Laura Arias Prada

Néstor Diazgranados

Andrea Fajardo

William Jiménez

Germán Morales

David Suárez

ALIMNOVA

Mayo 11/2010

SOFTWARE DESIGN DESCRIPTION



Historial De Cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Sección del documento modificada | Descripción de cambios (corta) | Responsable (S) |
| 0.1.0 | 24/04/2010 | Sección 1, 2.1, 2.2 | Documentación de dichas secciones | Laura Arias, Gerente de proyectos; David Suarez, Director de calidad y manejo de riesgos. |
| 0.1.1 | 25/04/2010 | Sección 3, 4,5 y 6 | Corrección de la numeración del documento | Andrea Fajardo, Arquitecta, Germán Morales, Director de desarrollo |
| 0.2.0 | 25/04/2010 | Sección 2.4.1, 2.4.2 | Documentación de dichas secciones | Laura Arias, Gerente de proyectos. |
| 0.3.0 | 25/04/2010 | Sección 3 | Documentación de dicha sección | Andrea Fajardo, Arquitecta |
| 0.4.0 | 26/04/2010 | Sección 2.3 y 2.5 | Documentación de dichas secciones | David Suarez, Director de Calidad y Manejo de Riesgos |
| 0.5.0 | 26/04/2010 | Sección 2.4.1, Referencias bibliográficas 1 - 5 | Documentación | Laura Arias, Gerente de proyectos. |
| 0.6.0 | 27/04/2010 | Sección 2.4.2 | Documentación de dicha sección | Laura Arias, Gerente de proyectos. |
| 0.7.0 | 27/04/2010 | Sección 3 y referencias | Documentación y corrección de las referencias | Andrea Fajardo, Arquitecta |
| 0.7.1 | 28/04/2010 | Sección 3 y referencias | Corrección referencias y redacción | Andrea Fajardo, Arquitecta |
| 0.8.0 | 01/05/2010 | Sección 3 | Cambio diagrama de componentes y documentación de las interfaces | Andrea Fajardo, Arquitecta |
| 0.8.1 | 2/05/2010 | Todo el documento | Revisión de calidad | David Suárez, Director de calidad |
| 0.9.0 | 2/05/2010 | Sección 3 | Documentación acerca del patrón Fachada | Andrea Fajardo, Arquitecta |
| 1.0.0 | 03/05/2010 | Sección 4 diagramas de secuencia servidor cobrar cover e ingresar partida | Documentación e Imagen | Laura Arias, Gerente de proyectos |
| 1.1.0 | 03/05/2010 | Sección 4 diseño Alto Nivel-Diagrama Despliegue | Documentación y Diagrama | Néstor Diazgranados, Diseñador Gráfico |
| 1.2.0 | 03/05/2010 | Sección 4.1.4 | Documentación | Néstor Diazgranados, Diseñador Gráfico |
| 1.3.0 | 04/05/2010 | Sección 4.4 Diagramas de Secuencia | Diagramas y documentación | David Suárez, Director de Calidad |
| 1.4.0 | 04/05/2010 | Sección 6. | Documentación de la interfaz de usuario y el árbol de navegabilidad | William Jiménez, Administrador de configuraciones y documentación. |
| 1.5.0 | 04/05/2010 | 4.4.1.6, 4.4.2.6 | Documentación | Laura Arias, Gerente de proyectos. |
| 1.6.0 | 04/05/2010 | Sección 5. | Elaboración completa de la sección a partir de los diagramas de clases cada una con su hipervínculo a su documentación. | Germán Morales, Director de desarrollo. |
| 1.6.1 | 09/05/2010 | Tabla de contenido, lista de tablas, lista de ilustraciones, sección 2.4 | Corrección de lo mencionado anteriormente | Laura Arias, Gerente de proyectos. |
| 1.7.0 | 10/05/2010 | Sección 3 y Revisión | Corrección Documento ampliación y mejora de la sección 3 | Andrea Fajardo, Arquitecta |
| 1.8.0 | 10/05/2010 | Sección 4 | Agregar imágenes de los diagramas | Laura Arias, Gerente de proyectos. |

Tabla 1: Historial de cambios

Tabla de Contenido

[1. Introducción 9](#_Toc261194703)

[1.1 Descripción Del Sistema 9](#_Toc261194704)

[1.2 Mapa del Diseño 10](#_Toc261194705)

[1.3 Referencias Bibliográficas 10](#_Toc261194706)

[1.4 Definiciones, Acrónimos y Abreviaciones 12](#_Toc261194707)

[2. Consideraciones de Diseño 12](#_Toc261194708)

[2.1 Suposiciones 12](#_Toc261194709)

[2.2 Restricciones 13](#_Toc261194710)

[2.3 Entorno del Sistema 13](#_Toc261194711)

[2.4 Metodología de Diseño 13](#_Toc261194712)

[2.4 Metodología de Diseño 14](#_Toc261194713)

[2.5 Riesgos 19](#_Toc261194714)

[3. Arquitectura 20](#_Toc261194715)

[3.1 Apreciación Global 20](#_Toc261194716)

[3.2 Diagrama de Componentes 22](#_Toc261194717)

[3.3 Estrategias de Diseño 28](#_Toc261194718)

[4. Diseño de Alto Nivel 30](#_Toc261194719)

[4.1 Diagrama de despliegue 30](#_Toc261194720)

[35](#_Toc261194721)

[4.2 Diagrama de Comportamiento e Interacción 39](#_Toc261194722)

[4.3 Diagrama de Actividad 39](#_Toc261194723)

[4.4 Diagrama de Secuencia 39](#_Toc261194724)

[5. Diseño de Bajo Nivel 50](#_Toc261194725)

[5.1 Subsistema Cliente 50](#_Toc261194726)

5.1.1 [Componente ComunicacionCliente 50](#_Toc261194727)

5.1.2 [Componente Coordinador 51](#_Toc261194728)

5.1.3 [Componente GUI 52](#_Toc261194729)

[5.2Subsistema Servidor 53](#_Toc261194730)

5.2.1 [Componente Logica 53](#_Toc261194731)

[6. Diseño de Interfaces de Usuario 54](#_Toc261194732)

[6.1 Diseño general de la aplicación 54](#_Toc261194733)

[6.2 Árbol de navegabilidad 57](#_Toc261194734)

[Anexos 57](#_Toc261194735)

Lista de Tablas

[Tabla 1: Historial de cambios 4](#_Toc260697229)

[Tabla 2: Definiciones y Acrónimos 11](#_Toc260697230)

[Tabla 3: Matriz de estructura de diseño. 15](#_Toc260697231)

[Tabla 4: Probabilidad y efecto de los riesgos en el SDD 18](#_Toc260697232)

[Tabla 5: planes de mitigación y contingencia de los riesgos en el SDD 18](#_Toc260697233)

[Tabla 8: Documentación de clases 46](#_Toc260697234)

[Tabla 9: Descripción de entradas y salidas 51](#_Toc260697235)

Lista de Ilustraciones

[Ilustración 1: Proceso para definición de diseño de alto nivel y bajo nivel. Adaptado de [4]. 13](#_Toc260697240)

[Ilustración 2: Proceso para definición de diseño de alto nivel y bajo nivel. Adaptado de [4]. 13](#_Toc260697241)

[Ilustración 3: Influencias sobre el arquitecto para definir la arquitectura. Tomado de [5]. 14](#_Toc260697242)

[Ilustración 4: Criterios de evaluación de la arquitectura. 16](#_Toc260697243)

[Ilustración 5: Riesgos del SDD 18](#_Toc260697244)

[Ilustración 6: Apreciación Global 21](#_Toc260697245)

[Tabla 7: Descripción Subsistemas 22](#_Toc260697246)

[Tabla 8: Descripción Componentes 22](#_Toc260697247)

[Ilustración 9: Diagrama de Componentes T-Monopoly 23](#_Toc260697248)

[Tabla 10: Interfaces en el diagrama de componentes T-Monopoly 23](#_Toc260697249)

[Tabla 11: Descripción Subsistema Cliente 24](#_Toc260697250)

[Tabla 12: Descripción Componente Comunicación Cliente 24](#_Toc260697251)

[Tabla 13: Descripción Componente Coordinador 25](#_Toc260697252)

[Tabla 14: Descripción Componente GUI 25](#_Toc260697253)

[Tabla 15: Descripción Subsistema Servidor 26](#_Toc260697254)

[Tabla 16: Descripción Componente Comunicación Servidor 26](#_Toc260697255)

[Tabla 17: Descripción Componente Lógica 26](#_Toc260697256)

[Tabla 18: Descripción Componente Comunicación Persistencia 27](#_Toc260697257)

[Tabla 19: Estrategia de diseño 29](#_Toc260697258)

[Tabla 20: Objetos del Diagrama de Componentes 30](#_Toc260697259)

[Ilustración 21: Vistas Diagrama Despliegue 31](#_Toc260697260)

[Ilustración 22: Conexiones 32](#_Toc260697261)

[Ilustración 23: Máquina Cliente 32](#_Toc260697262)

[Ilustración 24: Máquina Servidor 33](#_Toc260697263)

[Ilustración 25: Diagrama Despliegue 34](#_Toc260697264)

[Tabla 26: Plantilla Despliegue 35](#_Toc260697265)

[Tabla 27: Componente1 35](#_Toc260697266)

[Tabla 28: Componente2 35](#_Toc260697267)

[Tabla 29: Componente3 36](#_Toc260697268)

[Tabla 30: Componente4 36](#_Toc260697269)

[Tabla 31: Componente5 36](#_Toc260697270)

[Tabla 32: Componente6 37](#_Toc260697271)

[Tabla 33: Componente7 37](#_Toc260697272)

[Tabla 34: Componente8 37](#_Toc260697273)

[Tabla 35: Componente9 38](#_Toc260697274)

[Tabla 36: Componente9 38](#_Toc260697275)

[Tabla 20: Estrategia de diseño 39](#_Toc260697276)

[Ilustración 21: Diagrama de secuencia Ingresar Partida para servidor. 39](#_Toc260697277)

[Ilustración 22: Diagrama de secuencia Cobrar Cover para servidor. 40](#_Toc260697278)

[Ilustración 17: Diseño de bajo nivel 42](#_Toc260697279)

[Ilustración 18: Niveles de diseño 44](#_Toc260697280)

[Ilustración 19: Ejemplo diseño de bajo nivel 46](#_Toc260697281)

[Ilustración 20: Ejemplo Descripción de Clase 48](#_Toc260697282)

[Ilustración 21: Ejemplo Resumen de Métodos 49](#_Toc260697283)

[Ilustración 22: Ejemplo descripción de método 50](#_Toc260697284)

[Ilustración 23: Diseño general de la aplicación 51](#_Toc260697285)

[Ilustración 24: Ejemplo árbol de navegabilidad 52](file:///C:\Users\Kiquelin\Desktop\Temporal\SDD%5bAlimnova%5dLineaBaseV1.2.0.docx#_Toc260697286)

# 1. Introducción

### 1.1 Descripción Del Sistema

El objetivo del presente documento es brindar una descripción detallada del diseño y arquitectura de la aplicación T-Monopoly® el cual se encuentra en proceso de implementación por parte de los integrantes de Alimnova®. El propósito principal del SDD consiste en representar un sistema de software; este documento permite comunicar el diseño de software.

