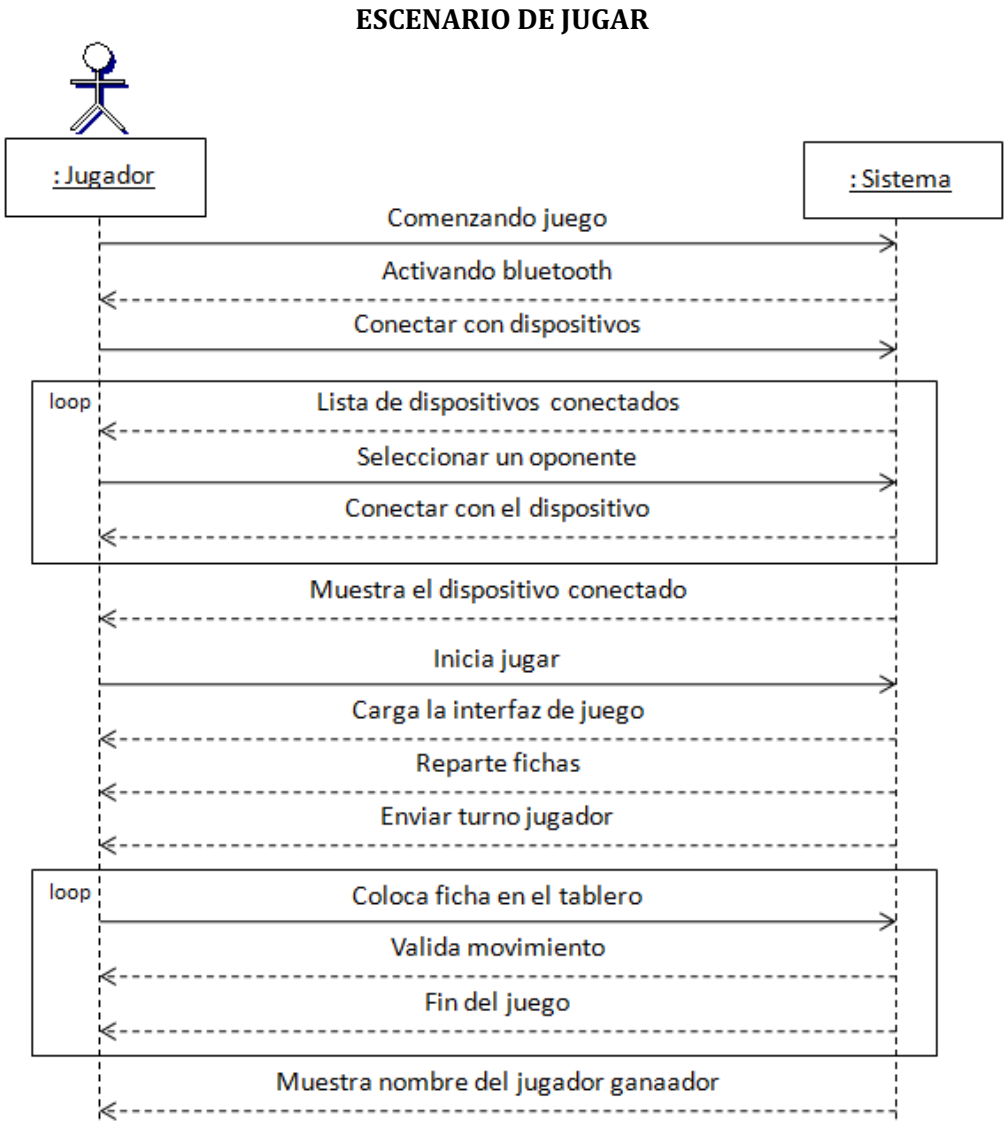



Diagrama 3. Diagrama de Secuencia de la aplicación.

### 6.2.4 Diagrama de Robustez

En el Diagrama 4 se puede apreciar el diagrama de robustez, el cual muestra iteración entre las actividades de la aplicación.

**DIAGRAMA DE ROBUSTEZ**

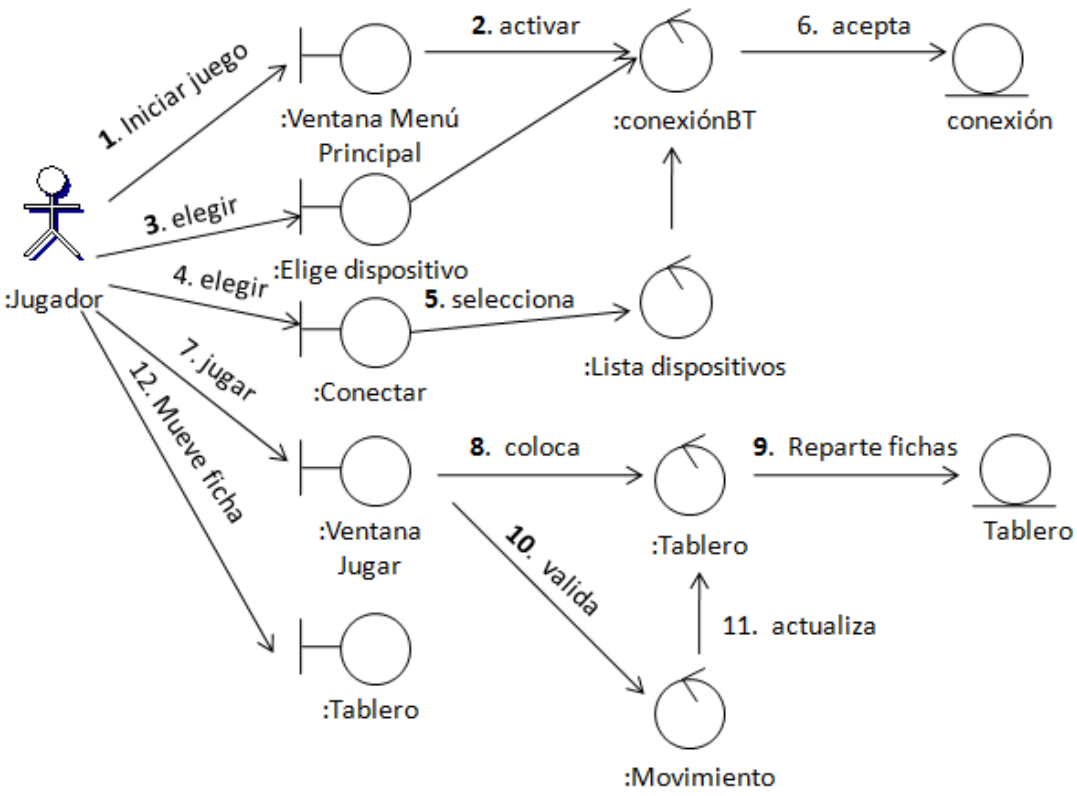



Diagrama 4. Diagrama de Robustez.

### 6.2.5 Diagrama de secuencias

El diagrama de secuencia (ver Diagrama 5) presenta las actividades de la aplicación que pueden ocurrir durante el transcurso de nuestra aplicación. Como podemos observar, se puede entender gráficamente la funcionalidad de nuestra aplicación.

**DIAGRAMA DE SECUENCIA**

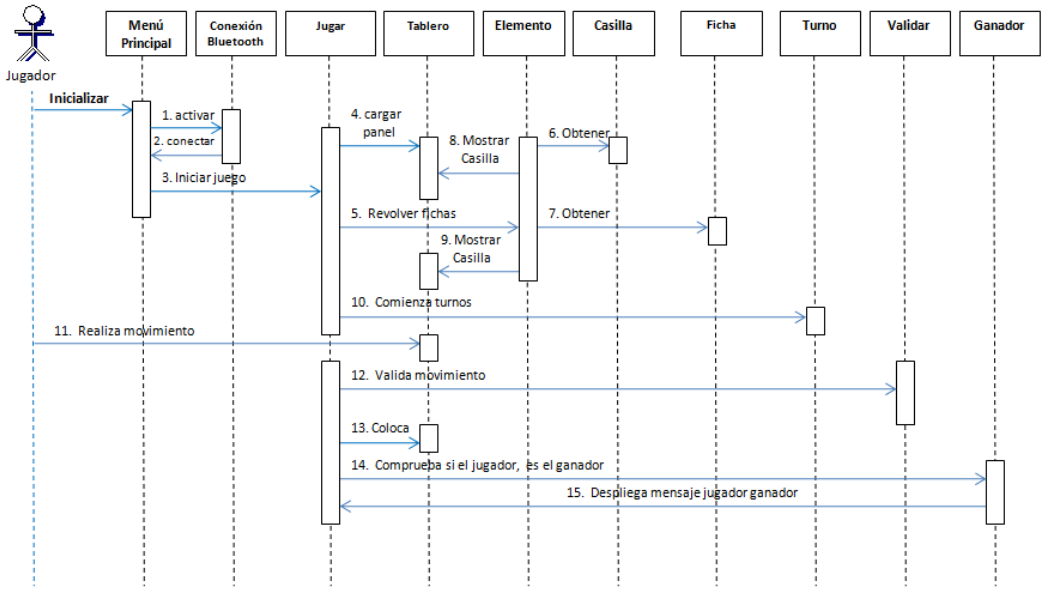



Diagrama 5. Diagrama de secuencia.

