En el presente trabajo se trata de crear una aplicación que simule un juego de Dominó. Este juego, además de popular, es muy versátil y existen muchas variantes en el mundo, unas más populares y conocidas, que otras. Entre su variabilidad se cuenta que se puede jugar con diversos números de fichas, diversas formas de definir un inicio, desarrollo del juego, varias variantes de finalizar partidas y juego, y además, diversos criterios para definir el equipo ganador; unido a esto, las múltiples y diferentes tácticas y estrategias, que se han utilizado en diferentes escenarios, hacen del Dominó uno de los juegos más complejos de automatizar y es, por lo tanto, muy difícil abarcar todas las combinaciones posibles.

Por todo lo anterior, se escogieron varias variantes de los diferentes aspectos del juego. El esquema general del juego consiste en tres etapas fundamentales:

1.- **Inicio del juego**: donde se define, de acuerdo a diversos criterios, quién o qué equipo comienza el juego, y además que tipo de ficha será la primera en tirarse. En esta etapa se repartirán un número de fichas a cada jugador, en dependencia del tipo de juego de dominó a efectuarse.

2. **Desarrollo del juego:** en esta etapa, de acuerdo al tipo de juego acordado, se realizará una o varias partidas para definir un ganador. Cada jugador tirará la ficha, que según su comportamiento o estrategia en el juego, le convenga, siempre y cuando posea al menos una posible de colocar en la mesa.

3. **Final de partida o juego:** en dependenciadel tipo de juego, previamente acordado, se puede ejecutar una partida o varias, hasta que se cumpla un criterio de finalización del juego y/o de cada partida que defina el juego.

Debido al comportamiento del juego, la manera más factible es controlar cada una de estas etapas por separado e incorporarle diferentes comportamientos. Para ello se declararon clases abstractas, para cada una de ellas y diversos comportamientos diseñados como interfaces, para simular las diferentes variantes. Esta manera modular y generalizada permite hacer el diseño escalable, sin comprometer las anteriores opciones ya incorporadas.

Por la necesidad de la interacción entre los diferentes módulos se crearon dos clases que aglutinan los datos fundamentales y funciones, estas son **MESA**, donde se aglutinan los datos necesarios para el juego, como las fichas disponibles y las que se van tirando por cada jugador, así como, los datos e informaciones que se van generando en el juego y que son necesarios acceder y/o modificar. La otra clase es **DOMINO**, donde se declaran y controlan las diferentes etapas del juego y muestran los resultados parciales.

A continuación se mostrarán las diferentes variantes tomadas en cuenta para el desarrollo de la aplicación:

**Tipo de juego**: Doble 6 0 Doble 9

**Cantidad de jugadores:** estos pueden ser a partir de 2.

**Tipos de jugadores**: Bota Gorda, Aleatorio, Tramposo

**Inicio del juego:** Salida Aleatoria, Mayor Ficha, Pares o nones

**Acción ante el Pase:** Pase usual, Robar ficha si no lleva.

**Final de partida:** Fin usual, Fin doble pase

**Final del juego**: Por número de partidas, Acumulación de puntos

**Sistema de puntaje:** Fichas por puntos (se cuentan los puntos que tiene cada ficha), Cantidad de fichas (cada ficha cuenta como 1 punto, los dobles cuentan como 2 puntos).

Las opciones aleatorias, aunque es raro en un juego, no obstante se incorpora como variante dada a la computadora.

La programación se realizó en plataforma Windows con .NET6.0, con salida a consola, cada una de las variantes se corresponden con interfaces o clases abstractas, para encapsular comportamientos comunes y permitir incorporar nuevas variantes.