**Documento de Arquitectura de Software**

**Alumno—**

Mauricio Bismarck Murillo Valle

Jose Fernando Parra Rodriguez

**ID—**

181911

182019

**Fecha—**

3 de Diciembre de 2019

**Materia—**

Arquitectura de Software

**Profesor—**

Gilberto Borrego Soto

**Tabla de contenidos**

1. **Introducción.**
   1. Propósito.
   2. Alcance.
   3. Definiciones, Acrónimos y abreviaciones.
2. **Representación Arquitectónica.**
3. **Objetivos arquitectónicos y restricciones de diseño.**
4. **Vista de escenarios**
   1. Diagrama de casos de uso
   2. CU1
   3. CU2
   4. CU3
5. **Vista lógica**
   1. Diagrama de dominio
   2. Diagramas de secuencia a nivel de análisis
6. **Vista de implementación**
   1. Diagrama de componentes
   2. Diagrama de clases de cada componente
   3. Realización de casos de uso
7. **Vista de despliegue**
   1. Diagrama de despliegue

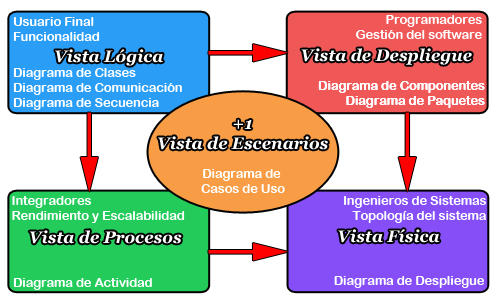
**Documento de arquitectura de software**

1. **Introducción**

Este documento proporciona una descripción general de alto nivel y explica la arquitectura del juego Timbiriche.

El documento define los objetivos de la arquitectura, los casos de uso admitidos por el sistema, los estilos arquitectónicos y los componentes que se han seleccionado. El documento proporciona una justificación para las decisiones de arquitectura y diseño tomadas desde la idea conceptual hasta su implementación.

* 1. **Propósito**

Este documento proporciona un descripcion arquitectonica completa del sistema Timbiriche. Presenta varias vistas arquitectónicas diferentes para representar los diferentes aspectos del sistema. Para representar el software con la mayor precisión posible, la estructura de este documento se basa en la vista arquitectónica de arquitectura “4 + 1” de Philippe Kruchten. 

* 1. **Alcance**

El alcance de este documento es explicar la arquitectura del juego Timbiriche; describe los diversos aspectos del diseño del sistema, que se consideran arquitectónicamente significativos. Estos elementos y comportamientos son fundamentales para guiar la construcción del sistema y para comprender este proyecto en su conjunto.

* 1. **Definiciones, acrónimos y abreviaciones**

1. **Representación arquitectónica**

Este documento detalla la arquitectura utilizando las vistas definidas en el modelo “4 + 1”[Kruchten]. Las vistas utilizadas para documentar el sistema son:

* **Vista de escenarios**
  + **Audiencia:** Todas las partes interesadas del sistema, incluidos los usuarios finales.
  + **Área:** Describe el conjunto de escenarios y / o casos de uso que representan alguna funcionalidad central significativa del sistema.
    - **Artefactos relacionados:** Modelo de casos de uso, documentos de casos de uso.
* **Vista lógica**
  + **Audiencia:** Diseñadores/Programadores
  + **Área:** Describe las entidades clave.
  + **Artefactos relacionados:** Diagramas de clases.
* **Vista de implementación**
  + **Audiencia:** Diseñadores/Arquitectos.
  + **Área:** Describe la organización en componentes del sistema, así como la comunicación entre ellos.
  + **Artefactos relacionados:** Diagrama de componentes.
* **Vista de despliegue**
  + **Audiencia:** Gerentes de implementación.
  + **Área:** Describe el mapeo del software en el hardware y muestra los aspectos distribuidos del sistema. Describe las posibles estructuras de implementación, al incluir escenarios de implementación conocidos y anticipados en la arquitectura, permitimos que los implementadores hagan ciertas suposiciones sobre el rendimiento de la red, la interacción del sistema, etc.
  + **Artefactos relacionados:** Diagrama de despliegue.