### 6.2.6 Casos de uso de texto

A continuación se presenta el caso de uso principal de nuestra aplicación.

**Caso de uso UC1: Jugar**

**Actor principal**: Jugador

**Personal involucrado e intereses:**

- Jugador: jugar con la aplicación Dominó - Tablas de Multiplicar.

**Precondiciones**: La comunicación vía bluetooth se encuentra habilitada y dos usuarios están conectados.

**Garantías de éxito (Postcondiciones)**:

El jugador se conecta a la aplicación. El jugador comienza el juego. El sistema va colocando las fichas de dominó en el tablero. El juego termina cuando un jugador no tenga ninguna ficha de domino, o cuando al decidir terminar el juego se tenga el menor número de fichas en comparación con su adversario.

**Escenario principal de éxito (o Flujo Básico):**

1. El jugador se une al juego (inicia la aplicación).
2. El sistema muestra la pantalla del menú principal, contiene las opciones conectar, jugar y salir.
3. El usuario oprime el botón conectar.
4. El sistema muestra la lista de dispositivos vinculados.
5. El jugador selecciona un dispositivo para conectarse.
6. El sistema se conecta con el dispositivo seleccionado y muestra el mensaje "conectando con dispositivo".
7. El sistema indica a los jugadores que están conectados y envía mensaje para que algún jugador oprima el botón Jugar.
8. El sistema espera que un jugador oprima botón Jugar para comenzar.
9. El jugador oprime el botón Jugar del Menú principal.
10. El sistema carga el escenario (la interfaz) del juego.
11. El sistema revuelve y reparte las fichas de dominó a los jugadores.
12. El sistema muestra nombre del jugador que comienza el juego e indica el turno del jugador.
13. En el turno de cada jugador, este elige una ficha de dominó, y la mueve hasta el tablero para colocarla.
14. El sistema comprueba que la ficha seleccionada por el jugador sea válida.
15. El sistema coloca la ficha en el tablero y el sistema refleja el movimiento para ambos jugadores.
16. El sistema repite los pasos 12-16, hasta que comprueba si algún jugador ya no tiene fichas, o que el tablero de juego no tenga casillas vacías.
17. El sistema muestra mensaje del jugador que ha ganado el juego.
18. El sistema muestra mensaje fin del juego.

**Extensiones (o Flujos Alternos):**

1. Al iniciar la aplicación, no está habilitada la conexión bluetooth.
    1a. El sistema solicita al usuario que habilite la conexión bluetooth.

*a. En cualquier momento:
1. El sistema no puede conectarse vía bluetooth.
    1a. El sistema informa del error al usuario y regresa al menú principal de la aplicación.
2. El jugador no tiene ficha válida para colocar en el tablero.
    2a. Seleccionar tomar ficha del montón.
        1. En caso de haber fichas disponibles tomará hasta que encuentre alguna ficha válida para seguir con el juego.
        2. Si no hay fichas disponibles, pasará su turno.
3. El jugador mueve ficha incorrecta.
    3a. El sistema informa al jugador que la ficha que movió al tablero no es correcta.
4. Se perdió la conexión vía bluetooth.
    4a. El sistema informa a los jugadores que no hay conexión vía bluetooth, y da por terminada la aplicación.
5. El jugador abandona el juego.
    5a. El sistema informa a los participantes que un jugador abandonó el juego, y da por terminada la aplicación.

**Requisitos especiales:**

- Interfaz de usuario con texto visible y legible.
- Tiempo de respuesta rápido para los movimientos de los jugadores.

### 6.2.7 Diagramas de clases

El diagrama de clases (ver Diagrama 6) presenta las clases del modelo lógico de la aplicación, así como las clases de Android que heredan o implementan para el funcionamiento de la aplicación.

**DIAGRAMA DE CLASES**

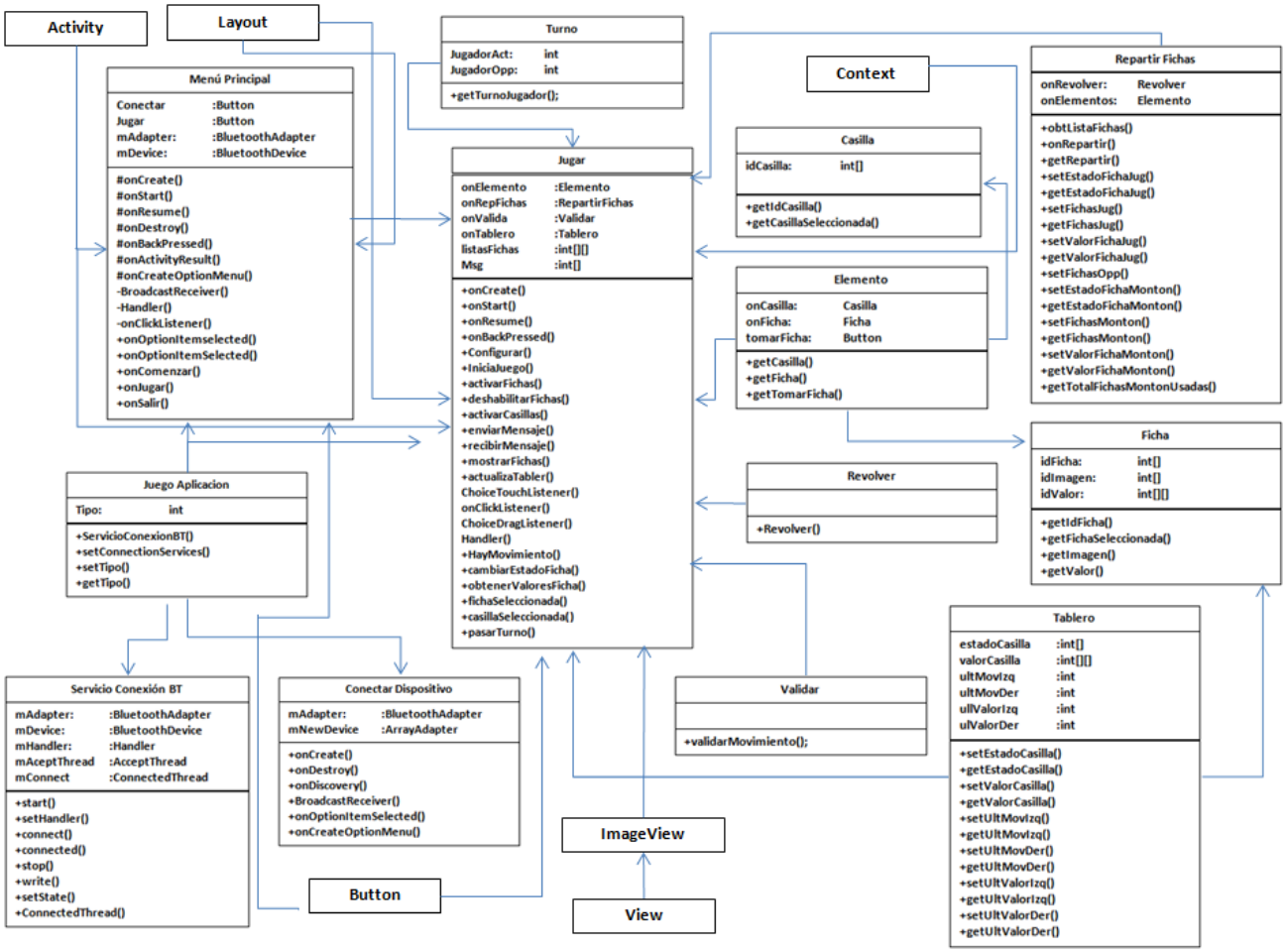



Diagrama 6. Diagrama de Clases General.

En las subsecciones siguientes se describe el diagrama de clases.

#### 6.2.7.1 Clases Android

- ✓ **Activity.** Una clase *Activity* crea una interfaz de usuario interactiva para una aplicación que va a funcionar en Android.

- ✓ **View.** Esta clase representa el bloque básico de construcción para los componentes de una interfaz de usuario.
- ✓ **Layout.** Un layout es una clase que actúa como un marco en la pantalla, el cual puede contener una o más vistas; sin embargo, sólo la vista superior es visible completamente, el resto de las vistas se encuentran apiladas por debajo de las demás capas.
- ✓ **ImageView.** La clase ImageView hereda los métodos y atributos de la clase View. ImageView permite desplegar una imagen a partir de distintos recursos, a saber: imágenes, bitmaps, drawables.
- ✓ **Button.** Esta clase representa a un botón, sobre el cual se programa una acción específica, a petición del usuario.
- ✓ **Context.** Context es una clase abstracta del sistema Android que permite el acceso a recursos específicos de la aplicación; permite además que la aplicación realice solicitudes al sistema Android tales como: lanzadores de actividades, etc.