La sección 3 presenta a nivel detallado el proceso que se llevo a cabo para determinar la arquitectura que corresponde al diseño de alto nivel de T-Monopoly® [[**sección 3**](#_Arquitectura)]. Alimnova® selecciono 2 tipos de arquitectura. La primera y principal es la arquitectura Cliente-Servidor que permite la conexión de varios usuarios a un servidor que contiene la información de la aplicación. La segunda arquitectura corresponde a modelo-vista-controlador que está asociada a los subsistemas cliente y servidor, donde el controlador y la vista se encuentran en el subsistema cliente y el modelo es responsable de manejar la lógica del juego y la persistencia de datos, este componente se encuentra en el subsistema servidor; de esta manera el servidor es pesado y el cliente es liviano [[**sección 3.2**](#_Diagrama_de_Componentes)].

La forma en que Alimnova® manejará la comunicación es por medio de RMI (Remote Method Invocation), el cual permite invocar métodos de manera remota desde cualquier computador.

### 1.2 Mapa del Diseño

El diseño de software es importante para el mantenimiento de software, de manera que si se realiza algún cambio o si se agrega una nueva característica no se desestabilice el software. Existen 2 tipos de diseño:

* Diseño de alto nivel: Es el primer paso que se realiza en el diseño de software, donde se selecciona una arquitectura que satisfaga los requerimientos funcionales y no funcionales. Un buen diseño de alto nivel lleva a un mejor diseño de bajo nivel. El diseño de alto nivel se compone de:
  + Arquitectura que se adapte a las necesidades de la aplicación T-Monopoly® [[**sección 3**](#_Arquitectura)].
  + Diagrama de despliegue el cual muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos [[**sección 4.1**](#_Diagrama_de_despliegue)].
  + Diagramas dinámicos permite validar las interacciones entre los distintos objetos.
* Diseño de bajo nivel: también denominado modulo de diseño considerar donde se debe considerar el lenguaje de programación a usar en la implementación de manera que se determinen los tipos de interfaces, para el caso de Alimnova® el lenguaje de programación será Java [[**sección 5**](#_Diseño_de_Bajo)].

### 1.3 Referencias Bibliográficas

1. IEEE Std. 1016-1998, IEEE Recommended Practice for Software Design Descriptions, IEEE Computer Society.
2. Consulta relacionada con diseño de software, <http://www.the-software-experts.de/e_dta-sw-design.htm>, última consulta 24/04/2010.
3. Consulta relacionada con diagrama de despliegue, virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc, última consulta 24/04/2010.
4. S. Albin, The Art of Software Architecture: Design Methods and Techniques, John Wiley & Sons 2003.
5. L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison Wesley, 2003.
6. K. Barclay, J. Savage, Object-Oriented Design with UML and Java, El Sevier Butterworth Heinemann, 2004.
7. Consulta de diagrama de clases. Disponible: <http://www.agilemodeling.com/artifacts/classDiagram.htm#Classes>, última consulta: 26/04/2010.
8. A. Shalloway, J. Trott, Design Patterns Explained: A New Perspective on Object Oriented Design.
9. Consulta relacionada con patrones, <http://www.oodesign.com/proxy-pattern.html>, última consulta 27/04/2010.
10. Consulta relacionada con patrones, <http://home.earthlink.net/~huston2/dp/state.html>, última consulta 27/04/2010.
11. Patrón Observer. Disponible: [http://www.proactiva-calidad.com/java/patrones/observador.html. Consultado el 25/04/2010](http://www.proactiva-calidad.com/java/patrones/observador.html.%20Consultado%20el%2025/04/2010)
12. Patrones de diseño. Disponible: <http://patronesdediseno.blogspot.com/2009/05/patron-observer.html>. Visitado el 27/04/2010
13. Patrones de diseño. Disponible: [http://patronesdediseno.blogspot.com/2009/05/patron-state.html . Visitado el 25/04/2010](http://patronesdediseno.blogspot.com/2009/05/patron-state.html%20.%20Visitado%20el%2025/04/2010).
14. Patron state, patrón Bridge. Biel Massot Puigserver y Edu Herraiz Aparicio. Disponible: [http://dmi.uib.es/~yuhua/APOO07-08/Presentation/Bridge-State.pdf. Visitado el 25/04/2010](http://dmi.uib.es/~yuhua/APOO07-08/Presentation/Bridge-State.pdf.%20Visitado%20el%2025/04/2010).
15. Definición arquitectura cliente servidor. Disponible: [http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml Visitado el 28/04/2010](http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml%20Visitado%20el%2028/04/2010)
16. Garlan D. et al, An Introduction to Software Architecture, Enero 1994, Carnegie Mellon University
17. Wapedia.Proxy patron de diseño. Disponible: <http://wapedia.mobi/es/Proxy_(patr%C3%B3n_de_dise%C3%B1o)>
18. Patrón Proxy. Raúl Heras, Alberto Blasco , José Manuel Arévalo. Disponible: <http://www.google.com.co/url?sa=t&source=web&ct=res&cd=2&ved=0CAoQFjAB&url=http%3A%2F%2Fkybele.escet.urjc.es%2Fdocumentos%2FSI%2FPatrones%2F18_Proxy.ppt&rct=j&q=proxy+patron+&ei=X13cS4aCIJCg8ATJvfmvBw&usg=AFQjCNFoc9pclXEPdCrVjWTk00Rc2cZwrA>.
19. Diseño de software con patrones. Java Hispano Disponible: <http://www.javahispano.org/contenidos/es/diseno_de_software_con_patrones_parte_4/> Visitada el 02/05/2010.
20. Jim Arlow and Il a Nesutadt, UML and the Unified Process, Practical object-oriented analysis and design, Pearson Education, pag. 355-364.
21. John P. Flynt with Omar Salem, Software Engineering for Game Developers, Editorial Thomson, páginas 117-120.
22. Jeff Garland and Richard Anthony, Large-Scale Software Architecture, A Practical Guide using UML, John Wiley & Sons LTD, páginas 177-201.
23. Harcker Gutierrez, Camilo Gonzales, Sebastián Plata,Santiago Vélez, Diego Castillo PX\_Solutions.
24. Consulta del patrón Facade*,* [*http://www.dofactory.com/Patterns/PatternFacade.aspx*](http://www.dofactory.com/Patterns/PatternFacade.aspx)*,* Ultima consulta realizada el 9 de mayo de 2010.
25. Arquitectura Modelo Vista Controlador, Exequiel Catalini. Disponible en: http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/ . Visitado el 10 de mayo de 2010.

### 1.4 Definiciones, Acrónimos y Abreviaciones

|  |  |
| --- | --- |
| LETRA | DEFINICIONES |
| A | * Arquitectura de software: es una extensión del proceso de diseño de ingeniería. Este se centra en la descomposición de una sistema en componentes y las interacciones entre ellos para satisfacer requerimientos funcionales y nos funcionales. |
| B |  |
| C | * Componente: Definición de lo que existe en un modulo. |
| D |  |
| E |  |
| F |  |
| G |  |
| H |  |
| I |  |
| J |  |
| K |  |
| L |  |
| M | * Módulo: unidad de software que puede ser diseñada, implementada y compilada en un sistema ejecutable[4]. |
| N |  |
| O |  |
| P |  |
| Q |  |
| R | * RMI: Remote Method Invocation |
| S | * SDD: Software Design Description * SAD: Software Architecture Document |
| T |  |
| U |  |
| V |  |
| W |  |
| X |  |
| Y |  |
| Z |  |

Tabla 2: Definiciones y Acrónimos

# 2. Consideraciones de Diseño

### 2.1 Suposiciones

* El diagrama de clases tendrá aplicados patrones con el fin de garantizar un buen diseño de bajo nivel.
* El equipo donde se instalara la aplicación deberá tener la maquina virtual JRE versión 6 Update 16.
* El equipo de desarrollo cuenta con las herramientas necesarias para el diseño y la implementación de la aplicación T-Monopoly®. Las herramientas estas descritas en el [**SPMP Documento**] y el [**SRS Documento**] Línea Alimnova®.
* El equipo de desarrollo utilizo las librerías de Look and Feel para el manejo de interfaz grafica.

### 2.2 Restricciones

* La resolución de la pantalla donde se instale la aplicación será de 1000x800 pixeles.
* Para restricciones de Hardware [**SRS Documento**] Alimnova®.
* El equipo donde se instale la aplicación deberá contar con parlantes que sean compatibles con dicho equipo.
* La aplicación deberá tener una arquitectura cliente-servidor.
* La aplicación contará con interfaz gráfica fuerte.
* La aplicación deberá manejar persistencia de datos.

### 2.3 Entorno del Sistema

La aplicación T-Monopoly® no está ligada a ningún otro sistema o aplicación por lo que en esta sección se describirán únicamente el entorno a nivel de software y hardware que este posee.

A nivel de software T-Monopoly® requiere de varias especificaciones que son mencionadas en la sección 3.3.2.2 del documento [**SRS Documento**].

A nivel de hardware también debe cumplir con ciertas restricciones y especificaciones, mencionadas en la sección 3.3.2.1 del documento [**SRS Documento**].

### 2.4 Metodología de Diseño

El diseño de software se maneja a 2 niveles: diseño de alto nivel y diseño de bajo nivel [[**sección 1.2**](#_1.2_Mapa_del)]. Esta sección presenta el proceso llevado a cabo por Alimnova® para definir de manera adecuada y coherente ambos diseños.



Ilustración 1: Proceso para definición de diseño de alto nivel y bajo nivel. Adaptado de .

### 2.4 Metodología de Diseño

El diseño de software se maneja a 2 niveles: diseño de alto nivel y diseño de bajo nivel [[**sección 1.2**](#_1.2_Mapa_del)]. Esta sección presenta el proceso llevado a cabo por Alimnova® para definir de manera adecuada y coherente ambos diseños.



Ilustración 2: Proceso para definición de diseño de alto nivel y bajo nivel. Adaptado de .

##### 2.4.1 Arquitectura

2.4.1.1 Entendimiento del Problema

Muchos proyectos de software fallan porque crean soluciones para un problema de negocio inválido. La tarea consiste en identificar el problema real que se desea solucionar.

Comprender los requerimientos es una técnica bastante útil para esta etapa . Dado que estos ya fueron verificados con los casos de uso. Esta etapa fue realizada en el Hito 2 y especificada en el documento SRS ya que se realizo el proceso de ingeniería de requerimientos. Esta técnica permite a Alimnova® determinar la forma de la arquitectura. Este proceso permite mantener la consistencia de la aplicación T-Monopoly® y la trazabilidad.

2.4.1.2 Identificar los Elementos de Diseño y las Relaciones

El primer paso es establecer la descomposición de los componentes basado en los requerimientos funcionales. La arquitectura de la aplicación de software es un conjunto de subsistemas interconectados denominados módulos. Estos módulos son la construcción organizacional del código fuente en el momento de la implementación. La arquitectura definida por Alimnova® después de aplicar el proceso la podrá ver más adelante en este documento [[**sección 3**](#_Arquitectura)].