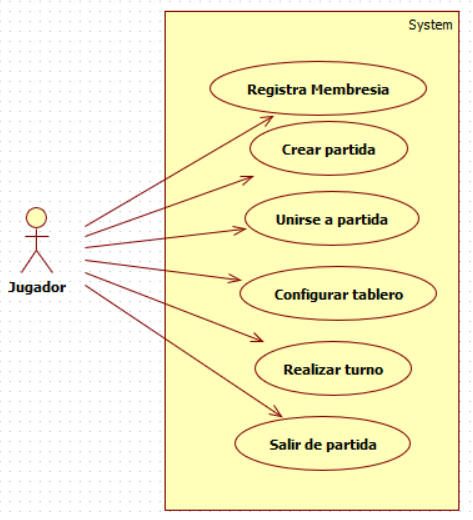
1. **Objetivos arquitectónicos y restricciones de diseño.**

Describir cual son los objetivos de su arquitectura(que no se funcional), así como las restricciones, Por ejemplo: el estilo arquitectónico, el uso de sockets, etc.

1. **Vista de escenarios**

El propósito de esta vista es proporcionar un contexto adicional que rodee el uso del sistema y las interacciones entre sus componentes. A continuación, se presenta el diagrama de casos de uso, así como la descripción de los mismo que son arquitectónicamente significativos del sistema.

* 1. **Diagrama de casos de uso**

****

* 1. **CU1 - Registrar jugador**

Este caso de uso permite al jugador registrarse dentro del sistema timbiriche.

* 1. **CU2 - Crear partida**

Este caso de uso permite al jugador crear una partida dentro del sistema de timbiriche.

* 1. **CU3 - Unirse a partida**

Este caso de uso permite al jugador unirse a una partida creada en el sistema de timbiriche.

* 1. **CU4 - Configurar tablero**

Este caso de uso permite al jugador configurar los colores de los jugadores en su tablero.