#### 6.2.7.2 Clases de la aplicación Dominó – Tablas de Multiplicar

- ✓ ***Menú Principal***. Esta clase hereda los atributos y métodos de la clase *Activity*. Esta clase tiene control directo con el adaptador bluetooth local. Utiliza la clase *BroadCastReceiver*, para la comunicación bluetooth, emplea las clases *Handler, AlertDialog*, *onClick*.
- ✓ ***Jugar***. Esta clase es la encargada de la lógica de la aplicación, es una de las clases más importantes ya que esta clase es el manejador de los eventos que se efectúan en el juego. Emplea las clases *OnClickListener*, *onClick*, para los eventos de mover las fichas en el panel del juego se emplean las clases *choiceTouchListener*, *choiceDragListener*.
- ✓ ***Conectar Dispositivos***. Representa la clase que nos permitirá conectar nuestros dispositivos. Extiende de Activity, emplea las clases *onClick*, *BluetoothDevice*(), *OnClickItemListener*, *BroadcastReceiver*.
- ✓ ***ContenedorMensajes***. Representa la clase que nos permite el envío y recepción de los mensajes. Implementa de *Serializable*.
- ✓ ***Casilla***. Esta clase contiene las imágenes y valores para nuestras casillas.
- ✓ ***Fichas***. Esta clase contiene las imágenes y valores para las fichas.
- ✓ ***Elementos***. Clase que sirve como intermediario para acceder a los elementos de la clase Ficha y Casilla.
- ✓ ***GestorSonido***. Clase que nos permite dar sonido al dar click en botones de la aplicación. Emplea la clase *Context*, *AudioManager*, *SoundPool*.
- ✓ ***JuegoAplicación***: Clase que sirve de intermediario para acceder a la Clase *ServicioConexiónBT* y la conexión bluetooth, extiende de *Application*.
- ✓ ***ServicioConexionBT***: Representa la clase que nos permite la conexión bluetooth entre dispositivos. Emplea la clase *BluetoothAdapter*, *BluetoothDevice*, *BluetoothServerSocket*, *BluetoothSocket*.
- ✓ ***Revolver***: Clase que nos permite revolver las fichas.
- ✓ ***RepartirFichas***. Clase que reparte las fichas a cada jugador y las asigna las fichas al montón.
- ✓ ***Tablero***: Clase que contiene los movimientos efectuados en el panel de casillas.
- ✓ ***Turno***. Clase que indica turno del jugador.
- ✓ ***Validar***: Representa la clase para validar un movimiento efectuado por el jugador.

### 6.2.8 Arquitectura del sistema

En este apartado se muestra el esquema de la arquitectura de nuestra aplicación y la descripción de sus elementos.

Como se ha mencionado anteriormente en este documento, nuestra aplicación tiene la habilidad de comunicarse con otro dispositivo simultáneamente, esto se logra por medio del envío y recepción de mensajes (Ver Diagrama 7).

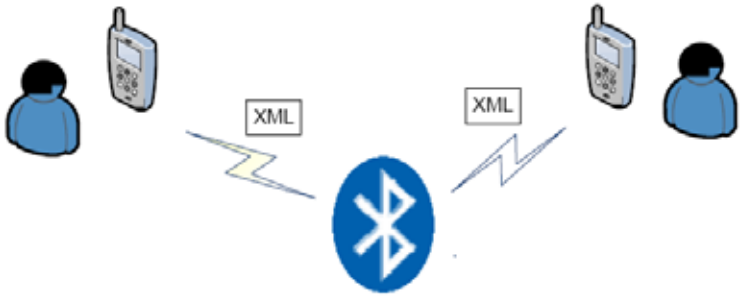



Diagrama 7. Interacción entre los dispositivos y la comunicación bluetooth

Nuestro proyecto emplea una arquitectura en tres capas (ver Diagrama 8) que nos permite dividir al sistema en varias capas independientes.

- **Capa de vistas.** Esta capa contiene todas pantallas de la aplicación.
- **Capa lógica.** Esta capa es la encargada de toda la lógica de la aplicación.
- **Capa de comunicación.** Esta capa se encarga de la transmisión y recepción de la información entre los dispositivos.

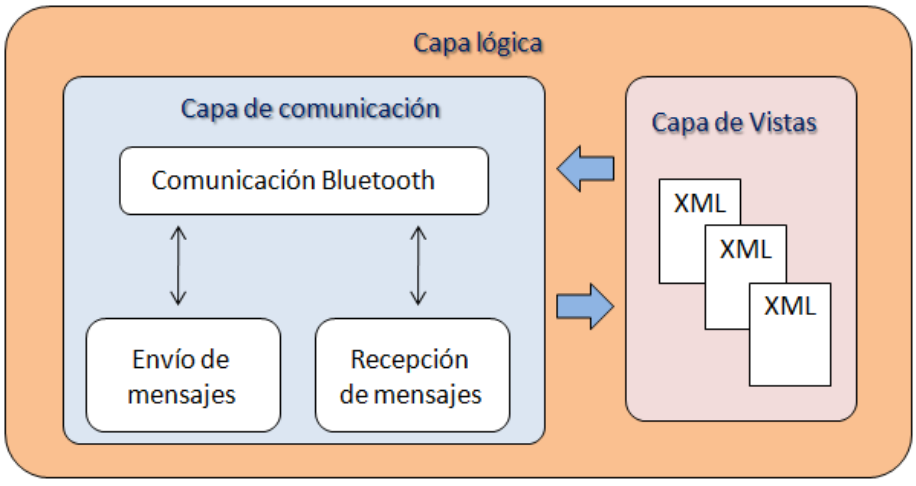



Diagrama 8. Arquitectura de la aplicación Dominó – Tablas de multiplicar

## 6.3 Uso del sistema

### 6.3.1 Iniciar la aplicación

Para usar la aplicación Dominó – Tablas de multiplicar dos usuarios deberán iniciar la aplicación, para esto cada usuario debe seleccionar el ícono Dominó (ver Figura 2).


Figura 2. Ícono de la aplicación.

### 6.3.2 Activar bluetooth

Al iniciar la aplicación, aparecerá un mensaje de activar bluetooth (ver Figura3). Esto sólo sucede si la conexión bluetooth del dispositivo móvil no está habilitada. En caso contrario al inicio del juego se mostrará el menú principal (ver Figura 4). En caso de que la conexión bluetooth no se active, los dispositivos no podrán conectarse vía bluetooth.


Figura 3.Habilitar conexión bluetooth.

La pantalla del menú principal (ver Figura 4) mostrará las opciones conectar, jugar y salir. El botón de jugar permanecerá desactivado hasta que se establezca la conexión entre los dispositivos.


Figura 4. Menú principal de la aplicación.

### 6.3.3 Conectar

Cuando el usuario oprima el botón de conectar en el panel principal (Ver Figura 5). La aplicación mostrará una lista de los nombres de los dispositivos que se encuentran disponibles para poder jugar. El usuario deberá seleccionar algún dispositivo con el cual se desee conectar para poder iniciar el juego, para esto el usuario deberá conocer el nombre del dispositivo móvil del jugador con el cual desea jugar (ver Figura 6).


Figura 5. Lista de dispositivos vinculados

Una vez seleccionado el nombre del dispositivo móvil con el que se desea conectar, la aplicación mostrará la pantalla de la Figura 6.


Figura 6. Conectando con dispositivo móvil.

Si la conexión fue exitosa, la aplicación mostrará la pantalla de la Figura 7. La aplicación desactivará el botón Conectar y habilitará el botón Jugar. Si la conexión no se puede realizar, la aplicación mostrará al usuario el mensaje de que no se pudo conectar con el dispositivo móvil elegido.

### 6.3.4 Jugar

Cuando se establece la conexión vía bluetooth entre los dispositivos, el jugador deberá seleccionar la opción Jugar del Menú Principal. Basta con que un jugador oprima el botón jugar para comenzar el juego.


Figura 7. Conexión exitosa.

### 6.3.5 Salir

En caso de que el usuario desee salir de la aplicación, deberá seleccionar la opción salir del menú principal (ver Figura 8)  y finalizará el juego.


Figura 8. Salir del juego.

## 6.4 Hardware y software necesario

### 6.4.1 Tecnología para el desarrollo de la aplicación

En este apartado se describen las características del hardware y software donde se desarrolló e instaló la aplicación. Además se presentará la estructura del directorio de la aplicación, el proceso de instalación y las pruebas que se realizaron en el desarrollo del proyecto.

#### 6.4.1.1 Hardware

Para el desarrollo y pruebas de la aplicación se empleó el siguiente hardware:

- ✓ Computadora portátil Samsung
  - 3GB de memoria RAM,
  - Procesador Dual Core,
  - Sistema Operativo Windows/Linux.

- ✓ *Dispositivos móviles*: Se utilizaron para las pruebas durante el desarrollo del juego.