2.4.1.2.1 Definir el Contexto del Sistema

Permite describir la aplicación desde una perspectiva externa (Stakeholders). En el documento SPMP hubo una descripción de todas las interfaces externas que afectarían la aplicación T-Monopoly®. La arquitectura es el resultado de un conjunto de decisiones técnicas y de negocio. Las decisiones de negocio son influenciadas por los Stakeholders, para este proceso Andrea Fajardo, Arquitecto deberá tomar en cuenta las opiniones de ellos para construir la arquitectura. Sin embargo, dicha arquitectura no solo es influenciada por los Stakeholders, sino también por otros aspectos.

Ilustración 3: Influencias sobre el arquitecto para definir la arquitectura. Tomado de .

El ambiente técnico para la aplicación T-Monopoly® está definido en el documento SRS.

2.4.1.2.2 Identificar los Módulos

Los módulos son unidades discretas de software, generalmente se denominan componentes y conexiones [[**sección 1.3**](#_1.3_Referencias_Bibliográficas)]. Una arquitectura clásica modular es el sistema cliente-servidor el cual constituye una de las arquitecturas de la aplicación T-Monopoly® [[**sección 1.1**](#_1.1_Descripción_Del)].

Una técnica útil para establecer las conexiones entre los componentes consiste en hacer la matriz de estructura de diseño y luego representar esta estructura en un diagrama de componentes [[**sección 3.2**](#_Diagrama_de_Componentes)].

Tabla 3: Matriz de estructura de diseño.

Luego de establecer el diagrama de componentes basado en la matriz anterior, se debe hacer la respectiva documentación de los componentes y los conectores.

2.4.1.3 Evaluar la Arquitectura

La evaluación del diseño se hace en medidas cualitativas o datos cuantitativos, con el fin de determinar si se satisface la arquitectura con los requerimientos. El encargado de evaluar la calidad del diagrama de componentes que representa la arquitectura será David Suárez, Director de calidad y manejo de riesgos; acompañado de Andrea Fajardo, Arquitecta. La siguiente ilustración definirá los 2 criterios para evaluar la calidad de la arquitectura propuesta.

Ilustración 4: Criterios de evaluación de la arquitectura.

Es importante que el arquitecto analice las dependencias entre los requerimientos, ya que esto permite realizar una mejor descomposición a nivel de subsistemas y componentes. Otro tema relevante consiste en que el arquitecto debe tener juicio crítico para juzgar que requerimientos implican una mayor complejidad y cuáles no.

Alimnova® realizo trazabilidad para el SDD [**SRS Trazabilidad**] donde definió para cada requerimiento en que componente de la arquitectura se encuentra este requerimiento, a nivel de diagrama de clases[[**Sección 5**](#_Diseño_de_Bajo)] en que clase, atributo o método se encuentra tal requerimiento, esto para ambos subsistemas tanto cliente como para servidor. Esto con el fin de mantener la coherencia entre el diseño y los requerimientos.

2.4.1.4 Trasformar la Arquitectura

Luego de la evaluación de la arquitectura si esta no satisface los atributos de calidad se deben realizar ciertos cambios. El diseño se transforma aplicando operadores de diseño, estilos o patrones.

Hay 2 tipos de operadores de diseño. Modulares (mejora la vista modular del sistema) y de Diseño (mejora el tiempo de ejecución del componente del sistema):

Modulares:

* División del diseño en 2 o más módulos
* Substituir un modulo de diseño por otro
* Agregar un nuevo modulo
* Sacar un modulo del sistema
* Convertir un modulo en una interfaz
* Trasladar un modulo a otro sistema

Diseño:

* Descomposición del sistema en componentes
* Replicar módulos para mejorar la confiabilidad
* Comprimir 2 o más módulos en uno para mejorar el desempeño
* Abstracción de un componente para mejorar la adaptación a los cambios o la adaptabilidad.

A nivel de transformación Alimnova® dividió la arquitectura en 2 subsistemas (cliente y servidor), esto para facilitar la comprensión del diseño y diferenciar la lógica de estos subsistemas, donde se estableció que el servidor será pesado y cliente será liviano.

##### 2.4.2 Diseño de Bajo Nivel

Este diseño determinara la clase de interfaces que Alimnova® podría usar. Germán Morales, director de desarrollo, será el responsable de llevar a cabo el desarrollo de esta etapa donde se debe establecer el diagrama de clases para la implementación, con la colaboración del todos los integrantes de Alimnova®. En las siguientes sub secciones se realizara una explicación de la forma en que se realizo el proceso para establecer el diagrama de clases, con su correspondiente documentación [[**sección 5**](#_5._Diseño_de)].

Un diagrama de clases describe los tipos de objetos en el sistema y las relaciones que existen entre ellos. El diagrama de clases debe ser coherente con los diagramas de colaboración y secuencia. Para lograr la completitud de los diagramas de clases, Alimnova® incluyo en los diagramas de clases los mensajes de los diagramas de secuencia.

2.4.2.1 Identificación de Clases

Un objeto es una persona, lugar, cosa, concepto, evento, pantalla o reporte aplicable a la aplicación. Los objetos conocen cosas (tienen atributos) y hacen cosas (tienen métodos). Un objeto debe mantener un estado interno, donde se representan los datos privados encapsulados por el objeto.

2.4.2.2 Identificación de Relaciones

Los objetos se relacionan de diferentes maneras:

* Asociación
* Agregación
* Composición
* Herencia

2.4.2.3 Aplicación de Patrones

El proceso descrito en los numerales permitió establecer un diagrama de clases inicial, luego se agregaron patrones. Los patrones permiten enseñar principios de orientación a objetos, además permiten explicar por qué y que se hace con los objetos. El uso de los patrones se describe en secciones posteriores [[**sección 5**](#_Diseño_de_Bajo)].

2.4.2.3.1 Observer

Define una o más dependencias entre objetos, es decir, que cuando un objeto cambie su estado, todos los objetos dependientes deberán ser notificados y actualizados automáticamente.

2.4.2.3.2 Proxy

El patrón de diseño Proxy es aplicable cuando existe la necesidad de controlar el acceso a un objeto, así como cuando existe la necesidad de hacer referencia a un objeto sofisticado.

2.4.2.3.1 Facade

Provee una interfaz unificada a un conjunto de interfaces en un subsistema. El patrón fachada define a un nivel alto la interface que permite que un subsistema sea más fácil de usar[24].

### 2.5 Riesgos

Los riesgos para este hito, están fuertemente ligados a la implementación del diseño, por lo tanto se identificaron los siguientes riesgos:

Ilustración 5: Riesgos del SDD

|  |  |  |
| --- | --- | --- |
| Riesgo | Probabilidad | Efecto |
| Retraso en la entrega del documento de diseño | Bajo | Catastrófico |
| Requerimientos no satisfechos | Media | Crítico |
| Modificación del diseño | Media | Serio |
| El diseño no cumple alguna de las restricciones impuestas por el cliente | Baja | Critico |

Tabla 4: Probabilidad y efecto de los riesgos en el SDD

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Plan de mitigación** | **Plan de contingencia** |
| **Retraso en la entrega del documento de diseño** | Buena planeación de actividades, siendo consientes de la cantidad de trabajo delegado y el tiempo para realizarlo. | Reasignación de actividades [[**ver sección 5.3.2**](#_Plan_de_Control_6)] |
| **Requerimientos no satisfechos** | Se llevara a cabo una buena planeación para la trazabilidad de los requerimientos y así evitar dejar de lado alguno de ellos. [ver sección 4.2 SRS[Alimnova]LineaBaseV3.0.0] | Se reasignaran actividades a fin de que se disponga de algún tiempo para poder cumplir con el requerimiento que no está implementado. |
| **Modificación de diseño** | Llevar a cabo un buen proceso en el análisis de la trazabilidad a fin de que cuando se deba cambiar algún requerimiento, esto se haga antes de implementarlo. | Reasignación de actividades y adaptación del nuevo diseño. |
| **El diseño no cumple aluna de las restricciones impuestas por el cliente.** | Seguir una buena trazabilidad de las restricciones que se convertirán en requerimientos y de ahí a ser implementados. [ver SRS sección 4.2] | Reasignación de actividades a fin de cumplir con todas las restricciones impuestas. |

Tabla 5: planes de mitigación y contingencia de los riesgos en el SDD

# Arquitectura

### 3.1 Apreciación Global

Para el desarrollo del juego T-Monopoly, Alimnova® decidió usar como arquitectura principal la arquitectura Cliente/Servidor, inicialmente porque fue una de las restricciones dadas por el cliente y porque permite el manejo de varios jugadores, con el l fin que estos puedan acceder a la aplicación y ejecutarla hasta el final. Esta arquitectura esta arquitectura se complementa con una arquitectura Modelo-Vista- Controlador, con el fin de poder manejar la información existente en el servidor, en donde estará ubicado el modelo, siendo este el encargado de recibir la información, procesarla y luego enviar la información necesaria para realizar las siguientes ejecuciones, del mismo modo que se controla la información que se envía al controlador ubicado en el subsistema cliente, para que luego se muestre al jugador en la vista , lo cual corresponde finalmente a lo que ve el jugador, durante la ejecución y desarrollo del juego.

Una vez se realizaron investigaciones acerca de esta arquitectura Alimnova® identificó que la arquitectura Cliente- Servidor permite que el servidor sea quién tenga la lógica del programa y del juego como tal, para así poder proteger los datos y la información importante para el desarrollo del juego, siendo esta seguridad una parte muy importante para la ejecución del juego, así como también esta arquitectura permite que el servidor sea independiente de los clientes, que no los conozca, pero que los clientes si puedan acceder al servidor para pedir determinados servicios que le ofrece, cumpliendo con las restricciones dispuestas por el [**SPMP Documento**].

Por otro lado esta arquitectura permite que el usuario vea el sistema de forma homogénea, ya que los cambios o actualizaciones son independientes a él, permitiendo así mantener la seguridad que todos los clientes están viendo lo mismo, reduciendo así el riesgo de presentar información errónea a todos o a algún cliente en determinado momento de la ejecución del juego, así como también mantener al cliente alejado de las actividades de la ejecución del juego a un nivel inferior de diseño.

Otra de las ventajas que brinda usar esta arquitectura es eliminar la necesidad de mover grandes bloques de información por la red, entre diferentes equipos, lo cual permite que tan solo se envíe la información necesaria y con ello reducir el tiempo de respuesta. Los servidores son quienes manejan los datos, procesan peticiones y luego transfieren solamente los datos pedidos por determinado cliente.

Dentro del juego T-Monopoly Alimnova® se basó en algunas restricciones dadas por el juego o por el cliente siendo estos datos determinantes para el diseño de la arquitectura, dentro de las cales estuvieron aspectos como:

* Para jugar T-Monopoly, se necesitan como mínimo 2 personas y máximo 5 jugadores, cada jugador tendrá su turno para poder realizar diferentes actividades dentro del desarrollo normal del juego. Para llevar a cabo esta característica es importante contar con un mecanismo que controle la asignación y ejecución de los turnos para cada cliente y que valide si puede realizar determinada transacción, acorde con las reglas establecidas por el [**SPMP Documento**].
* Debido a que durante la ejecución del juego habrán varios jugadores es importante tener un mecanismo que controle la interfaz gráfica de cada uno y que esta corresponda con la realidad de la ejecución del juego.