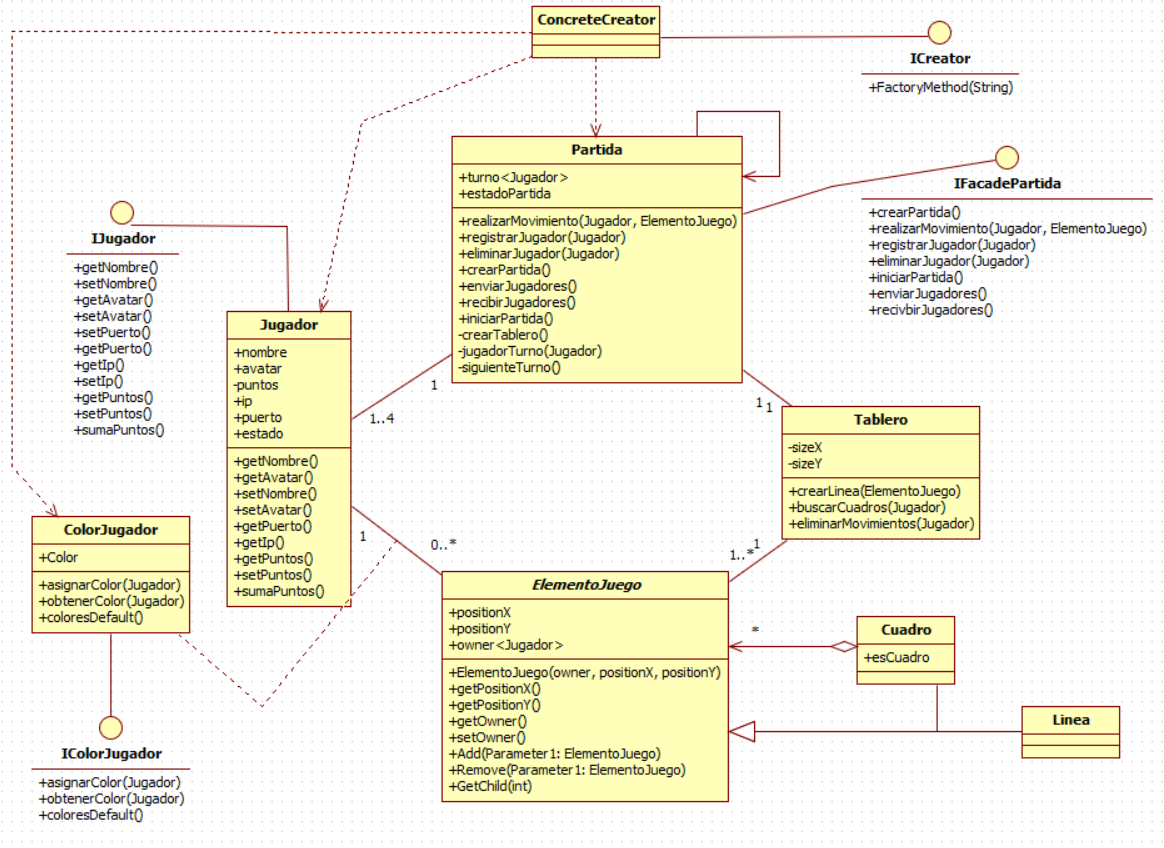
* 1. **CU5 - Realizar turno**

Este caso de uso permite al jugador realizar un turno dentro del juego de timbiriche.

* 1. **CU6 - Salir de partida**

Este caso de uso permite al jugador salirse de la partida de un sistema de timbiriche.

1. **Vista lógica**
   1. **Diagrama de dominio**

****

Representación del diagrama de clases para el sistema timbiriche, el el cual se necesito crear una clase partida que contuviera todo lo relacionado con gestionar los turnos de los jugadores y los movimientos realizados en el tablero, además se necesito crear una clase jugador para representar a cada uno de los cuatro posibles jugadores que se pueden unir a timbiriche dentro de una sola partida, se creó una clase tablero, line y cuadro que se integraron con un patrón composite para representar el tablero del sistema timbiriche.

Además se necesito crear una clase de descripción de relación entre jugador y elemento juego para representar el color de cada uno de los elementos de juego que se representaría más tarde en la parte gráfica de timbiriche, se necesito la creacion e implementacion de las interfaces IJugador, IFacadeJugador e IColorJugador para representar las fachadas en el sistema, así como también la implementación de la interface creator para lograr crear la implementación de la fábrica con la clase ConcreateCreator.

Patrón fábrica: El patrón fábrica es un patrón creacional que nos permite crear las clases a partir de un simple método que nos regresa las instancias de las clases que necesitamos de el componente.

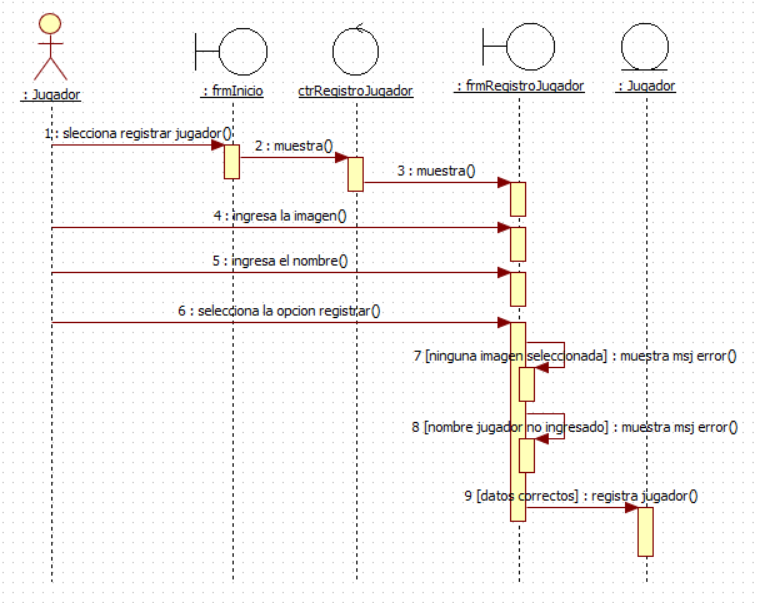
Patrón composite: El patrón composite nos sirve para componer estructuras complejas en otras estructuras mucho más simples, en el caso de el timbiriche como los cuadros están conformados por líneas.

Patrón mediador: El patrón mediator nos permite encapsular la comunicación entre objetos para evitar que estos se llamen directamente entre sí, en el caso de timbiriche se necesito editar el patrón para que la comunicación del timbiriche no le hablara directamente a las clases sino que todo lo relacionado con el juego pasará a través del mediador y este se encargará de llamar a las clases correspondientes.

