**Arquitectura de Software**

**Introducción:**

Este documento constituye un sumario de la arquitectura de software del proyecto, todo esto con el propósito puntual de brindar una visión de manera incluyente y de conjunto, de tal forma que sea lo más estructurada posible, para luego establecer el papel de esta disciplina en relación con la estrategia arquitectónica utilizada en este proyecto, además de lo anterior mencionado, en él se incluyen herramientas y patrones de diseño utilizados para el desarrollo y tura implementación del mismo

**Propósito**

Proveer una vista general de la organización y distribución de la arquitectura del sistema, además de capturar y describir las decisiones de mayor importancia en la arquitectura. Para describir dicha arquitectura se utilizan distintas vistas del sistema, las cuales muestran distintos aspectos del sistema.

**Alcance**

El alcance de este documento se limita a mostrar una visión general de la arquitectura y proporcionar una base para la creación de un sistema de juego interactivo de estrategia

**Definiciones, acrónimos y abreviaturas**

Referirse al Glosario

**Referencias**

Las referencias aplicables son:

1. Glosario (Glosario.doc)
2. Casos de uso (CasosUso.doc)
3. Plantilla de Software Arquitecture Document, como guía de redacción y de contenidos a considerar. <Http://www.ts.mah.se/RUP/RationalUnifiedProcess/webtmpl/templates/a_and_d/rup_sad.htm>
4. Ejemplo citado en documentación oficial, como una ilustración de la forma correcta ade construir un documento de este tipo.

<http://www.ts.mah.se/RUP/RationalUnifiedProcess/examples/csports/ex_sad.htm>

**Presentación de la Arquitectura**

Para el diseño del sistema de este juego de estrategia se utilizara el modelo de 4 + 1 vistas en el cual se describen las siguientes vistas:

**Vista Logica:**

En esta vista se crea un modelo que describe las estructuras del software para resolver los requerimientos funcionales

**Vista de Procesos**

En esta vista se describen los objetos instanciados y procesos de ejecución que existen en el sistema

**Vista de Despliegue**

En esta vista se describen los equipos que se utilizan en el proceso

**Vista de implementación**

Esta vista se enfoca en la organización de módulos

También dentro de este modelo se describen escenarios que satisfacen la arquitectura y que representan requerimientos importantes para el sistema

**Objetivos y Restricciones de la Arquitectura**

A continuación se muestran los requerimientos y restricciones que tiene un impacto significativo en el sistema:

* Se debe de proveer una interfaz fácil de configurar
* El juego debe de ser portable, quiere decir, que debe de poder ejecutarse en cualquier maquina o marca de procesador
* Se deben de proveer distintas formas utilización de las que el juego provee al momento de juego
* El sistema debe de proteger la información de records o estadísticas de usuarios con punteos más altos obtenidos durante su utilización
* El usuario podrá cargar partidas para continuar en donde se haya quedado la última vez antes de grabar
* El juego permitirá modificar y personalizar las características de cada uno de los personajes, proporcionándoles ventajas sobre tipos de elementos de ataque al igual que las armas que puedan manejar cada uno de ellos

**Vista de Casos de Uso**

Esta vista proporciona una visión global de la funcionalidad que tienen los casos de uso para el sistema y la manera en que estos están organizados, además representando claramente la forma en que interactúan los distintos actores involucrados con los casos de uso que describen al sistema. Los casos de uso involucrados son los siguientes:

* Nuevo Juego
* Abrir juego existente
* Seleccionar bando
* Atacar
* Mover
* Defender
* Tomar arma
* Guardar juego
* Guardar estadísticas
* Detener juego
* Reiniciar juego
* Salir

**Casos de Uso importantes para la arquitectura**



**Nuevo Juego:** El jugador pude seleccionar un juego nuevo con el cual se inicia un juego con estadísticas iníciales y puntajes a 0

**Abrir juego existente:** El jugador puede abrir un juego existente, un juego que ya halla sido iniciado anteriormente y se halla guardado.

**Seleccionar Bando:** El jugador luego de seleccionar un juego nuevo ahora debe seleccionar un bando, para que el sistema conozca para qué lado juegue.