A partir de la información obtenida y antes expresada Alimnova® decidió tomar esta arquitectura, puesto que permite cumplir con lo propuesto , en donde existe un servidor que es un mecanismo encargado de controlar todos los clientes que ingresan a una partida, validará la información de cada jugador tanto respecto al perfil de cada uno, como las actividades llevadas a cabo durante la ejecución de T-Monopoly , siendo este el encargado de modificar o actualizar la información según sea requerido y su objetivo es gestionar toda la información necesaria para poder llevar a cabo el objetivo del juego T-Monopoly. Por otro lado el cliente corresponde a la representación del jugador humano dentro del juego; el cual estará comunicándose continuamente con el servidor mientras se esté ejecutando T-Monopoly.

Respecto a la arquitectura Modelo-Vista-Controlador ofrece a la aplicación ventajas como :

* Separación entre los componentes de un programa, permitiendo así realizar la implementación por separado.
* Conexión dinámica entre el modelo y sus vistas, producida en tiempo de ejecución y no en tiempo de compilación.
* Permite tener diferentes vistas para el mismo modelo, lo cual es muy útil para la aplicación, ya que con el mismo modelo podemos ofrecer diferentes modelos.

Ilustración 6: Apreciación Global

### 3.2 Diagrama de Componentes

El propósito del diagrama de componentes es poder reflejar a partir de los requerimientos identificados por Alimnova®, la razón de ser de cada uno de los componentes y la relación que existe entre cada componente con la trazabilidad de cada requerimiento identificado, con el fin de implementarlos, evaluados, modificados de forma independiente.

Con el fin de mantener la claridad de este documento y el entendimiento del mismo Alimnova® dividió la aplicación en dos subsistemas, cada uno con sus respectivos componentes. Los cuales han sido documentados teniendo en cuenta la siguiente tabla:

|  |  |
| --- | --- |
| Nombre Subsistema | * *Componente N* |
| Propósito del Subsistema | *Descripción del Propósito* |
| Composición del Subsistema | * *Componente 1* * *Componente 2* * *Componente N* |

Tabla 6: Descripción Subsistemas

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente |  |  |  |
| Propósito del  Componente | *Descripción del propósito* | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *Interfaz 1* | *Componente 2* | *Descripción de la utilidad o función de la interfaz mencionada. ¿Qué servicios presta a los componentes que la utilizan?* |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *Interfaz 2* | *Componente 3* | *Descripción de la utilidad o función de la interfaz mencionada. ¿Qué servicios presta a este componente?* |

Tabla 7: Descripción Componentes

El siguiente diagrama, representa los subsistemas, las relaciones entre ellos, sus componentes y las relaciones entre estos dentro del subsistema, las cuales fueron identificadas por Alimnova® y se definió para T-Monopoly.



Ilustración 7: Diagrama de Componentes T-Monopoly

|  |  |  |
| --- | --- | --- |
| Nombre de la Interface | Componentes que la utilizan | Servicios prestados |
| ComunicaciónRMI | Cliente y Servidor | Encargada de solicitar determinado servicio al servidor por parte del Cliente.  Esta interface conoce los servicios que presta el servidor y por ende están dispuestos para el Cliente |

Tabla 8: Interfaces en el diagrama de componentes T-Monopoly

##### 3.2.1Subsistema Cliente

|  |  |
| --- | --- |
| Nombre Subsistema | * *Cliente* |
| Propósito del Subsistema | * Es la representación del jugador humano, encargado de solo interactuar con la interfaz gráfica , donde se le muestra el tablero y demás objetos importantes para llevar a cabo el desarrollo del juego. * Es liviano ya que tiene tan solo los componentes de interfaz gráfica, es decir, sólo la parte visual del juego como tal y solicita los servicios al servidor por medio de la interface (ComunicacionRMI). |
| Composición del Subsistema | * Comunicación Cliente * Coordinador * GUI |

Tabla 9: Descripción Subsistema Cliente

3.2.1.1 Componente Comunicación Cliente

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *Comunicación Cliente* |  |  |
| Propósito del  Componente | Encargado de solicitar los servicios ofrecidos por la interface en el subsistema del servidor (comunicación servidor) por medio de la comunicaciónRMI | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *N/A* |  |  |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *ComunicaciónRMI* | *T-Monopoly* | *Esta interfaz permite realizar la solicitud al servidor como tal, de forma que si esta comunicación no está bien hecha la información no llega a este subsistema y por ello ya existen fallas en la ejecución de este.* |

Tabla 10: Descripción Componente Comunicación Cliente

3.2.1.2 Componente Coordinador

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *Coordinador* |  |  |
| Propósito del  Componente | Pide ejecución del servicio ofrecido por el servidor, por medio de comunicación cliente y a partir de esta información actualiza la interfaz del usuario(GUI). | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *Solicita* | *Comunicación cliente*  *Coordinador* | *Permite al Coordinador solicitar determinados servicios al servidor* |
| *Actualiza* | *Coordinador*  *GUI* | *Permite que el controlador actualice la interfaz del jugador, acorde a la información que disponga en determinado momento de ejecución del juego* |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *ComunicaciónRMI* | *T-Monopoly* | *Esta interfaz permite realizar la solicitud al servidor como tal, de forma que si esta comunicación no está bien hecha la información no llega a este subsistema y por ello ya existen fallas en la ejecución de este.* |

Tabla 11: Descripción Componente Coordinador

3.2.1.3 Componente GUI

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *GUI* |  |  |
| Propósito del  Componente | Encargada de mostrar gráficamente el juego (tablero, fichas, propiedades….) al cliente ,muestra lo que está sucediendo en el servidor, a medida que se ejecuta o termina la ejecución de un turno | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *N/A* |  |  |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *Actualiza* | *Coordinador* | *Permite a la interfaz actualizarse acorde a la información que tiene el coordinador* |

Tabla 12: Descripción Componente GUI

##### 3.2.2 Subsistema Servidor

|  |  |
| --- | --- |
| Nombre Subsistema | * *Servidor* |
| Propósito del Subsistema | Representa la lógica del juego, el cual es el manejador del sistema, encargado de ejecutar cada una de las actividades necesarias para llevar a cabo el juego. Así mismo es el encargado de realizar la conexión a determinada partida creada y habilita la opción de almacenar la información del cliente (perfil) o de cada partida. |
| Composición del Subsistema | * Comunicación Servidor * Lógica * Persistencia |

Tabla 13: Descripción Subsistema Servidor

3.2.2.1 Componente Comunicación Servidor

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *Comunicación Servidor* |  |  |
| Propósito del  Componente | Encargado de mostrarle al cliente, por medio del RMI, los servicios que él puede brindarle al cliente | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *N/A* |  |  |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *ComunicaciónRMI* | *T-Monopoly* | *Esta interfaz permite realizar la solicitud al servidor como tal, de forma que si esta comunicación no está bien hecha la información no llega a este subsistema y por ello ya existen fallas en la ejecución de este.* |

Tabla 14: Descripción Componente Comunicación Servidor

3.2.2.2 Componente Lógica

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *Lógica* |  |  |
| Propósito del  Componente | Representa el juego, donde se llevan a cabo todas las acciones del juego y demás actividades correspondientes a este. | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *Ejecuta* | *Comunicación Servidor* | *Permite a la lógica iniciar operaciones acorde a la información que le envía por medio de esa interfaz* |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *N/A* |  |  |

Tabla 15: Descripción Componente Lógica

##### 3.2.3 Componente Persistencia

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre Componente | *Persistencia* |  |  |
| Propósito del  Componente | Encargado de almacenar en archivos planos la información correspondiente a:  Ranking  Perfiles  Historial del juego | | |
| Interfaces Disponibles | ***Nombre de la Interfaz*** | ***Componentes que la utilizan*** | ***Servicios prestados*** |
| *Guarda* | *Lógica* | *Permite almacenar la información de la partida y demás operaciones que requieran ser almacenadas por lo lógica* |
| Dependencias | ***Nombre de la Interfaz*** | ***Componente que la ofrece*** | ***Descripción de la dependencia*** |
| *Ejecuta* | *Lógica* | *El almacenamiento se realiza a partir de la ejecución de la lógica, por lo cual es importante que se ejecute, para realizar el almacenamiento* |

Tabla 16: Descripción Componente Comunicación Persistencia

### 3.3 Estrategias de Diseño

Con el fin de facilitar el desarrollo del a aplicación es importante definir e implementar determinadas estrategias de diseño, ya que al usarlas facilita la identificación del esquema general de las arquitectura a la vez que la forma en la que se va a implementar.

Para esto Alimnova® ha definido estrategias a partir de patrones de diseño que facilitarán la implementación de la arquitectura y el entendimiento de la misma, acorde a la arquitectura ya escogida (Cliente- Servidor) y teniendo en cuenta los requerimientos funcionales conforme a la prioridad asignada a cada uno de ellos.

|  |  |  |
| --- | --- | --- |
| **Estrategia de Diseño** | | **Descripción** |
| **Patrones de Diseño** | OBSERVER | * Define una dependencia uno a muchos entre objetos de tal forma que cuando el estado de un objeto cambia, se les notifica el cambio a todos los que dependen de él y se actualizan de forma automática. * Será implementado en la parte del tablero, en donde este está pendiente cuando se presente un movimiento de una ficha para actualizar el tablero. * Es útil también cuando un cambio en un objeto exige cambios en otros, pero se desconoce a cuántos objetos afectan dichos cambios. En este caso, cuando es necesario informar un cambio de estado de una ficha dentro del tablero de juego, no se conoce con certeza la cantidad de jugadores o clientes que se encontrarán en ese momento usando la aplicación y quienes necesitan dicha información de estado. |
| PROXY | * Para crear objetos que consumen muchos recursos, pero instanciarlos cuando el cliente lo solicite o se cumplan las condiciones para que esto suceda. * Este patrón ayuda a facilitar la comunicación entre el cliente y el servidor |
| FACHADA | * Simplifica la interface entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace las veces de pantalla o fachada. * Este patrón es empleado en el subsistema cliente, con el fin de aislar cada una de las clases presentes en la Interfaz Gráfica de usuario (GUI), de la comunicación del cliente con el servidor, permitiendo así aislar este sistema complejo de las demás clases. |
| **Estilo arquitectónico** | CLIENTE – SERVIDOR | [**3.Apreciación Global**](#_3.Apreciación_Global) |
| **Priorización de Requerimientos** |  | Implementación de la aplicación de acuerdo a la priorización de los requerimientos funcionales. Es decir, primero se implementarán los requerimientos con mayor importancia y así hasta los de menos prioridad. |

Tabla 17: Estrategia de diseño

# 4. Diseño de Alto Nivel

### 4.1 Diagrama de despliegue

Dentro del diseño y el paradigma orientado objetos compete el diagrama de despliegue. Hay dos tipos de diagramas que se desarrollan para tal objetivo :

1. Descripción: Los nodos describen un tipo de hardware (como un PC), y los componentes representan un tipo de software.

2. Instanciación: Los nodos son instancias que describen un tipo de hardware, se relacionan con otras instancias de nodos y con instancias de componentes que describen un tipo de software.

Alimnova® decidió trabajar con el diagrama de descripción, ya que para los propósitos de un diagrama de instanciación, se busca una aproximación hacía sistemas distribuidos, que no se contempla en nuestros objetivos, ni corresponde al alcance del proyecto T-Monopoly. Adicionalmente no se requiere como regla de de negocio [**SPMP Documento**].

Un diagrama de componentes debe proveer la información de localización física, donde se encuentran los diferentes paquetes y/o componentes del sistema. De esta manera identificando no sólo software sino también hardware .