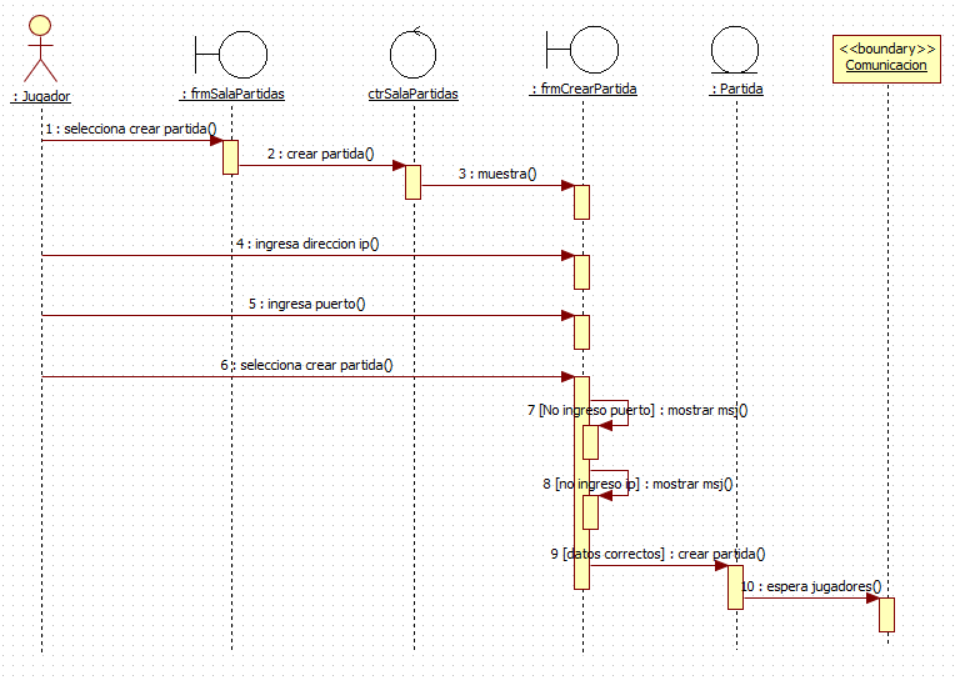
Patron fachada: el patrón fachada tiene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos.

Patrón singleton: El patrón de diseño singleton nos permite solucionar el problema de múltiples instancias de una misma clase tengan diferentes referencias en memoria logrando así que la primera clase que se instancie todas las que se instancian a través del singleton estén compartiendo la misma referencia de memoria y así poder hablarle desde diferentes clases teniendo la seguridad de que sea a la misma referencia de memoria que le estemos hablando.

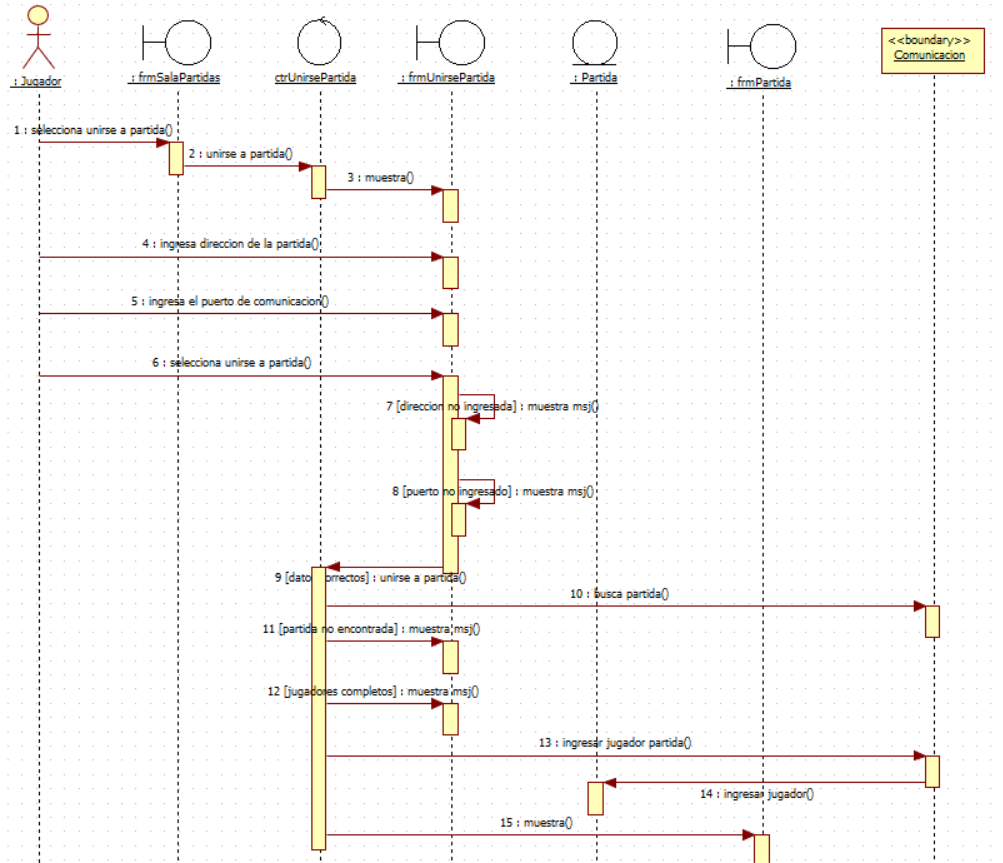
* 1. **Diagramas de secuencia a nivel de análisis**
     1. Registrar Jugador