- Samsung Galaxy Ace
  - Procesador de 800 MHz,
  - Pantalla LCD HVGA 3.5
  - Sistema Operativo Android 2.2.1.(Froyo)

- Samsung Galaxy Mini
  - Procesador de 600 MHz,
  - Pantalla LCD QVGA 3.13
  - Sistema Operativo Android 2.2.1.(Froyo)

- Samsung Galaxy Pocket
  - Procesador de 850 MHz,
  - Pantalla LCD QVGA Full-Touch 75.6mm (3.0")
  - Sistema Operativo Android 4.1.2 (Jelly Bean)

#### 6.4.1.2 Software

El software utilizado durante el desarrollo del proyecto es el que se describe en los apartados siguientes:

- **Java SE** es la plataforma utilizada en el proyecto, versión 7.
- **Android SDK**[7] es un kit de desarrollo que provee las bibliotecas y las herramientas de desarrollo necesarias para construir, probar y depurar aplicaciones para Android. El emulador de Android permite elegir entre distintos terminales móviles y la versión del sistema operativo. Para la realización del proyecto se empleó Android SDK para Windows.

[7] Del inglés *Standard Edition Kit.*

- **ADT Plugin para Eclipse**. Se trata de un plugin[8] diseñado por Android para el IDE de Eclipse que incluye un conjunto completo de desarrollo y herramientas de depuración. Entre sus funciones se encuentran configurar rápidamente nuevos proyectos y crear interfaces gráficas.
- **IDE Java Eclipse (JUNO)**. Java Eclipse JUNO es un entorno de desarrollo integrado (IDE, Integrated Development Environment), multiplataforma de código abierto que facilita las tareas de edición, compilación, ejecución y depuración de programas. Éste fue el principal entorno de desarrollo utilizado, debido a que aporta un plugin (ADT plugin) para Eclipse que extiende la funcionalidad de éste y facilita el desarrollo de aplicaciones para Android.
    - **Editor de código para la creación y configuración XML**. Dispone de herramientas de desarrollo de Android que permiten: tomar capturas de pantalla, depuración y visualización del sistema en desarrollo. Los editores de código XML se pueden crear dentro del IDE eclipse.
    - **Interfaces gráficas.** Permiten el desarrollo de componentes visualmente.
- **Emulador de Android.** Permite elegir entre distintos terminales móviles y la versión del sistema operativo.

#### 6.4.1.3 Proceso de instalación del software

En esta sección se describe el proceso de instalación del software para la implementación del proyecto terminal.

- **Instalar Eclipse Juno**

La versión que se utilizó fue Eclipse Juno, se puede descargar en el siguiente enlace https://www.eclipse.org/downloads/ . Una vez descargado sólo se necesita descomprimir el archivo y ejecutar Eclipse para poder utilizarlo.

- **Instalar Android SDK**

Para la instalación de Android SDK se necesita descargar del siguiente enlace http://developer.android.com/sdk/index.html#win-bundle la última versión del SDK para la plataforma en la que se desarrollará, en este caso será para Windows.

- **Instalar ADT plugin de Android**

1. Para la instalación del plugin ADT de Android, se tendrá que abrir la aplicación Eclipse, posteriormente seleccionar la opción *Help>Install New Software.*
2. A continuación pulsar el botón *Add*. Aparecerá una ventana como la de la **Figura 9**.

[8] *Android Developmet Kit* o paquete de desarrollo de Android.

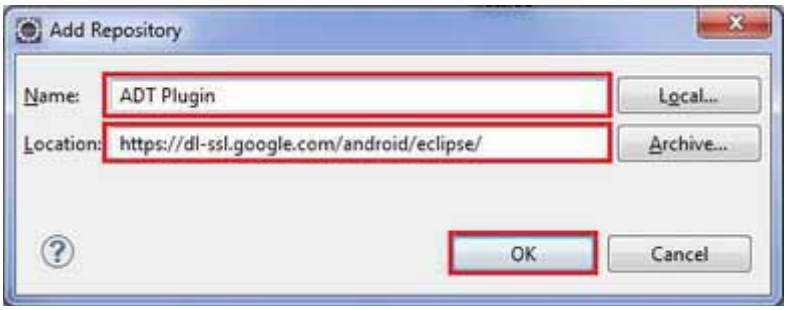



Figura 9. Agregar repositorio.

*3.* Escribir los siguientes datos en el campo de Name->ADT plugin y en el campo de Location -> https://dl.ssl.google.com/android/eclipse/, como se observa en la **Figura 10**. Se deberá habilitar las dos opciones que se muestran, dar click en Next.

4. Se aceptarán los términos de las licencias y se pulsa *Finish.* Entonces empezará a descargar todo lo necesario para que funcione el plugin.

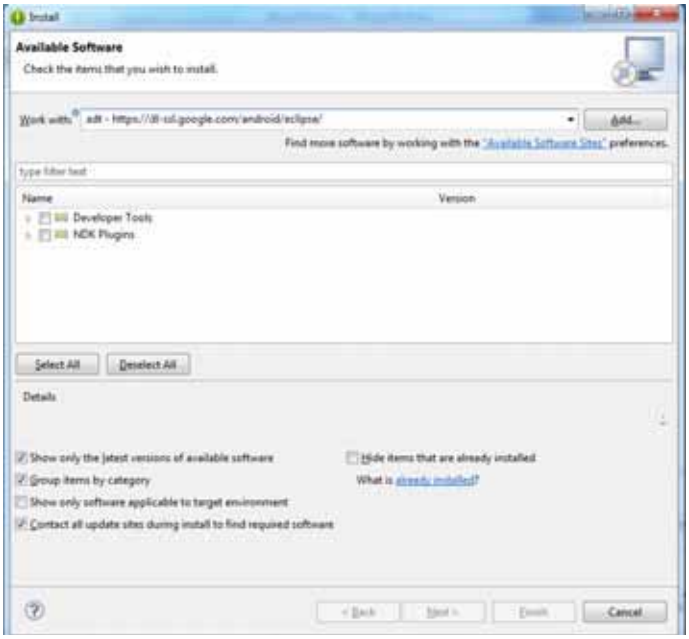



Figura 10. Agregar *plugin ADT para Android*

5. Por último se debe indicar dónde está el *Android SDK,* se selecciona la opción *window* del menú de herramientas*, window>preferences>Android.*
6. Por último aparecerá la lista de todo lo que hemos instalado como se muestra en la **Figura 11**. Pulse OK para terminar la instalación.

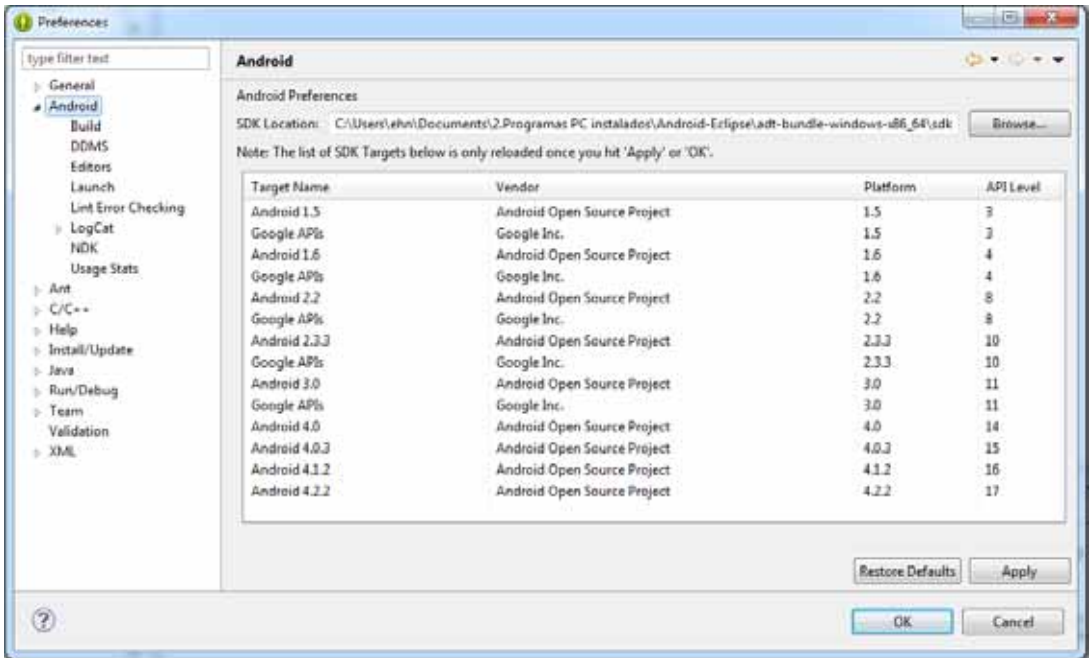



Figura 11. Paquete de Programas instalados

#### 6.4.1.4 Estructura del directorio de la aplicación

La **Figura 12** muestra la estructura del directorio del proyecto Dominó – Tablas de multiplicar:

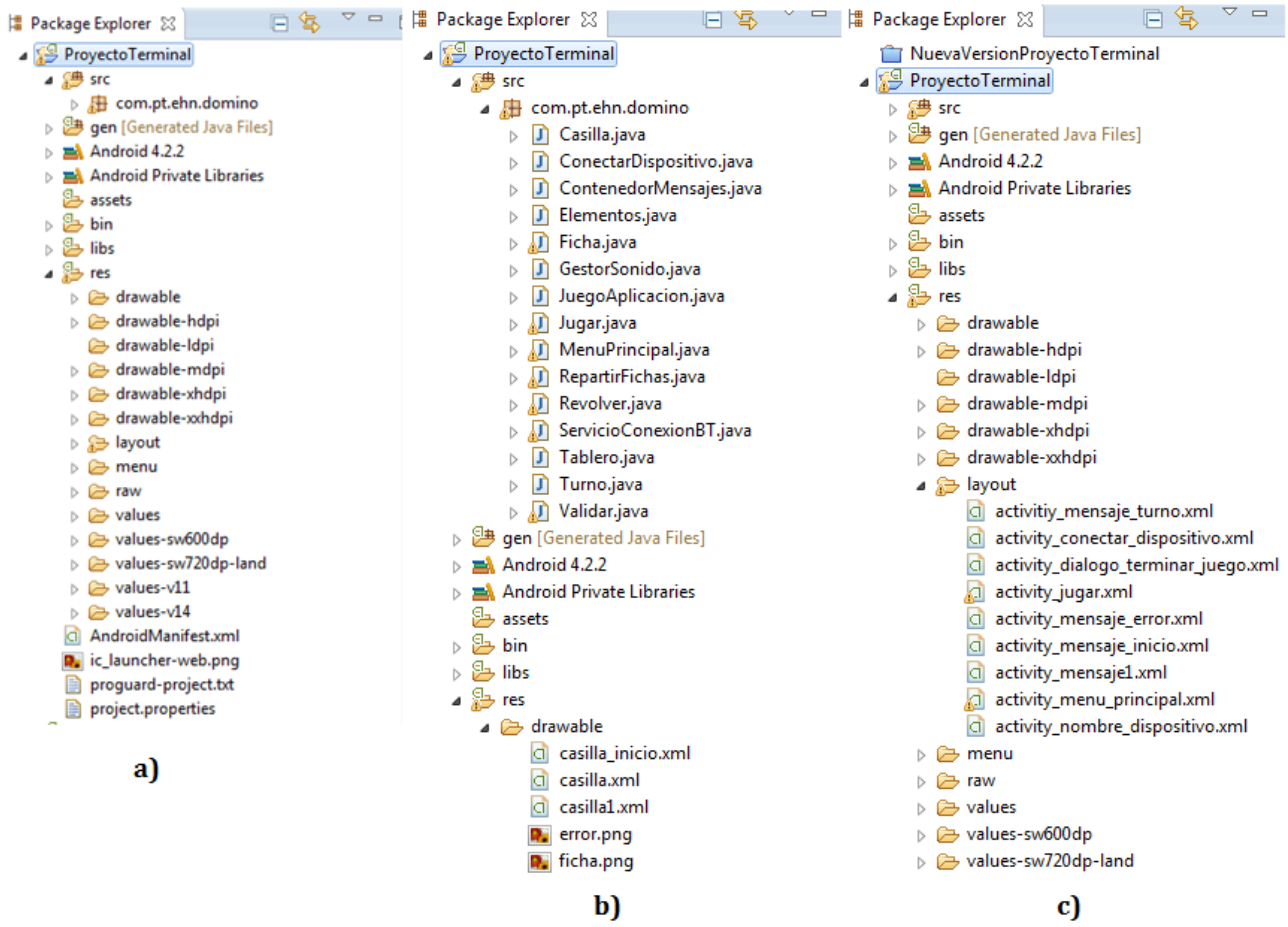



Figura 12. (a, b, c) Estructura jerárquica de la aplicación Dominó – Tablas de multiplicar.

A continuación se hace una breve descripción del contenido de la **Figura 12**.

- **src/**: La carpeta de contenido (*source folder*) incluye todas aquellas clases *Java* generadas para el funcionamiento de la aplicación Dominó - Tablas de Multiplicar.

- **gen/**: Contiene archivos de Java generados por ADT *plugin*. El *ADT* crea un archivo *R.java*, el cual contiene las referencias a cada uno de los recursos de la carpeta *res*, es mediante este archivo que se pueden invocar los recursos en la clases requeridas.

- **doc/**: Contiene archivos de la documentación del proyecto en formato HTML generados por JavaDoc, dichos archivos se generan a partir del código fuente de nuestra aplicación.

- **Android versión JellyBean/** incluye el archivo *Android.jar.*

- **res/**: Esta carpeta contiene todos los recursos de nuestra aplicación. Está divida en carpetas dependiendo del tipo de archivo. Los archivos que utiliza nuestra aplicación son:

- ✓ ***/drawable***: Contiene aquellas imágenes utilizadas en la aplicación, y archivos XML los cuales contienen el diseño de los mensajes empleados en nuestro proyecto.
- ✓ ***/layout:*** Contiene archivos XML que definen la interfaz gráfica.
- ✓ ***/raw***: Contiene los sonidos empleados en nuestro proyecto.
- ✓ ***/values***: Contiene archivos XML que describen color, dimensión, texto y estilo que puede tener los elementos utilizados en cada actividad como pueden ser botones, etiquetas, layouts, etc.

- **AndroidManifest.xml**: Es un archivo de configuración que es generado automáticamente por el IDE. Es quizá uno de los archivos más importantes requeridos para cualquier aplicación Android, está localizado en la raíz del directorio del proyecto, y su función es describir los valores globales del proyecto, incluyendo los componentes de la aplicación (actividades, servicios, etc.). Es importante mencionar que al declarar un componente, se declaran al mismo tiempo algunos de sus atributos como son: el momento de iniciar el componente, la posición de la pantalla que adoptará una vez iniciado el componente.

Cabe mencionar que al generar un proyecto Android, este genera automáticamente directorios que no se pueden modificar como son: /gen que contiene archivos de configuración BuildConfig.java y R.java. Al realizar algún cambio al proyecto estos archivos se van modificando automáticamente. (ver **Figura 13**).

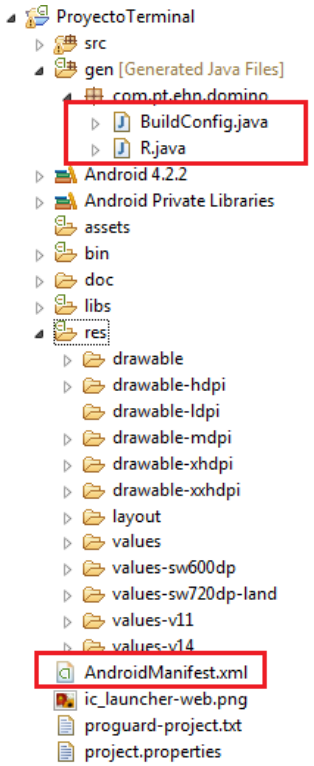



Figura 13. Estructura automática al generar un proyecto Android.

### 6.4.2 Instalación de la aplicación en el dispositivo móvil

1. Una vez terminada la aplicación, se necesita ubicar el archivo *Dominó.apk* para instalarla en el dispositivo móvil. Este archivo lo podemos ubicar en nuestra carpeta */bin* de nuestro proyecto.
2. Debemos copiarlo y guardarlo en una de la tarjeta de almacenamiento externa en cada uno de los dispositivos móviles.
3. Para la instalación del archivo en el dispositivo móvil, utiliza el explorador de archivos y busca el archivo *Domino.apk* . Una vez ubicado se deberá seleccionar y luego nos aparecerá una ventana donde nos pregunta si deseamos instalarlo, tendremos que dar aceptar, una vez terminado el proceso, se podrá iniciar la aplicación.

# 6 Resultados

Al finalizar la codificación; la aplicación fue probada en dos dispositivos móviles con una versión de Android 4.1.2. A continuación se muestra la ejecución de nuestra aplicación.

**Vistas de la aplicación en ejecución**

La **Figura 14** muestra los dos dispositivos que se utilizaron, y en el panel principal de cada dispositivo se encuentra el ícono de nuestra aplicación desarrollada. Para poder acceder a nuestra aplicación debemos dar doble clic en el ícono.


Figura 14. Acceso directo de nuestra aplicación.

La **Figura 15**, muestra la pantalla de inicio de la aplicación..


Figura 15. Pantalla menú principal

Cuando el usuario oprima el botón de conectar en el panel principal, a aplicación mostrará una lista de los nombres de los dispositivos que se encuentren disponibles para poder jugar. El usuario deberá seleccionar algún dispositivo con el cual se desee conectar para**Figura 16** poder iniciar el juego, para esto el usuario deberá conocer el nombre del dispositivo móvil del jugador con el cual desea jugar (ver **Figura 16**a).
Una vez seleccionado el nombre del dispositivo móvil con el que se desea conectar, la aplicación mostrará la pantalla de la **Figura 16**b.


a) b)

Figura 16. a) Lista de dispositivos vinculados. b) Conectando con dispositivo móvil.

Si la conexión fue exitosa, la aplicación mostrará la pantalla de la **Figura 17**. La aplicación desactivará el botón Conectar y habilitará el botón Jugar. Si la conexión no se puede realizar, la aplicación mostrará al usuario el mensaje no se pudo conectar con el dispositivo móvil elegido.

Cuando se establece la conexión vía bluetooth entre los dispositivos. El jugador deberá seleccionar la opción Jugar del Menú Principal. Basta con que un jugador oprima el botón jugar para comenzar el juego.


Figura 17. Conexión exitosa.