Es importante recordar el tipo de arquitectura seleccionada por Alimnova® para el desarrollo de T-Monopoly [[ver sección 3](#_Arquitectura)] y de esta manera poder contextualizarla al despliegue del sistema. De la misma manera para poder comprender este diagrama, es necesario tener en cuenta el diagrama de componentes [[ver sección 3.2](#_3.2_Diagrama_de)] pues la idea de esta representación es extrapolar dichos componentes en archivos ejecutables, librerías dinámicas y archivos de almacenamiento o persistencia así como la comunicación física de los nodos contenedores de los paquetes.

Con el fin de representar el sistema desde el punto de vista físico es importante describir los ítems u objetos que componen un diagrama de despliegue:

Tabla 18: Objetos del Diagrama de Componentes

Los diagramas de despliegue contemplan varios puntos de vista, buscan relacionar el hardware y software de diversas maneras .

Para ello se trabajan 3 vistas:

Ilustración 8: Vistas Diagrama Despliegue

Para lo que compete a nuestro proyecto resulta útil utilizar únicamente la vista de despliegue, ya que no contamos con modelos de instanciación, donde sea necesario localizar y estructurar varias máquinas. En referencia a la vista de procesos, no necesitamos hilos de ejecución debido a que su administración corre por cuenta del servicio RMI, y la ejecución de procesos es simple, se basa en una lógica para cada uno de los paquetes definidos (cliente, servidor).

Para la construcción de él diagrama de Despliegue de Descripción se definen dos tiempos:

1. El flujo de los nodos y sus conexiones.
2. Asignación pertinente y coherente de los componentes a los nodos, que concuerdan con la arquitectura del sistema.

##### 4.1.1 Flujo de los nodos y sus conexiones.

Ilustración 9: Conexiones

Para un mejor conocimiento de las restricciones del Sistema T-Monopoly [**SRS Documento**].

##### 4.1.2 Asignación componentes por nodo

Alimnova® escogió dos nodos cada uno de los cuales ejecutará un paquete de funcionalidad, según la división del diagrama de componentes en cliente y servidor [[ver sección 3.2](#_Diagrama_de_Componentes)].

**Para el cliente:**

Ilustración 10: Máquina Cliente

**Para el servidor:**

Ilustración 11: Máquina Servidor

##### 4.1.3 Diagrama de DespliegueC:\Users\Willie\Documents\PUJ\7mo\Ingenieria de software\Proyecto\Despliegue\Despliegue\Deployment Model.bmp

Ilustración 12: Diagrama Despliegue

##### 4.1.4 Documentación Diagrama Despliegue

Para la especificación del diagrama de despliegue, el cual está basado en el de componentes se tendrá la siguiente tabla de especificación de las asociaciones entre componentes.[23]

|  |  |
| --- | --- |
| ID | Identificador único de la asociación. |
|  | |
| Componente origen | De los dos componentes involucrados en una asociación, es quien asocia a otro, presta algún servicio al componente asociado. |
|  | |
| Componente asociado | De los dos componentes involucrados en una asociación, el quien es asociado, es decir quien depende de o solicita un servicio. |
|  | |
| Descripción | Define el objetivo de la asociación |

Tabla 19: Plantilla Despliegue

|  |  |
| --- | --- |
| ID | C1 |
|  | |
| Componente origen | Cliente |
|  | |
| Componente asociado | J.V.M 1.6 |
|  | |
| Descripción | Define como el cliente utiliza las funcionalidades de la máquina virtual de java , para poder realizar las solicitudes necesarias al cliente |

Tabla 20: Componente1

|  |  |
| --- | --- |
| ID | C2 |
|  | |
| Componente origen | Cliente |
|  | |
| Componente asociado | Coordinador |
|  | |
| Descripción | Define como el cliente usa efectivamente las reglas de negocio, y como accede a las funcionalidades que presta el servidor por medio de RMI. |

Tabla 21: Componente2

|  |  |
| --- | --- |
| ID | C3 |
|  | |
| Componente origen | Coordinador |
|  | |
| Componente asociado | Comunicación |
|  | |
| Descripción | Define la interacción de cómo el cliente envía las solicitudes en tiempo de ejecución, con el objetivo de comunicarse con el servidor, esto para tener la información suficiente para realizar sus tareas. |

Tabla 22: Componente3

|  |  |
| --- | --- |
| ID | C4 |
|  | |
| Componente origen | Cliente |
|  | |
| Componente asociado | J.V.M 1.6 |
|  | |
| Descripción | Define como el cliente utiliza las funcionalidades de la máquina virtual de java , para poder realizar las solicitudes necesarias al cliente |

Tabla 23: Componente4

|  |  |
| --- | --- |
| ID | C5 |
|  | |
| Componente origen | Cliente |
|  | |
| Componente asociado | J.V.M 1.6 |
|  | |
| Descripción | Define como el cliente utiliza las funcionalidades de la máquina virtual de java , para poder realizar las solicitudes necesarias al cliente |

Tabla 24: Componente5

|  |  |
| --- | --- |
| ID | S1 |
|  | |
| Componente origen | Servidor |
|  | |
| Componente asociado | J.V.M 1.6 |
|  | |
| Descripción | Define como el servidor utiliza las funcionalidades de la máquina virtual de java , para poder realizar las respuestas necesarias a l as solicitudes de los clientes |

Tabla 25: Componente6

|  |  |
| --- | --- |
| ID | S2 |
|  | |
| Componente origen | Servidor |
|  | |
| Componente asociado | Lógica |
|  | |
| Descripción | Define como el servidor ejecuta las reglas de negocio, realiza verificaciones y validaciones. |

Tabla 26: Componente7

|  |  |
| --- | --- |
| ID | S3 |
|  | |
| Componente origen | Lógica |
|  | |
| Componente asociado | Comunicación |
|  | |
| Descripción | Define como el servidor puede interactuar con el cliente, brindándole respuesta para cumplir reglas de negocio, verificaciones y validaciones. |

Tabla 27: Componente8

|  |  |
| --- | --- |
| ID | S4 |
|  | |
| Componente origen | Lógica |
|  | |
| Componente asociado | Persistencia |
|  | |
| Descripción | Define como el servidor toma los datos esenciales de la partida y los jugadores, para mantener los perfiles y las estadísticas de la partida. |

Tabla 28: Componente9

|  |  |
| --- | --- |
| ID | S4 |
|  | |
| Componente origen | Comunicación (Cliente) |
|  | |
| Componente asociado | Comunicación (Servidor) |
|  | |
| Descripción | Define los servicios mediante los cuales el cliente hace solicitudes al servidor |

Tabla 29: Componente9

### 4.2 Diagrama de Actividad

##### 4.2.1 Diagrama de Actividad Negociar

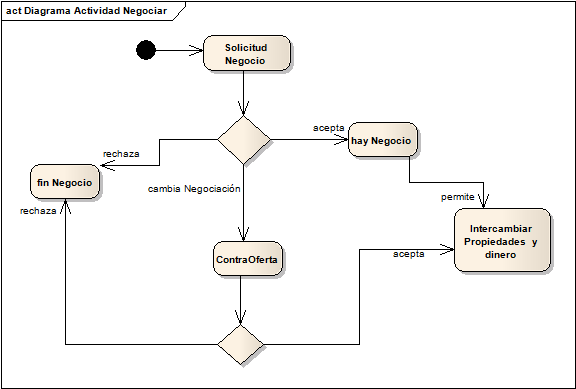


Ilustración 13: Diagrama de actividad Negociar.

##### 4.2.2 Diagrama de Actividad Subastar

Ilustración 14: Diagrama de actividad subastar

### 4.3 Diagrama de Secuencia

La selección de los casos de uso por Alimnova® se realizo de acuerdo a la priorización de estos; los cuales presentaban una prioridad en escala (1-10) igual o superior a 7. A continuación se describen los casos de uso a los cuales se le aplicaran diagramas de secuencia. Cada caso de uso tiene diagramas de secuencia para cliente y para servidor.



Tabla 30: Estrategia de diseño

##### 4.3.1 Diagramas de Secuencia Para Servidor

4.3.1.1 Crear Partida

Ilustración 15: Diagrama de secuencia crear partida para servidor

La documentación la encontrara en el siguiente link: [ServidorCrearPartida].

4.4.1.2 Negociar

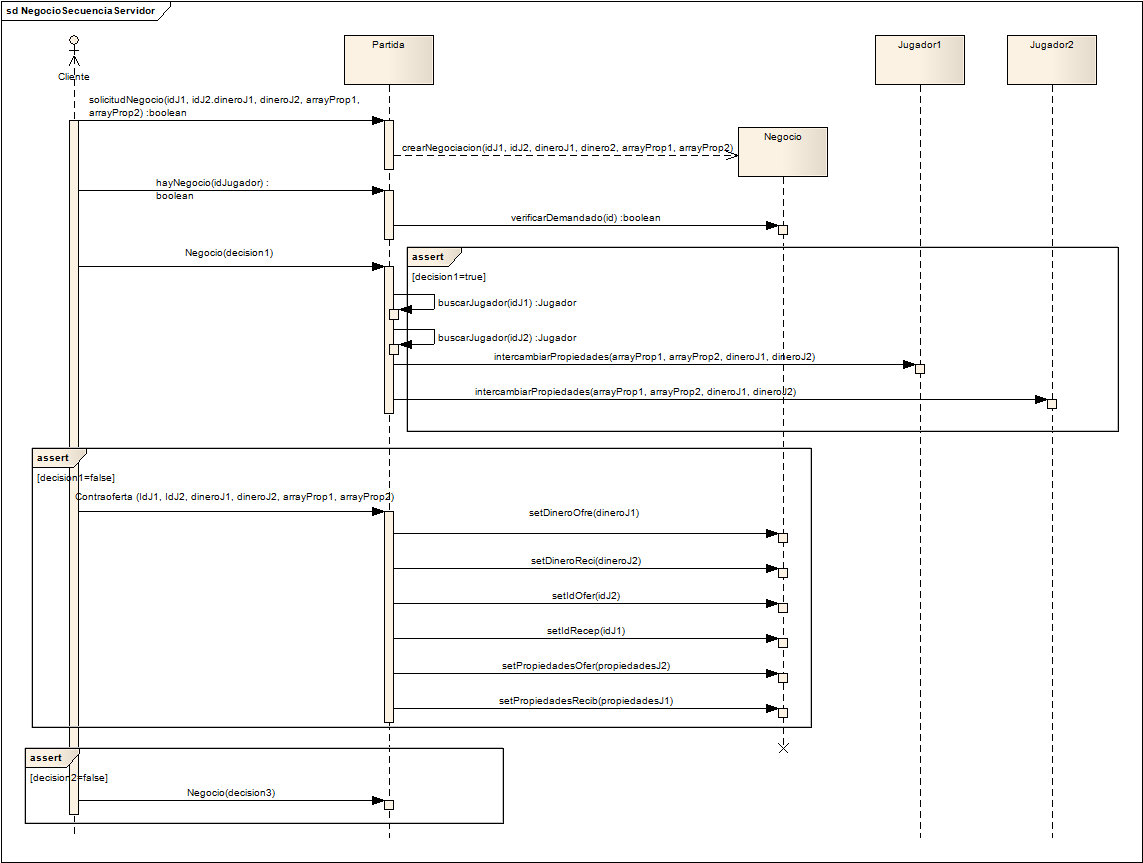


Ilustración 16: Diagrama de secuencia Negociar para servidor.