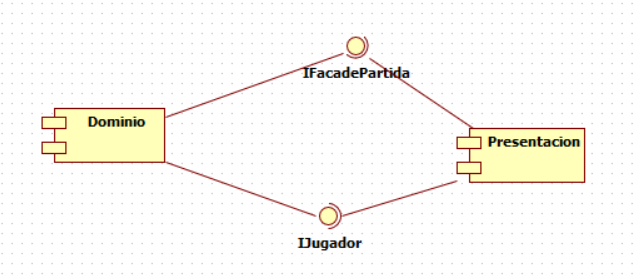
* + 1. Crear Partida



* + 1. Unirse Partida



1. **Vista de implementación**
   1. **Diagrama de componentes**

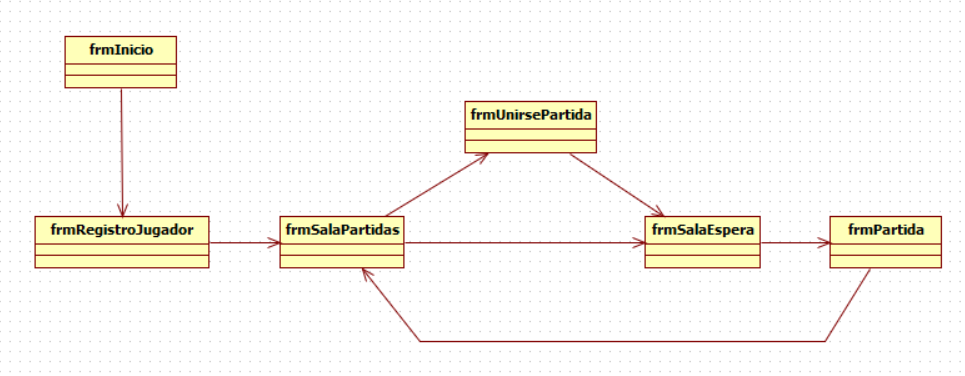
****

Estilo arquitectónico peer to peer, o red de pares iguales iguales, en esta arquitectura cada nodo del sistema o peer es igual de importante para la funcionalidad del sistema pero asu vez no es crítico, es decir el sistema puede funcionar sin un nodo fijo que funcione como servidor sino que todos los nodos comparten esta característica por igual, realizando la función de tanto cliente como servidor, de esta manera se cumple la funcionalidad peer to peer o red de iguales.

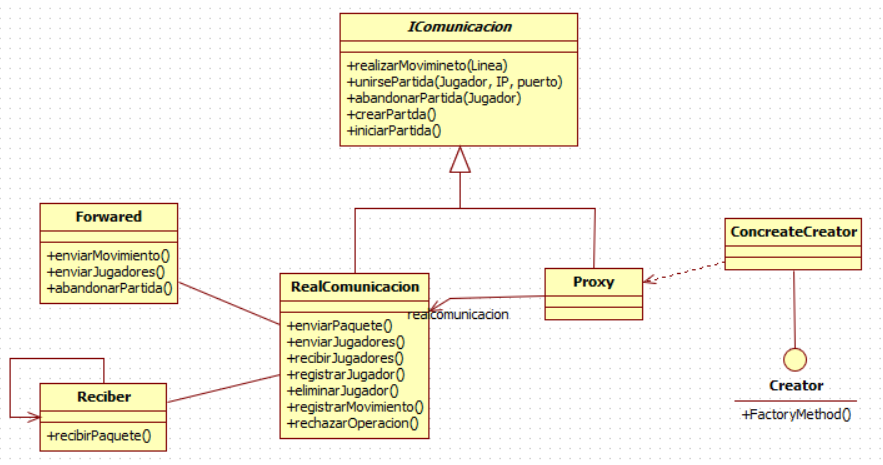
**Tabla 6.1. Responsabilidades de los componentes**

|  |  |
| --- | --- |
| **Componente** | **Responsabilidades** |
| Presentación | * Se encarga de recibir los eventos del usuario para realizar las diferentes jugadas dentro del sistema de timbiriche además de enviar los eventos realizadas por cada jugador de timbiriche. |
| Negocio | * Se encarga de proporcionar la parte lógica del sistema de timbiriche. |