Una vez conectados los dispositivos, los usuarios pueden jugar, se mostrará la pantalla principal del juego (**Figura 18**) en cada uno de los dispositivos. Esta pantalla le muestra a cada usuario, el tablero de juego con 28 casillas y 7 de las fichas. Mostrará en la pantalla el nombre del jugador al que le toca el turno actual para iniciar el juego, así como los botones de fichas del montón y pasar turno. Para poder visualizar las 7 fichas, el usuario deberá mover el scroll de la barra horizontal de fichas.


a) b)

Figura 18. a) Pantalla principal del jugador 1. b) Pantalla principal del jugador 2.

El jugador que comience el juego tendrá el turno de "Jugador 1" (ver **Figura 18**a), mientras que su adversario tendrá el turno de "Jugador 2" (ver **Figura 18**b).

Para colocar la primera ficha en el tablero, la ficha seleccionada se deberá colocar en la casilla de inicio que será diferente a las demás casillas. La **Figura 19** muestra que si un jugador tira una ficha en el tablero y está es correcta se visualizará el movimiento en el otro dispositivo, de lo contrario no se podrá colocar la ficha en la casilla.


a) b)

Figura 19. a) Jugador 1 realiza movimiento válido. b) Movimiento del Jugador 1, en la pantalla del Jugador 2.

Cuando el turno del "Jugador 1" termina la aplicación mostrará el turno del jugador a quién le toca tirar. (**Figura 20**). En este caso el turno es para "Jugador 2". Como se puede observar en esta, el jugador 2 selecciona una ficha válida y la coloca en el tablero. Posteriormente la aplicación válida el movimiento del jugador 2 y si es correcto, actualiza el turno y el tablero del jugador 1.


a) b)

Figura 20. a) Jugador 2 realiza movimiento. b) Se visualiza movimiento del Jugador 1.

La ronda de juego continúa como se puede ver en **Figura 21**, **Figura 22**, **Figura 23** y **Figura 24**. En caso de que un jugador no tenga una ficha válida para realizar el movimiento, este deberá tomar una ficha extra del botón Ficha, esto lo podrá realizar hasta que encuentre una ficha válida, o si lo desea podrá pasar el turno a su oponente.

a) b)

Figura 21. a) Movimiento Jugador 2. b) Pantalla del Jugador 1.

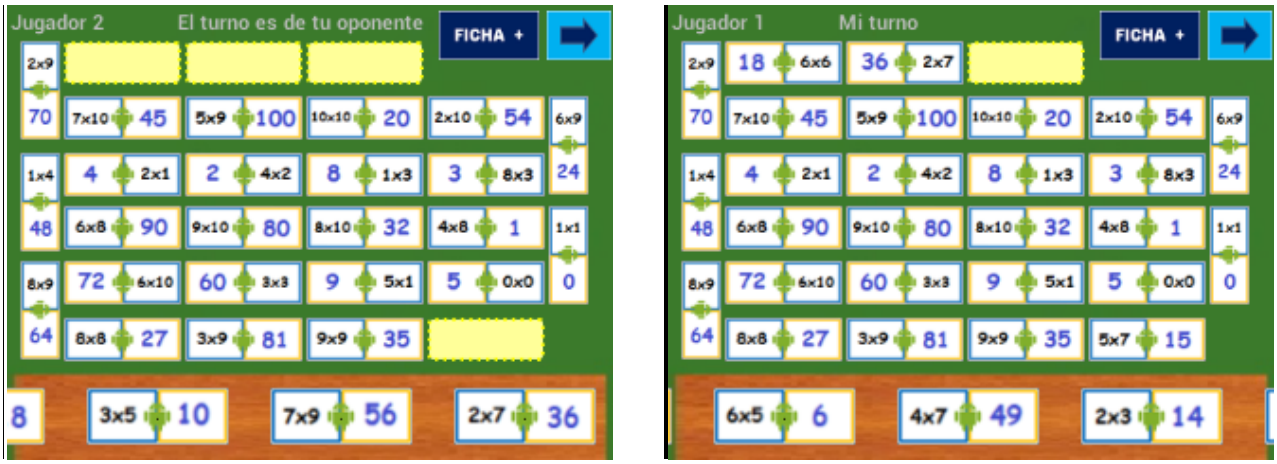
La Figura 22 y la Figura Figura 23 muestra que el jugador 2, ha tomado ficha del montón.


a) b)

Figura 22. a) Movimiento Jugador 1. b) Jugador 2 toma ficha del montón.


a) b)

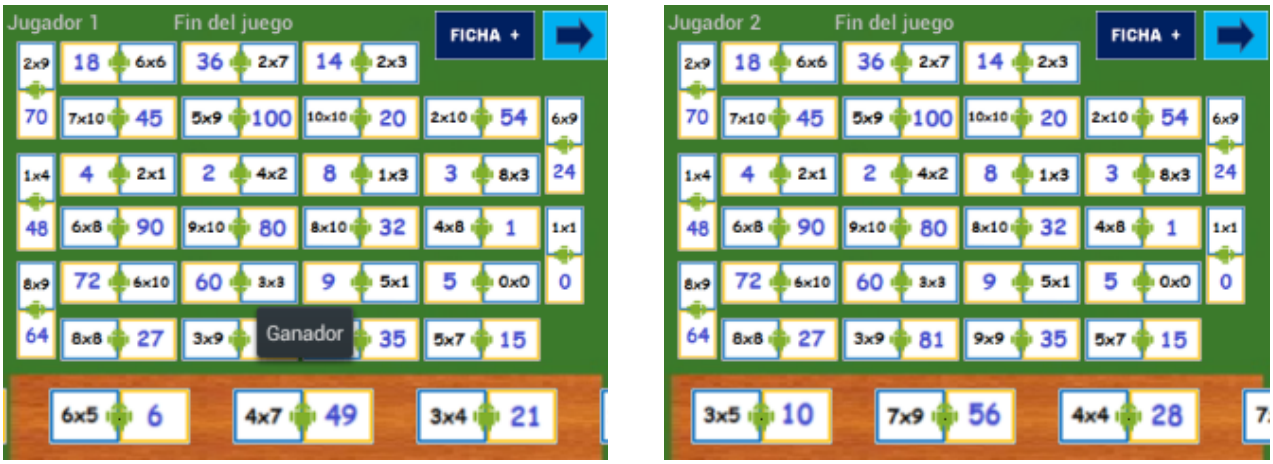
Figura 23. a) Movimiento Jugador 1. b) Jugador 2 toma ficha del montón.




a) b)

Figura 24. a) Movimiento Jugador 2. b) Pantalla del Jugador 1.

Cuando el jugador realice el último movimiento en el tablero. La aplicación validará que jugador tienen el menor número de fichas y enviará el mensaje “Ganador “ al jugador que ganó la partida de juego, así como el letrero de Fin de juego (ver **Figura 25**).




a) b)

Figura 25. a) Gano ronda de juego el Jugador 1. b) Envía mensaje “Fin de juego” pantalla Jugador 2.

Cuando un jugador no tenga una ficha para realizar un movimiento válido, el jugador podrá tomar una ficha del montón para continuar. Esto solo ocurrirá si hay fichas del montón (ver **Figura 26**).


Figura 26. Botón tomar ficha del montón.

El jugador podrá pasar turno,  cuando no tenga una ficha válida para colocar en el tablero, o bien, cuando el jugador no desee colocar ninguna ficha en el tablero (ver **Figura 27**).


Figura 27. Botón pasar turno.

# 7 Análisis y Discusión de Resultados

En esta sección se describen los límites, características y resultados que se obtuvieron de nuestra aplicación.

Una vez finalizado el desarrollo de nuestro proyecto y en base a los objetivos planteados, podemos decir que una limitante del proyecto se basa a que inicialmente se tenía contemplado el poder interactuar con mínimo de dos dispositivos y un máximo de cuatro. Se logró la comunicación mediante el uso de dos dispositivos y por cuestiones de tiempo no se pudo desarrollar la comunicación para un máximo de cuatro dispositivos.

Durante el desarrollo de la aplicación, fueron realizándose pruebas que nos indicaban el avance del proyecto. (Ver Tabla 1)

| PRUEBA | RESULTADO |
|---|---|
| Diseño capa de vista. | Para el diseño de la capa de vista, se tuvo que probar en el emulador de Android, este nos indicaba el aspecto visual de nuestra aplicación. |
| Diseño de la comunicación. | Esta prueba fue una de las más importantes y complicadas a la vez, debido a que el emulador de Android, no dispone de comunicación vía bluetooth que pueda ser visualizada en la computadora. Por lo tanto para efectuar esta prueba, cada vez que se tenía que probar el avance de la aplicación, se tuvo que instalar el archivo .apk de la aplicación en cada dispositivo, para poder visualizar los resultados. |
| Desarrollo del juego | Al igual que en el diseño de la comunicación. Al querer ver el avance del desarrollo del juego se tuvo que instalar el archivo .apk en cada dispositivo y así poder ver los avances del desarrollo de la aplicación. Cada vez que se tuvo que probar la aplicación en los dispositivos, se requiere de bastante tiempo en comparación si se usa el emulador de Android. |

Tabla 1. Pruebas y resultados en el desarrollo del proyecto.