La documentación la encontrará en el siguiente link: [SecuenciaNegociarServidor].

4.3.1.3 Ingresar a una Partida

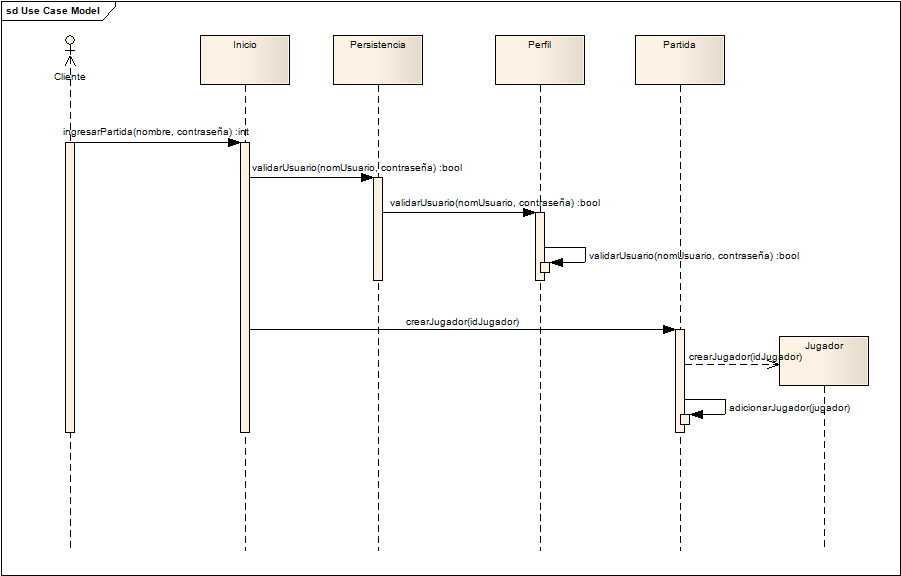


Ilustración 17: Diagrama de secuencia Ingresar Partida para servidor.

La documentación la encontrara en el siguiente link: [[SecuenciaIngresarPartidaServidor](file:///G:\Mis%20Documentos\Universidad\Semestre_8\Ingesoft\Pro\alimnova\SDD\Diagramas\DiagramasSecuencia-Acividad\IngresarPartida\index.htm)].

4.3.1.4 Subastar

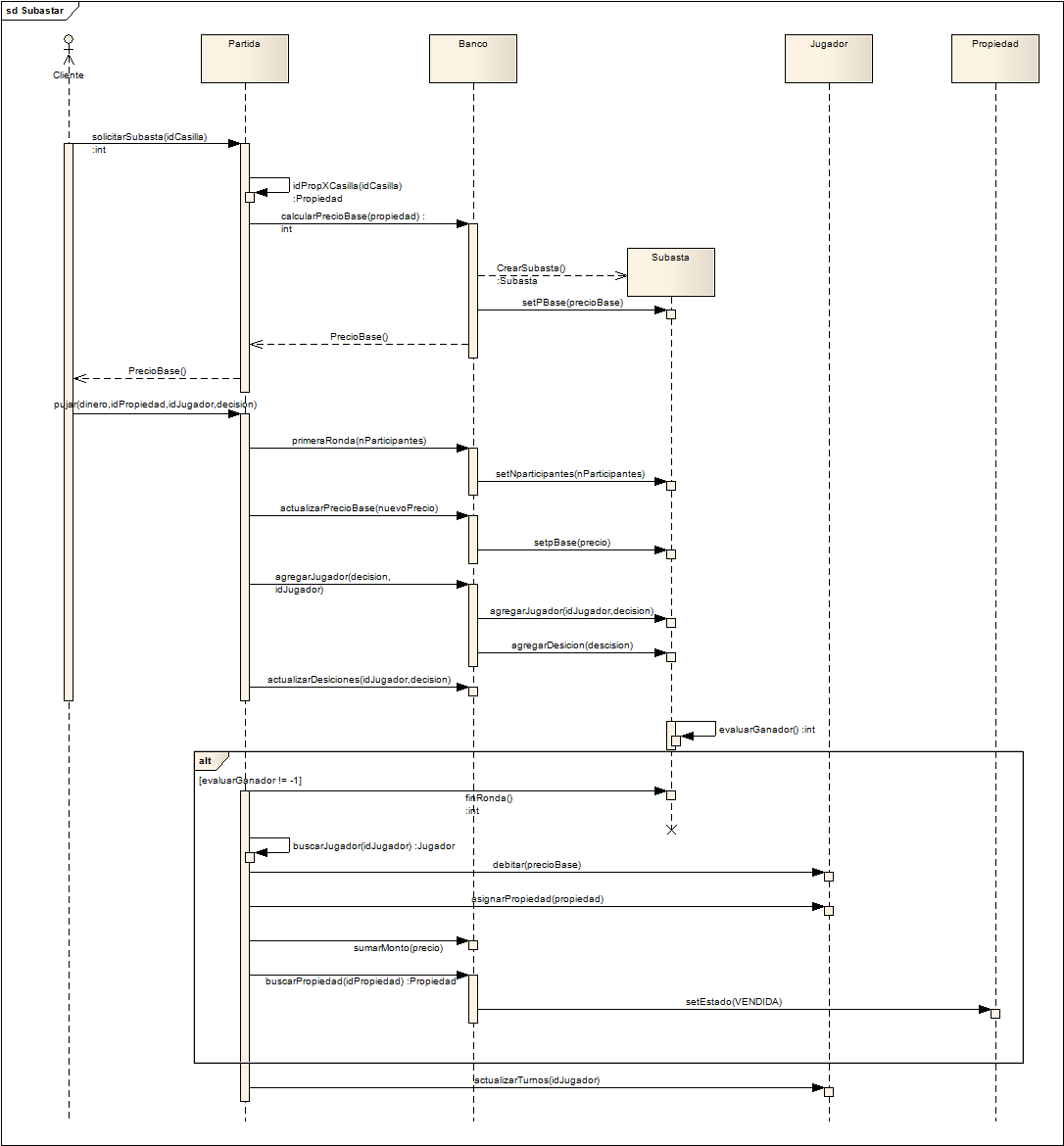


Ilustración 18: Diagrama de secuencia Subastar para servidor.

La documentación la encontrara en el siguiente link: [SecuenciaSubastarServidor]

4.3.1.5 Cobrar Cover

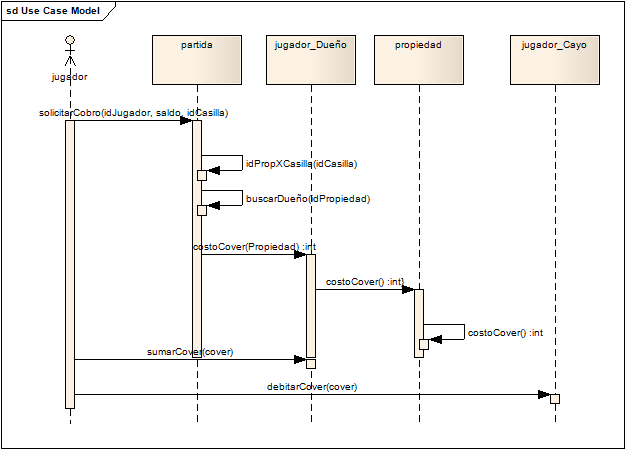


Ilustración 19: Diagrama de secuencia Cobrar Cover para servidor.

La documentación la encontrara en el siguiente link: [[SecuenciaCobrarCoverServidor](file:///G:\Mis%20Documentos\Universidad\Semestre_8\Ingesoft\Pro\alimnova\SDD\Diagramas\DiagramasSecuencia-Acividad\CobrarCover\index.htm)].

4.3.1.6 Inicializar Partida

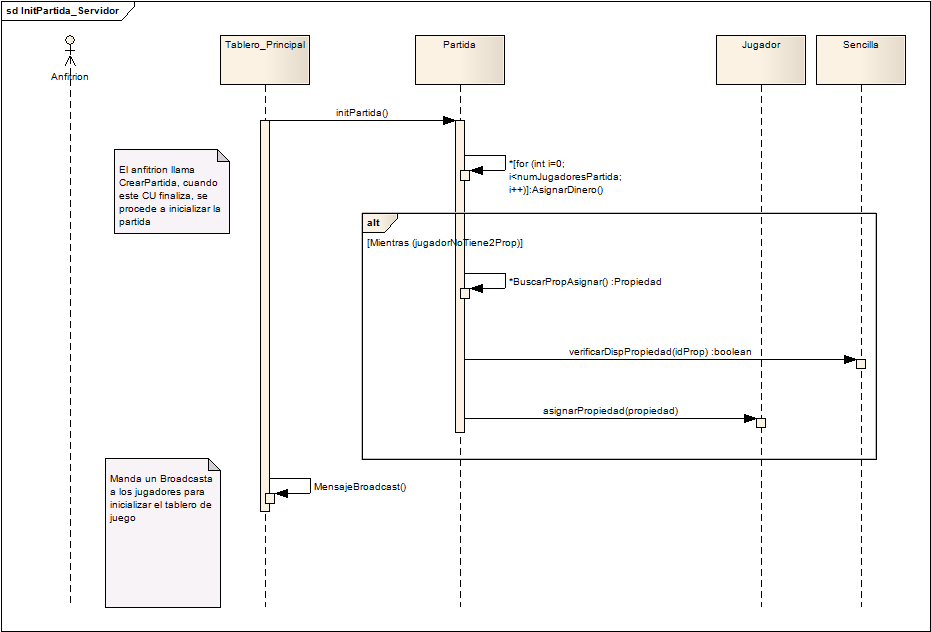
****

Ilustración 20: Diagrama de secuencia Inicializar Partida para servidor.

La documentación la encontrara en el siguiente link: [SecuenciaInicializarPartidaServidor].

4.4.1.7 Comprar Propiedad

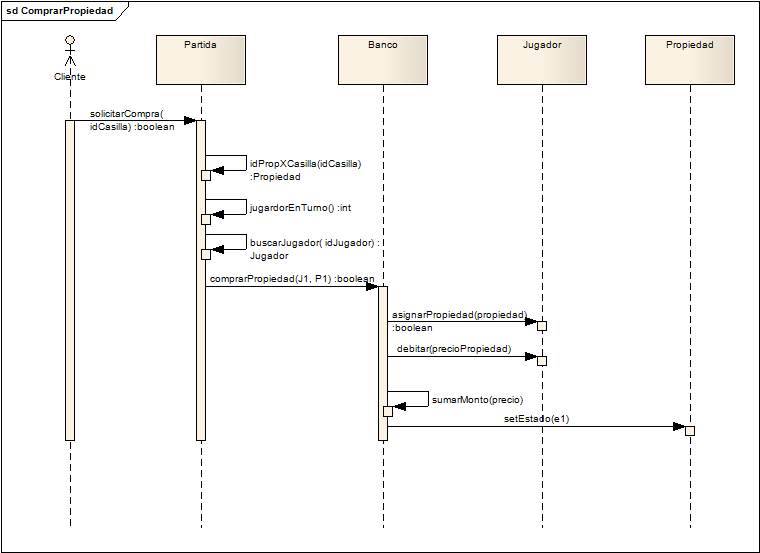
****

Ilustración 21: Diagrama de secuencia Comprar Propiedad para servidor.