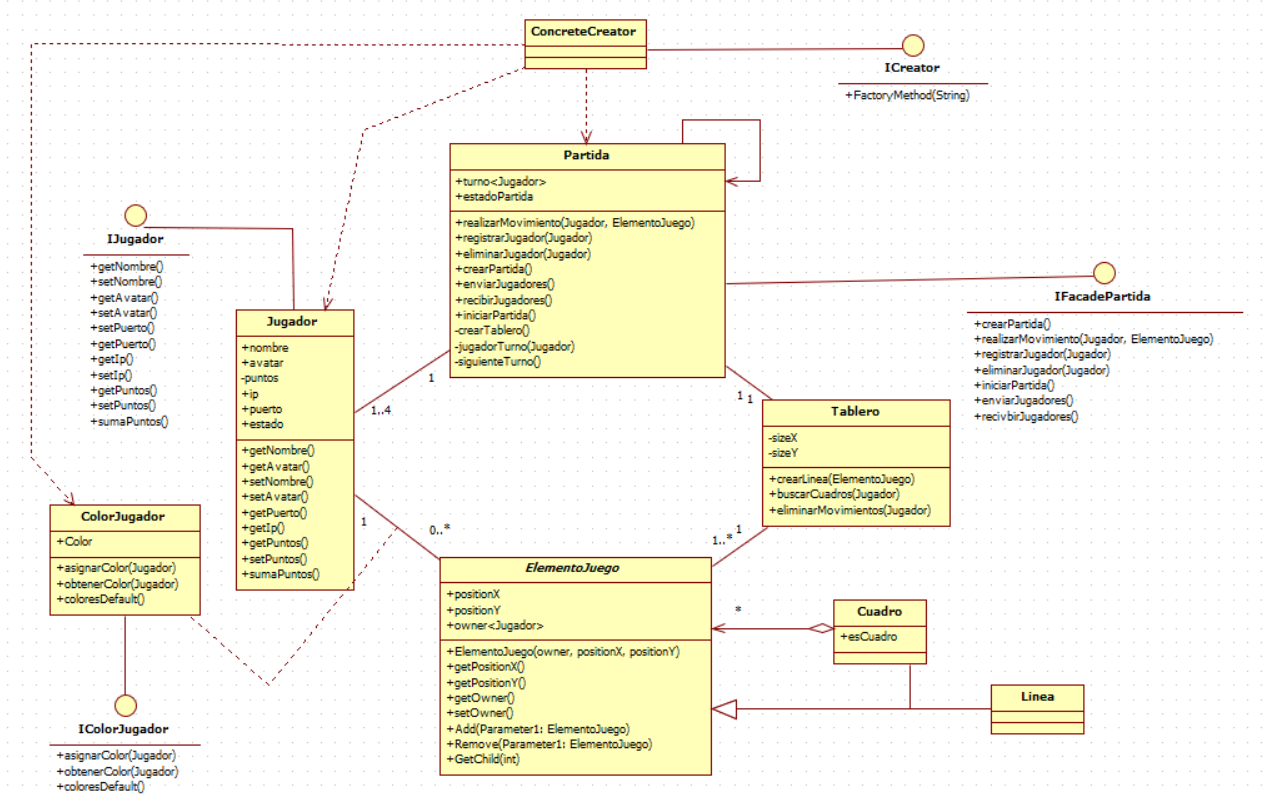
* 1. **Diagrama de clases de cada componente**
     1. **Presentación**