El rendimiento de ambos celulares fue fluido durante la ejecución de la aplicación, sin embargo al iniciar la programación del modulo de la comunicación bluetooth, surgieron algunos inconvenientes como fue el que se detenía la aplicación, o no dejaba avanzar debido a que el dispositivo se trababa y tenía que reiniciar tanto la aplicación como el dispostivo, estos errores surgían porque no se estaba programando correctamente la comunicación bluetooth. Dichos errores fueron detectados y corregidos, y una vez finalizado el desarrollo de la aplicación no se han vuelto a detectar dichos errores.

# 8 Conclusiones

En este apartado se describen las conclusiones que se obtuvieron al finalizar el desarrollo del proyecto.

El objetivo general de nuestro proyecto fue implementar un juego multiusuario para dispositivos móviles con sistema operativo Android, tomando como caso de estudio una versión del juego de mesa Dominó, podemos asumir que este objetivo se logró parcialmente.

En la Tabla 2 se detallan los objetivos propuestos alcanzados.

| OBJETIVO | RESULTADO |
|---|---|
| Módulo gestión del juego. | Se creó un módulo gestión del juego que se encarga de la funcionalidad el juego. Este objetivo se cumplió parcialmente. Esto se debe a que la aplicación aún se encuentra en su etapa inicial. |
| Módulo gestión de mensajes. | Se creó el módulo encargado del paso de mensajes para la comunicación entre los jugadores de nuestra aplicación. Este objetivo se cumplió completamente, y por medio de este modulo se envían y reciben los mensajes entre ambos dispositivos. |
| Módulo gestión de comunicación entre dispositivos. | Se creó un módulo conexión bluetooth, el cual nos permite comunicarnos con otro dispositivo. Este módulo se definió la conexión de un mínimo de 2 personas y un máximo 4 para la comunicación bluetooth. Por el momento nuestra aplicación solo tiene la posibilidad de comunicación entre dos dispositivos. Por lo tanto se puede decir que nuestro objetivo se alcanzó de manera parcial. |
| Módulo de interfaz gráfica del juego. | Se creó un módulo de interfaz gráfica, el cual nos permite visualizar las distintas pantallas de nuestra aplicación.<br>La aplicación fue diseñada para que la interfaz gráfica tuviera cierta adaptabilidad a las dimensiones de la pantalla de cada dispositivo, es decir, la distribución de los gráficos estará en función de las dimensiones de la pantalla del dispositivo. Sin embargo, dada la tecnología usada, la aplicación no puede adaptarse al 100% de las dimensiones de pantalla. Los gráficos de la aplicación se adaptarán a pantallas con dimensiones superiores, pero con dimensiones inferiores la aplicación de adaptara de una manera aceptable. El objetivo se logro completamente. |

Tabla 2. Objetivos propuestos alcanzados.

# 9 Perspectivas

Con la finalidad de incrementar la robustez de nuestra aplicación es deseable implementar otras funcionalidades que hagan de este juego una aplicación más completa y también con el objetivo de poder hacer más atractivo el juego al usuario y alargar la vida de la aplicación.

- ✓ Diseñar un módulo que controle las puntuaciones de los jugadores.
- ✓ Diseñar un módulo de ayuda, el cual contenga una lista tablas de multiplicar del 1 al 10, que va a servir como repaso o memorización, para el usuario.
- ✓ Diseñar un módulo de preferencias, el cual controlará las funcionales del juego (por ejemplo el sonido de la aplicación, el número de fichas para cada jugador, el número de fichas del montón que puede tomar cada jugador.
- ✓ A pesar de que la versión original del juego fue diseñada para un mínimo de 2 y un máximo de 4 jugadores, la implementación de este proyecto permite la conexión con dos dispositivos. Por lo cual en un futuro se podría implementar dicha funcionalidad permitiendo conectar a un máximo de cuatro jugadores.
- ✓ Nuestra aplicación hace uso de la comunicación bluetooth para la conexión de nuestros dispositivos. Otra línea futura que podemos incluir, sería realizar la conexión a través del protocolo IP, mediante una red WiFi. La ventaja de esta opción es que ofrece más seguridad de transferencia de datos que bluetooth.

# Referencias Bibliográficas


[1] M. Gargenta, *Learning Android*, Primera Edición, California, O'Reilly Media, 2011.

[2] AndEngine, (2012, 05, 02), *AndEngine*, [en línea], Disponible: http://www.andengine.org/

[3] I. A. Pérez Segura, *"Implementación de un sistema de noticias con dualidad en la obtención de la información: vía aérea y vía Bluetooth"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2007.

[4] A. W. García Ríos, *"Sistema de transmisión y recepción vía Bluetooth para votación"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2008.

[5] M. D. Cruz Rodríguez, *"Aplicación Android para la sincronización de las tareas y el material didáctico de sistemas Moodle"*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autonóma Metropolitana Azcapotzalco, México, 2011.

[6] E. J. Gil Izquierdo, *"ChessAsin, servidor de ajedrez por correspondencia"*, Tesis, Departamento de Informática, Universidad Carlos III de Madrid, España, 2011. [En línea] Disponible: http://e-archivo.uc3m.es/handle/10016/13635

[7] D.M. Catalán, *"POKERAN: Juego de poker"*, Tesis, Departamento de Informática, Universidad Autonóma de Barcelona, 2012. [En línea] Disponible: http://ddd.uab.cat/pub/trerecpro/2013/hdl_2072_208418/MolinaCatalanDavidR-ETIGa2011-12.pdf

[8] J.S. pizarro, "Desarrollo de un Sistema de Juego Ubicuo bajo Plataforma Android":, Tesis, Facultad de Ingeniería, Universidad de Talca, 2012. [En línea] Disponible: http://www.slideshare.net/jpizarrom/memoria-18433248

[9] Proceso Unificado [En línea] Disponible: http://es.wikipedia.org/wiki/Proceso_Unificado

[10] C. Larman, *UML y Patrones Introducción al análisis y diseño orientado a objetos*, Prentice Hall, noviembre 2004.

# Apéndice A. Listado del API del código fuente desarrollado.

## Package com.pt.ehn.domino

Class Summary

| Class | Description |
|---|---|
| BuildConfig | |
| Casilla | Casilla.java Esta clase contiene las imagenes para nuestras casillas. |
| ConectarDispositivo | |
| ContenedorMensajes | |
| Elementos | Elementos.java Clase que sirve como intermendiario para acceder a los elementos de la clase Ficha y Casilla.. |
| Ficha | |
| GestorSonido | |
| JuegoAplicacion | |
| Jugar | |
| MenuPrincipal | |
| R | |
| R.attr | |
| R.color | |
| R.dimen | |
| R.drawable | |
| R.id | |
| R.layout | |
| R.menu | |
| R.raw | |
| R.string | |
| R.style | |
| RepartirFichas | RepartirFichas.java Representa la clase que reparte las fichas a cada jugador, y asigna las fichas al monton. |
| Revolver | |
| ServicioConexionBT | |
| Tablero | Tablero.java Representa la clase para los movimientos efectuados en el panel de casillas. |
| Turno | Turno.java Clase que indica el turno del jugador. |
| Validar | |

Figura 28. Lista de clases del proyecto Dominó – Tablas de Multiplicar.

**Class Hierarchy**

- java.lang.Object
  - Activity
    - com.pt.ehn.domino.**MenuPrincipal**
  - Activity
    - com.pt.ehn.domino.**Jugar**
  - Activity
    - com.pt.ehn.domino.**ConectarDispositivo**
  - Application
    - com.pt.ehn.domino.**JuegoAplicacion**
  - com.pt.ehn.domino.**BuildConfig**
  - com.pt.ehn.domino.**Casilla**
  - com.pt.ehn.domino.**ContenedorMensajes** (implements java.io.Serializable)
  - com.pt.ehn.domino.**Elementos**
  - com.pt.ehn.domino.**Ficha**
  - com.pt.ehn.domino.**GestorSonido**
  - com.pt.ehn.domino.**R**
  - com.pt.ehn.domino.**R.attr**
  - com.pt.ehn.domino.**R.color**
  - com.pt.ehn.domino.**R.dimen**
  - com.pt.ehn.domino.**R.drawable**
  - com.pt.ehn.domino.**R.id**
  - com.pt.ehn.domino.**R.layout**
  - com.pt.ehn.domino.**R.menu**
  - com.pt.ehn.domino.**R.raw**
  - com.pt.ehn.domino.**R.string**
  - com.pt.ehn.domino.**R.style**
  - com.pt.ehn.domino.**RepartirFichas**
  - com.pt.ehn.domino.**Revolver**
  - com.pt.ehn.domino.**ServicioConexionBT**
  - com.pt.ehn.domino.**Tablero**
  - com.pt.ehn.domino.**Turno**
  - com.pt.ehn.domino.**Validar**

Figura 29. Lista de clases en forma de directorio

com.pt.ehn.domino

## Class Casilla

java.lang.Object
  com.pt.ehn.domino.Casilla

```
public class Casilla
extends java.lang.Object
```

Casilla.java Esta clase contiene las imagenes para nuestras casillas

Version:

1.0

Author:

ehn

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| Casilla() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| int | getCasillaSeleccionada(int id) |
| int | getEstadoCasilla(int posicion) |
| int | getIdCasilla(int posicion) |
| int | getValorCasilla(int posicion, int lado)<br>Enviar 0-izquierda 1-der |
| void | setEstadoCasilla(int posicion, int estado) |
| void | setValorCasilla(int posicion, int lado, int valor) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Figura 30. Clase Casilla

com.pt.ehn.domino

## Class JuegoAplicacion

java.lang.Object
  Application
    com.pt.ehn.domino.JuegoAplicacion

```
public class JuegoAplicacion
extends Application
```

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| JuegoAplicacion() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| ServicioConexionBT | getConnectionService() |
| int | getTipo() |
| void | onCreate() |
| void | setConnectionService(ServicioConexionBT mConnectionService) |
| void | setTipo(int tipo) |

Figura 31. Clase JuegoAplicacion.

com.pt.ehn.domino

## Class Jugar

java.lang.Object
  Activity
    com.pt.ehn.domino.Jugar

---

```
public class Jugar
extends Activity
```

### Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | MOVIMIENTO_INICIAL |
| static int | NO_HAY_MOVIMIENTO |
| static int | SI_HAY_MOVIMIENTO |

### Constructor Summary

Constructors

| Constructor and Description |
|---|
| Jugar() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
|---|---|
| void | activarCasillas() |
| void | activarFichas() |
| void | deshabilitarFichas() |
| void | iniciaJuego() |
| void | onBackPressed() |
| int | posRealFichaMonton(int posFichaMontonSel) |
| void | recepcionMensajes(ContenedorMensajes m) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Figura 32. Clase Jugar

com.pt.ehn.domino

## Class Revolver

java.lang.Object
  com.pt.ehn.domino.Revolver

---

```
public class Revolver
extends java.lang.Object
```

### Constructor Summary

Constructors

| Constructor and Description |
|---|
| Revolver() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
|---|---|
| int[] | getFichasRevueltas() |
| void | revolver() |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Figura 33. Clase Revolver

com.pt.ehn.domino

## Class Ficha

java.lang.Object
    com.pt.ehn.domino.Ficha

---

```
public class Ficha
extends java.lang.Object
```

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| Ficha() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| int | getFichaSeleccionada(int idFicha) |
| int | getIdFicha(int posicion) |
| int | getIdFichaMonton(int posicion) |
| int | getImgFichaPos1(int posicion) |
| int | getImgFichaPos2(int posicion) |
| int | getImgFichaPos3(int posicion) |
| int | getValorFicha(int posicion, int lado) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Figura 34. Clase Ficha

com.pt.ehn.domino

## Class MenuPrincipal

java.lang.Object
    Activity
        com.pt.ehn.domino.MenuPrincipal

---

```
public class MenuPrincipal
extends Activity
```

### Field Summary

Fields

| Modifier and Type | Field and Description |
| --- | --- |
| static java.lang.String | DEVICE_NAME |
| static java.lang.String | mConnectedDeviceName |
| static int | MENSAJE_TOAST |
| static int | MESSAGE_DEVICE_NAME |
| static int | MESSAGE_READ |
| static java.lang.String | TOAST |

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| MenuPrincipal() |

Figura 35. Clase Menú Principal

com.pt.ehn.domino

## Class Elementos

java.lang.Object
    com.pt.ehn.domino.Elementos

---

```
public class Elementos
extends java.lang.Object
```

Elementos.java Clase que sirve como intermendiario para acceder a los elementos de la clase Ficha y Casilla..

**Version:**

1.0

**Author:**

ehn

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| Elementos() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| int | getCasillaSel(int id) |
| int | getEstadoCasilla(int pos) |
| int | getFichaSel(int id) |
| int | getIdBtnPasarTurno() |
| int | getIdBtnTomarFicha() |
| int | getIdCasilla(int posicion) |
| int | getIdFicha(int posicion) |
| int | getIdFichaMoton(int posicion) |
| int | getIdMsgJugador() |
| int | getIdMsgTurno() |
| int | getImagenFichaSel(int casilla, int ficha) |
| int | getValorFicha(int pos, int lado) |
| void | inicializaCasilla(int estado) |
| void | inicializaValoresCasilla(int lado, int valor) |

Figura 36. Clase Elementos

com.pt.ehn.domino

## Class RepartirFichas

java.lang.Object
  com.pt.ehn.domino.RepartirFichas

```
public class RepartirFichas
extends java.lang.Object
```

RepartirFichas.java Representa la clase que reparte las fichas a cada jugador, y asigna las fichas al monton.

Version:

1.0

Author:

ehn

### Constructor Summary

Constructors

| Constructor and Description |
| --- |
| RepartirFichas() |

### Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| int | getEstadoFicha(int posicion) |
| int | getEstadoFichaMonton(int posicion) |
| int | getFichaJug(int posicion) |
| int | getFichaMonton(int posicion) |
| int[] | getFichasJug() |
| int[] | getFichasMonton() |
| int[] | getObtListaFichas() |
| int | getTotFichaMontonRestas() |
| int | getValorFicha(int posicion, int lado) |
| int | getValorFichaMonton(int posicion, int lado) |
| void | repartir(int[] todasFichas, int iniJug, int iniOpp, int iniMonton) |
| void | setEstadoFicha(int posicion, int estado) |
| void | setEstadoFichaMonton(int posicion, int estado) |
| void | setTotFichaMontonRestas(int valor) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Figura 37. Clase RepartirFichas

com.pt.ehn.domino

## Class ContenedorMensajes

java.lang.Object
  com.pt.ehn.domino.ContenedorMensajes

All Implemented Interfaces:

java.io.Serializable

```
public class ContenedorMensajes
extends java.lang.Object
implements java.io.Serializable
```

See Also:

Serialized Form

### Field Summary

Fields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | MENSAJE_ACEPTAR_CONEXION |
| static int | MENSAJE_ENVIAR |
| static int | MENSAJE_GANAR |
| static int | MENSAJE_NUEVO_JUEGO |
| static int | MENSAJE_RECIBIR |
| static int | MENSAJE_SALIR |
| static int | MENSAJE_TERMINAR_JUEGO |

Figura 38. Clase Contenedor de Mensajes

com.pt.ehn.domino

## Class ServicioConexionBT

java.lang.Object
  com.pt.ehn.domino.ServicioConexionBT

---

```
public class ServicioConexionBT
extends java.lang.Object
```

### Nested Class Summary

Nested Classes

| Modifier and Type | Class and Description |
|---|---|
| static class | ServicioConexionBT.State |

### Constructor Summary

Constructors

| Constructor and Description |
|---|
| ServicioConexionBT(Handler handler) |

### Method Summary

Methods

| Modifier and Type | Method and Description |
|---|---|
| void | connect(BluetoothDevice device) |
| void | connected(BluetoothSocket socket) |
| ServicioConexionBT.State | getState() |
| void | setHandler(Handler handler) |
| void | start() |
| void | stop() |
| void | write(byte[] out) |

#### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### ServicioConexionBT

```
public ServicioConexionBT(Handler handler)
```

Figura 39. Clase ServicioConexionBT

com.pt.ehn.domino

## Class ContenedorMensajes

java.lang.Object
  com.pt.ehn.domino.ContenedorMensajes

**All Implemented Interfaces:**

java.io.Serializable

---

```
public class ContenedorMensajes
extends java.lang.Object
implements java.io.Serializable
```

See Also:

Serialized Form

### Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | MENSAJE_ACEPTAR_CONEXION |
| static int | MENSAJE_ENVIAR |
| static int | MENSAJE_GANAR |
| static int | MENSAJE_NUEVO_JUEGO |
| static int | MENSAJE_RECIBIR |
| static int | MENSAJE_SALIR |
| static int | MENSAJE_TERMINAR_JUEGO |

Figura 40. Clase contenedor de mensajes

## Enum ServicioConexionBT.State

java.lang.Object
  java.lang.Enum<ServicioConexionBT.State>
    com.pt.ehn.domino.ServicioConexionBT.State

**All Implemented Interfaces:**

java.io.Serializable, java.lang.Comparable<ServicioConexionBT.State>

**Enclosing class:**

ServicioConexionBT

---

```
public static enum ServicioConexionBT.State
extends java.lang.Enum<ServicioConexionBT.State>
```

### Enum Constant Summary

Enum Constants

| Enum Constant and Description |
|---|
| connected |
| connecting |
| disconnected |

### Method Summary

Methods

| Modifier and Type | Method and Description |
|---|---|
| static ServicioConexionBT.State | valueOf(java.lang.String name)<br>Returns the enum constant of this type with the specified name. |
| static ServicioConexionBT.State[] | values()<br>Returns an array containing the constants of this enum type, in the order they are declared. |

**Methods inherited from class java.lang.Enum**

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

**Methods inherited from class java.lang.Object**

getClass, notify, notifyAll, wait, wait, wait

### Enum Constant Detail

**disconnected**

```
public static final ServicioConexionBT.State disconnected
```

Figura 41. Clase Enum ServicioConexionBT.State