La documentación la encontrara en el siguiente link: [SecuenciaComprarPropiedadServidor].

##### 4.3.2 Diagramas de Secuencia Para Cliente

4.3.2.1 Crear Partida

Ilustración 22: Diagrama de secuencia Crear Partida cliente

La documentación la encontrara en el siguiente link: [SecuenciaCrearPartidaCliente].

4.3.2.2 Negociar

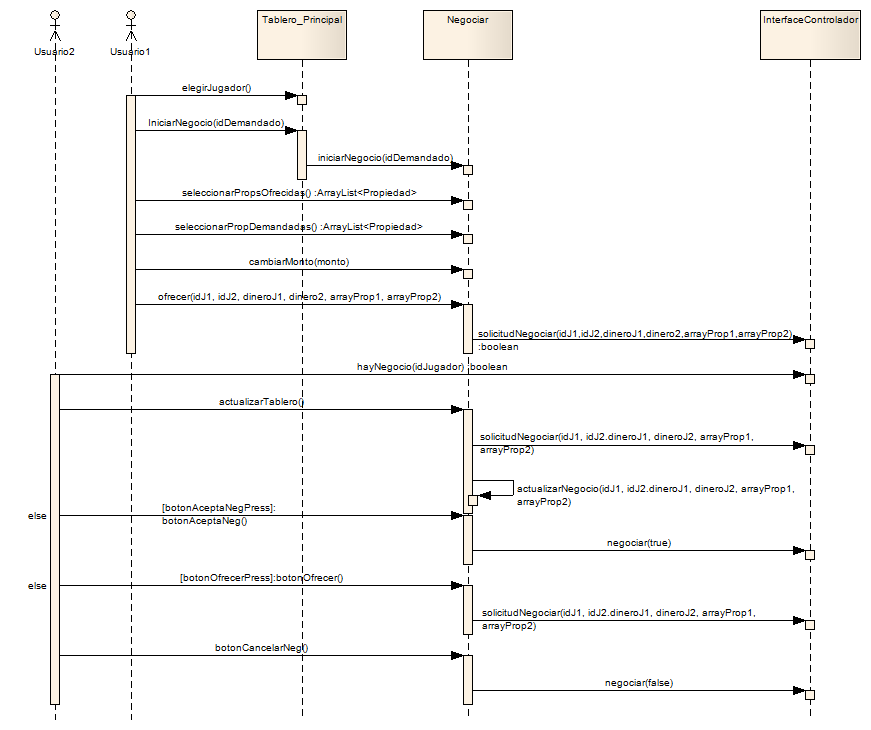


Ilustración 23: Diagrama de secuencia Negociar para Cliente.

La documentación la encontrara en el siguiente link: [SecuenciaNegociarCliente].

4.3.2.3 Ingresar a una Partida

4.3.2.4 Subastar

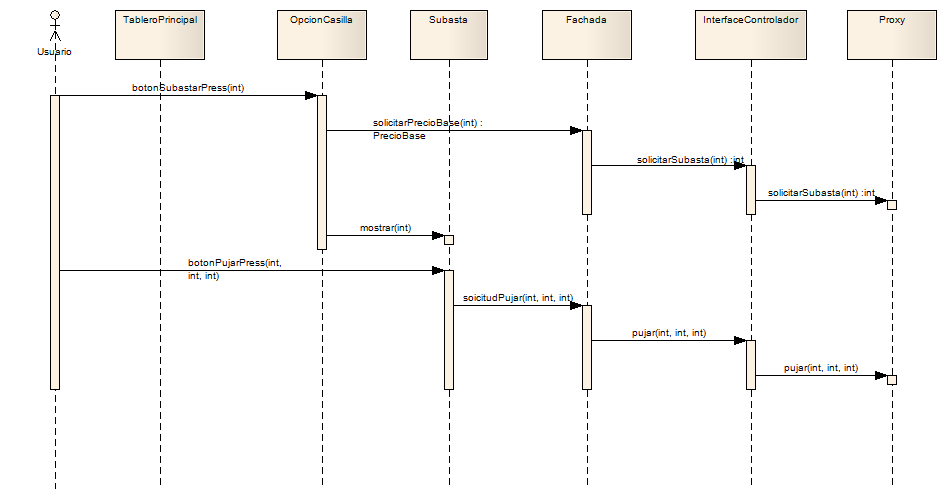


Ilustración 24: Diagrama de secuencia Subastar para Cliente.

La documentación la encontrara en el siguiente link: [SecuenciaSubastarCliente].

4.3.2.5 Cobrar Cover

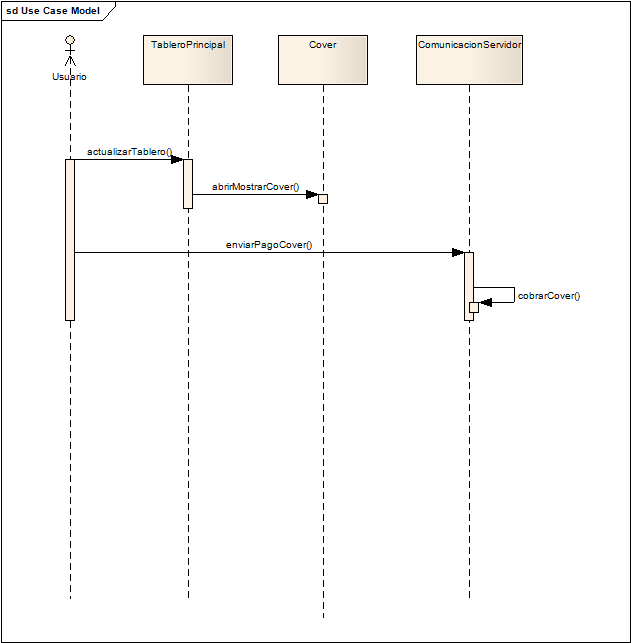


Ilustración 25: Diagrama de secuencia Cobrar Cover para Cliente.

La documentación la encontrara en el siguiente link: [SecuenciaCobrarCoverCliente].

4.3.2.6 Inicializar Partida

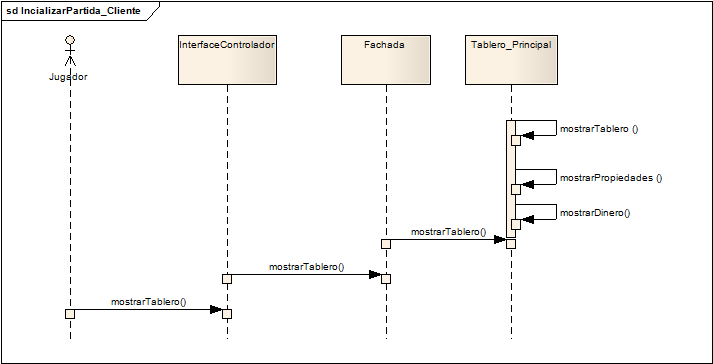


Ilustración 26: Diagrama de secuencia Inicializar Partida para Cliente.

La documentación la encontrara en el siguiente link: [SecuenciaInicializarPartidaCliente].

4.3.2.7 Comprar Propiedad

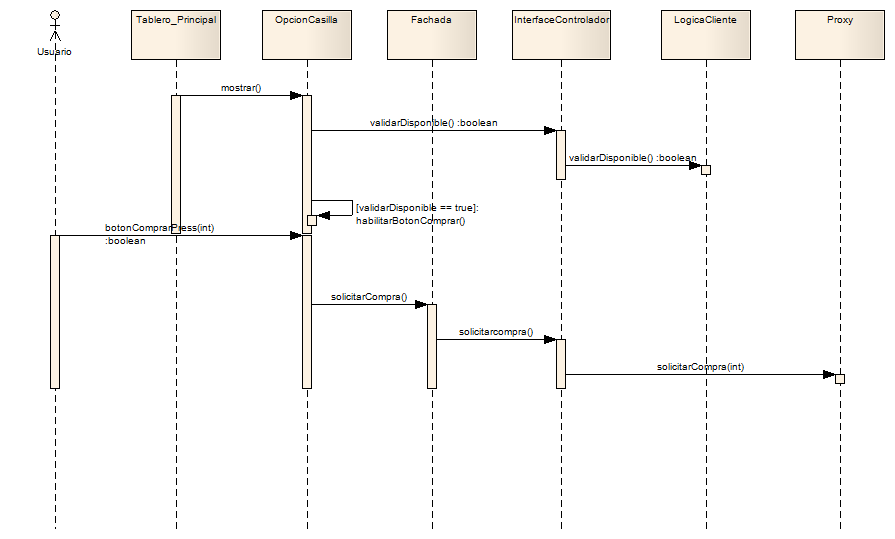


Ilustración 27: Diagrama de secuencia Comprar Propiedad para Cliente.

La documentación la encontrara en el siguiente link: [SecuenciaComprarPropiedadCliente].

5. Diseño de Bajo Nivel

Esta parte del diseño, Alimnova® presentara una solución a nivel lógico con la intención de poder satisfacer todos los requerimientos planteados como meta según la prioridad dada a cada uno de estos. Alimnova, basándose en la información obtenida en diseño de alto nivel realizo el de bajo nivel haciendo los diagramas de clases de cada uno de los componentes de los subsistemas que componen la arquitectura del T-Monopoly.

### 5.1 Subsistema Cliente

A continuación presentaremos los diagramas de clases de cada componente del subsistema Cliente cada uno con su correspondiente documentación.

##### 5.1.1. Componente ComunicacionCliente

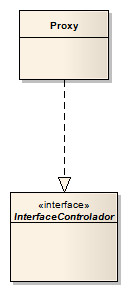


Ilustración 28: Diagrama de clases del componente ComunicacionCLiente

[Documentación del diagrama de clases del componente ComunicacionCliente].

##### 5.1.2 Componente Coordinador

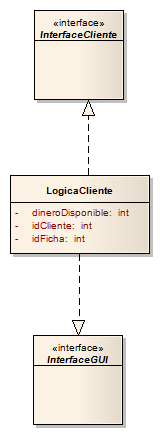


Ilustración 29: Diagrama de clases del componente Coordinador

[Documentación del diagrama de clases del componente Coordinador].