****

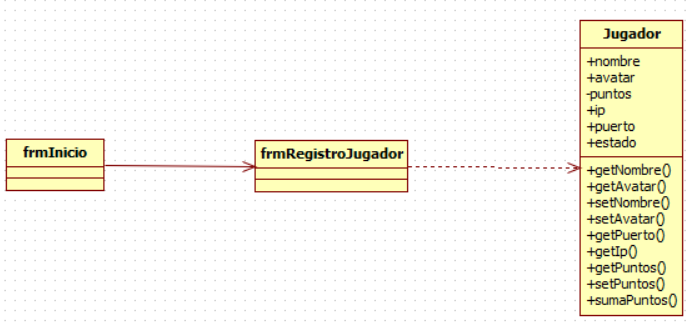
* + 1. **Comunicación**

****

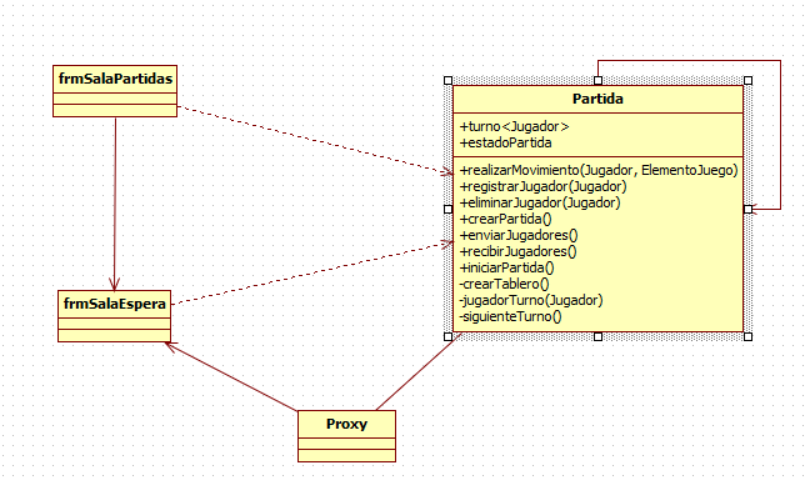
* + 1. **Negocio**

****

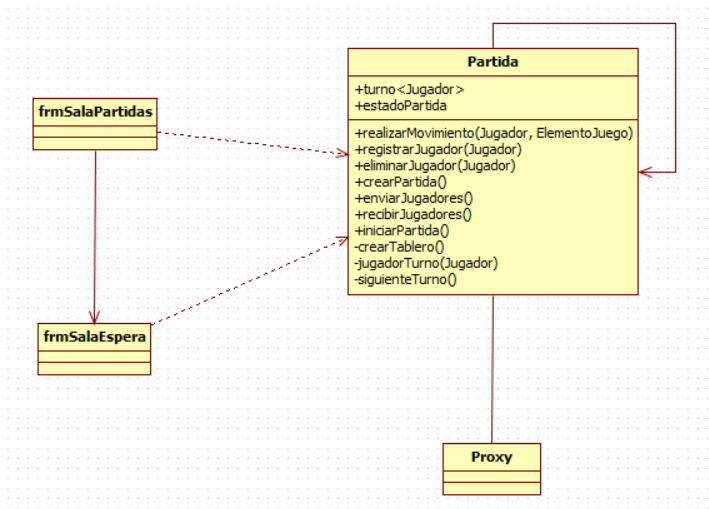
* 1. **Realización de casos de uso**
     1. **Registrar jugador**

****

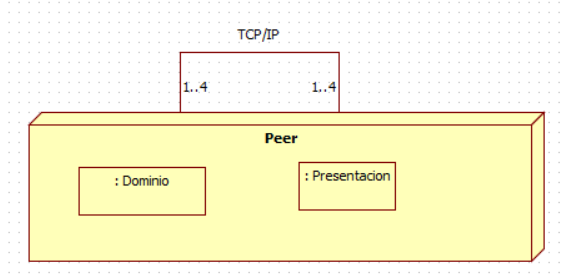
* + 1. **Unirse partida**

****

* + 1. **Crear partida**

****

1. **Vista de despliegue**
   1. **Diagrama de despliegue**

****

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos. ​Muestra la arquitectura del sistema como el despliegue de los artefactos de software a los objetivos de despliegue.

Al ser arquitectura peer to peer so se necesita instalar el sistema e indicar a un nodo externo o otro sistema para comenzar a compartir datos entre ellos.

**Tabla 7.1. Descripción de los elementos del diagrama de despliegue**

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| Peer | * El la arquitectura peer to peer cada peer tiene el mismo valor en el sistema asi como tambien las mismas responsabilidades, funcionando como iguales, es decir cada sistema(peer) funciona como igual no importa desde donde se contacte o cuando se conecte al sistema. |