**Atacar:** El jugador puede atacar objetivos dentro del juego los objetivos son otros personajes, el ataque puede ser con armas o sin armas

**Mover:** El jugador puede movilizarse dentro del juego en cualquier dirección

**Defender:** los personajes de otros bandos pueden atacar al personaje del jugador por medio del personaje que lo representa, por lo cual el jugador debe defenderse del ataque.

**Tomar Arma:** el jugador puede tomar armas de las que se encuentran disponibles para el personaje

**Guardar Juego:** Luego que el jugador haya jugado, el jugador tiene la opción de guardar el juego.

**Guardar Estadísticas:** El sistema lleva el control de los puntos que tiene el jugador, el nivel y otros datos, estas estadísticas pueden ser guardadas por el jugador, en cualquier momento.

**Detener Juego:** El jugador puede detener el juego en cualquier momento

**Reiniciar Juego:** Luego que un juego haya sido puesto en pausa el jugador puede volverlo a reiniciar.

**Salir:** El jugador puede salir del juego en el momento que lo desee

**Vista Lógica**

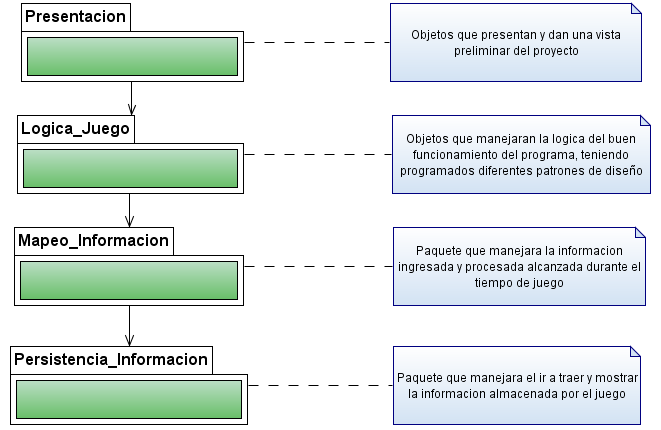
* **Vista General**

La vista general describe las clases más importantes, como estas clases se agrupan en paquetes para proporcionar servicios; como los paquetes forman subsistemas y como estos subsistemas se organizan en capas. También se describen los casos de uso de mayor importancia. Se muestran diagramas de clases para ilustrar las relaciones entre las clases, paquetes, subsistemas y capas de mayor importancia dentro de la arquitectura de la aplicación.

La vista lógica está basada en el modelo MVC, por lo que el sistema esta resumido en tres paquetes principales, Interfaz de Usuario, Reglas de negocio y Persistencia.

El paquete de Interfaz de usuario contiene las clases que serán las responsables de mostrar un ambiente agradable y de fácil uso para el usuario final. El paquete de reglas de Negocio contiene las clases de control de la información de usuarios, productos e inventarios; clases de control de decoración, estrategia, observador y factory, el paquete de Persistencia, contiene las clases de control de información, ordenamiento de la data que se manejara, estadísticas y records.

**Paquetes importantes para la arquitectura**

****

Ver esquema presente en el ejemplo propuesto en la vista lógica del ejemplo oficial de la referencia. Allí se utiliza paquetes para representar las capas de la arquitectura a este nivel de detalle, más adelante se profundiza mas para definir de mejor manera la arquitectura.

**Capa de presentación:** Se encarga de manejar la interacción entre el usuario y la aplicación en ambas vías. Las clases de esta capa representan las pantallas que el usuario utilizará

**Capa de Lógica de Juego:** Esta capa representa los requerimientos del negocio como reglas o estados. Se encarga de hacer funcionar los procesos internos del programa, este paquete contiene las clases que se encargaran de controlar el comportamiento del sistema definido por los casos de uso, el manejo de los datos y su persistencia, también obtiene las interfaces necesarias para comunicar con la capa de lógica del Juego.

**Capa de Mapeo de Información:** Esta capa se encarga de brindar todos los servicios necesarios para sincronizar la capa Lógica con las clases o sistema que manejará los archivos necesarios que permitirán la persistencia de la información almacenada.

**Capa de Persistencia de Información:** Es un sistema que puede manejar archivos binarios y de texto simples, estos archivos manejara información ingresada por cada uno de los usuarios de este juego, por ejemplo: records alcanzados por los jugadores, nombre de perfil creado, estadísticas de armas personalizadas, etc.