##### 5.1.3 Componente GUI

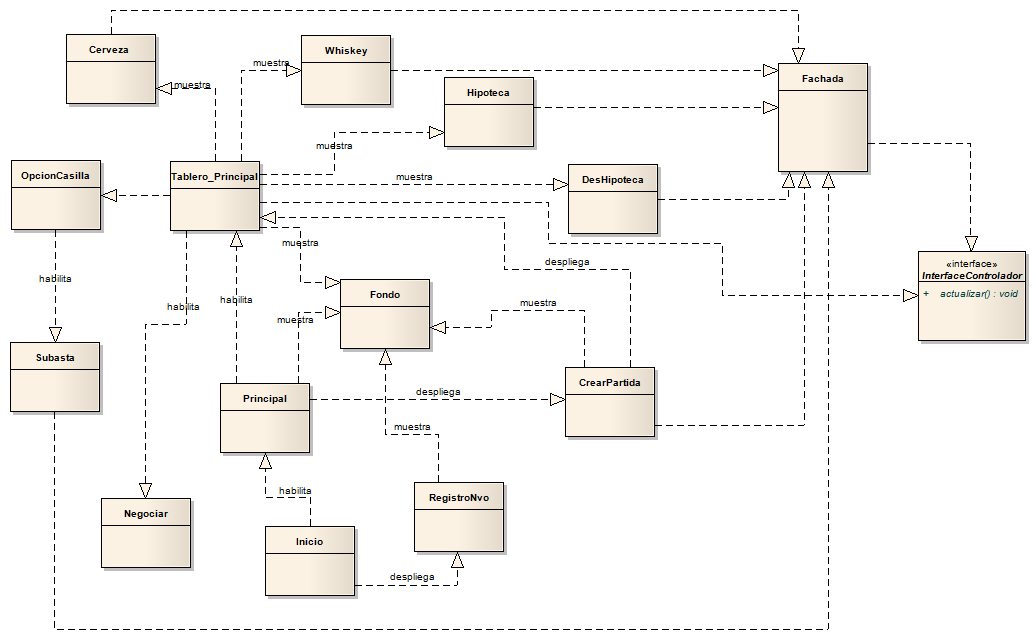


Ilustración 30: Diagrama de clases del componente GUI

[Documentación del diagrama de clases del componente GUI].

### 5.2 Subsistema Servidor

A continuación presentaremos los diagramas de clases de cada componente del subsistema Cliente cada uno con su correspondiente documentación.

##### 5.2.1 Componente Lógica

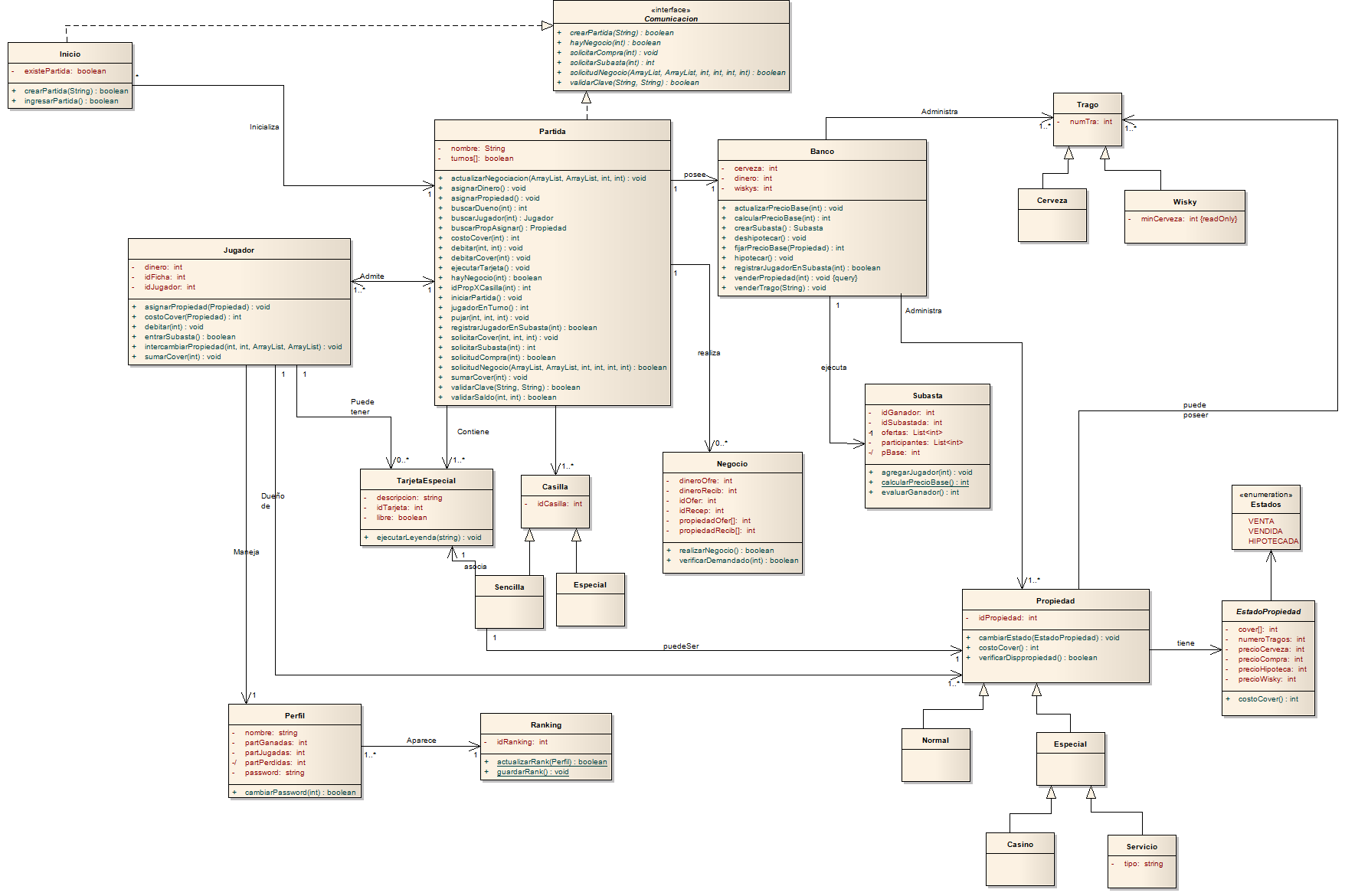


Ilustración 31: Diagrama de clases del componente Lógica

[Documentación del diagrama de clases del componente GUI].

# 6. Diseño de Interfaces de Usuario

### 6.1 Diseño general de la aplicación

La interfaz gráfica de T-Monopoly® esta implementada en Java mediante el uso de una librería externa denominada Look and Feel que le da un aspecto más sofisticado y apropiado para la aplicación.

El desarrollo del juego, específicamente de una partida, se da por medio de la navegación de los frames de la aplicación por parte del usuario, las cuales a su vez representan una funcionalidad dentro de la aplicación que son correspondientes con las opciones presentadas en los frames mencionados.

A su vez estos frames manejan la entrada de datos por teclado y el procesamiento de solicitudes mediante la selección de opciones por medio del mouse. Por esta razón y para tener una idea clara del alcance y las restricciones de los datos externos a procesar por la aplicación se presenta a continuación la especificación de entradas y salidas de la aplicación.

El formato a utilizar es el sugerido por la plantilla Ironworks con las modificaciones pertinentes y adaptadas a las necesidades de Alimnova®.

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Nombre de la entrada/salida dado en las documentaciones* |
| Alias (Interfaz) | *Nombre o Label con el que identifican la entrada/salida* |
| Alias (Código) | *Nombre a nivel del código fuente del programa de la entrada/salida* |
| Propósito | *Propósito de funcionalidad de la entrada/salida* |
| Tipo de Entrada | *Por ejemplo: Campo de Texto, Tabla, Botón, etc.* |
| Tipo de Dato | *Por ejemplo: Numérico, Alfanumérico o Alfabético* |
| Longitud mínima | *Longitud mínima asignada a la entrada/salida* |
| Longitud máxima | *Longitud máxima asignada a la entrada/salida* |
| Valor por defecto | *Valor por defecto que se encuentra en la entrada/salida* |
| Rango | *Rango de los valores que puede tener la entrada/salida* |
| Validación | *Como se verifica que la entrada/salida es valida* |
| Comentarios | *Comentarios como por ejemplo: obligatoriedad, medio por el cual entra o sale* |

Tabla 31: Formato para especificación de entradas y salidas.

***Inicio***

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Usuario.* |
| Alias (Interfaz) | *Usuario.* |
| Alias (Código) | *usuario.* |
| Propósito | *Almacenar el nombre de usuario de un jugador.* |
| Tipo de Entrada | *Campo de texto.* |
| Tipo de Dato | *Alfanumérico.* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres.* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

Tabla 32:Formato de inicio

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Contraseña.* |
| Alias (Interfaz) | *Contraseña* |
| Alias (Código) | *contrasenia.* |
| Propósito | *Almacenar la contraseña de un jugador.* |
| Tipo de Entrada | *Campo de texto.* |
| Tipo de Dato | *Alfanumérico.* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres.* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

***Registro***

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Usuario.* |
| Alias (Interfaz) | *Usuario.* |
| Alias (Código) | *usuario.* |
| Propósito | *Almacenar el nombre de usuario de un jugador.* |
| Tipo de Entrada | *Campo de texto.* |
| Tipo de Dato | *Alfanumérico.* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres.* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

Tabla 33:

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *contraseña.* |
| Alias (Interfaz) | *Contraseña* |
| Alias (Código) | *contrasenia.* |
| Propósito | *Almacenar la contraseña de un jugador.* |
| Tipo de Entrada | *Campo de texto.* |
| Tipo de Dato | *Alfanumérico.* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres.* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

Tabla 34:

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Validar Contraseña.* |
| Alias (Interfaz) | *Contraseña (repita)* |
| Alias (Código) | *contrasenia2.* |
| Propósito | *Almacenar la contraseña de un jugador por segunda vez para una posterior verificación de consistencia.* |
| Tipo de Entrada | *Campo de texto.* |
| Tipo de Dato | *Alfanumérico.* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres.* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

Tabla 35:

***Crear Partida***

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Nombre de Partida* |
| Alias (Interfaz) | *Nombre Partida* |
| Alias (Código) | *nombrePartida* |
| Propósito | *Almacenar el nombre de una partida para su identificación única.* |
| Tipo de Entrada | *Campo de texto* |
| Tipo de Dato | *Cadena de caracteres* |
| Longitud mínima | *4 caracteres.* |
| Longitud máxima | *15 caracteres* |
| Valor por defecto | *Cadena vacía (“”).* |
| Rango | *Cualquier combinación de 4 a 15 caracteres que no contenga caracteres especiales como =, /, \, ¿, ?, etc.* |
| Validación | *Debe ser diferente a la cadena vacía (””).* |
| Comentarios | *Se ingresa por teclado.* |

Tabla 36:

***Negociar***

|  |  |
| --- | --- |
| Convención | *Identificador de la entrada/salida dentro de la interfaz* |
| Nombre | *Monto a ofrecer.* |
| Alias (Interfaz) | *Monto de dinero.* |
| Alias (Código) | *monto.* |
| Propósito | *Guardar el monto a ofrecer por un jugador en una negociación.* |
| Tipo de Entrada | *Elemento de datos con varios valores posibles.* |
| Tipo de Dato | *Numérico.* |
| Longitud mínima | *Cero (0).* |
| Longitud máxima | *Valor máximo de un entero en Java.* |
| Valor por defecto | *0.* |
| Rango | *Entre 0 y el valor máximo de un entero en Java.* |
| Validación | *monto mayor o igual a cero.* |
| Comentarios | *Se modifica por medio del mouse, por medio de un botón de incremento.* |

Tabla 37:

### 6.2 Árbol de navegabilidad

El árbol de navegabilidad muestra los diferentes flujos de navegación de la aplicación que el usuario puede ejecutar. Para este apartado se utilizo la herramienta Mindmanager documentando cada frame en un mapa mental, que a su vez se conectara por medio de hipervínculos a sus frames adyacentes.

Los archivos con extensión de Mindmanager (.mmap) son externos a este documento y se encuentran almacenados en una carpeta llamada Árbol de navegabilidad [**ver Árbol de Navegabilidad**].