**Universidad de San Carlos de Guatemala**

**Facultad de Ingeniería**

**Escuela de Ciencias y Sistemas**

**Analisis y Diseño de Sistemas 2**

**Arquitectura de Software**

|  |  |
| --- | --- |
| **Marlon Alfredo Manzo Iboy** | **2003 13178** |
| **Juan Carlos López Pirir** | **2003 12928** |
| **Erick Giron** | **2003 13492** |
| **Baudilio O. Perello** | **1994 19320** |
| **Edwar Fernando Barrios** | **2003 13015** |
| **Saulo Baten Caravantes** | **2003 13099** |

**Historial de Revisión**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| **23-06-2008** | **1.0.0** | **Versión Preliminar** | **Marlon Manzo** |
| **24-06-2008** | **1.1.0** | **Revisión en vista de Casos de Uso** | **Saulo Baten** |
| **25-06-2008** | **2.0.0** | **Segundo Versión** | **Erick Giron** |
| **26-06-2008** | **3.0.0** | **Tercera Versión** | **Baudilio Perello** |

**Arquitectura de Software**

1. **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

* 1. ***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.

* 1. ***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

* 1. ***Definiciones, acrónimos y abreviaturas***

Referirse al Glosario

* 1. ***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>

* 1. ***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

1. **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

1. **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
  1. ***Casos de Uso importantes para la arquitectura***



* + 1. **Nuevo Juego.**

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

* + 1. **Abrir juego existente**

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

* + 1. **Seleccionar Bando**

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

* + 1. **Atacar**

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

* + 1. **Mover**

El jugador puede movilizarse dentro del juego en cualquier dirección

* + 1. **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.

* + 1. **Tomar Arma**

El jugador puede tomar armas de las que se encuentran disponibles para el personaje

* + 1. **Guardar Juego**

Luego que el jugador haya jugado, el jugador tiene la opción de guardar el juego.

* + 1. **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.

* + 1. **Detener Juego**

El jugador puede detener el juego en cualquier momento

* + 1. **Reiniciar Juego**

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

* + 1. **Salir**

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

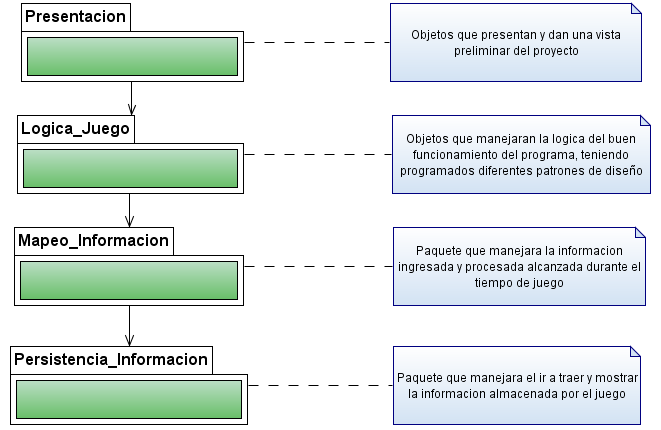
1. **Vista Lógica**
   1. **Vista General**

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 lógica de funcionamiento e instalación**

****

Ver el 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.

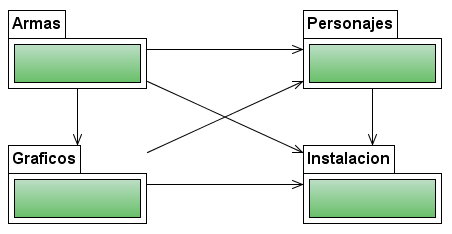
**Capa de Mapeo de Información:** Esta capa se encarga de brindar todos los servicios necesarios para sincronizar la capa Lógica con las clases o sistema que manejará los archivos necesarios que permitirán la persistencia de la información almacenada.

**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.

**Vista de Despliegue**

En esta sección se describe la configuración del hardware sobre el cual será desplegada y ejecutada la aplicación. Se muestra la configuración de los nodos físicos y como las tareas establecidas en la vista de procesos se reparten en los nodos

**Paquetes importantes para la lógica de desarrollo**



Ver el esquema presente en el ejemplo propuesto en la vista lógica de desarrollo. 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

**Paquete Armas:** Se manejaran y desarrollaran todas las clases necesarias para poder desarrollar los diferentes tipos de armas permitidas en el juego, al igual que se manejara la lógica de personalización y manejo de estadísticas para cada uno de las armas de cada jugador.

**Paquete Gráficos:** Este paquete manejara todas las configuraciones necesarias con respecto a la tarjeta de video en la cual se esté trabajando y poder diseñar de una mejor manera cada una de los objetos que se vean involucrados para el buen desarrollo y lógica del juego.

**Paquete Personajes:** Aquí se diseñan y personalizan a cada uno de los personajes que el usuario haya creado dentro de su perfil como jugador.

**Paquete de Instalación:** Paquete que contendrá todos los programas y herramientas necesarias de instalación del producto, para poder hacer que esta solución sea a un buen nivel de portabilidad, es decir, que sea capaza de poder ejecutarse en diferentes computadoras con diferente arquitectura de software o diferentes sistemas operativos existentes.

**Vista de Despliegue**

En esta sección se describe la configuración del hardware sobre el cual será desplegada y ejecutada la aplicación. Se muestra la configuración de los nodos físicos y como las tareas establecidas en la vista de procesos se reparten en los nodos.



**Computadora:** La aplicación podrá ser instalada en una sola maquina, este nodo representa el hardware en el cual podremos tener instalada nuestra aplicación, contando con los requerimientos mínimos especificados en el manual de usuario, presente al momento de su adquisición

**Joystick:** Este dispositivo será como un medio de comunicación para la aplicación, la aplicación podrá ser controlada por un Joystick, CamePade, etc. Para facilitar el nivel de uso y capacidad de control para los usuarios finales

**Teclado:** La aplicación también puede ser controlada por un dispositivo de entrada muy común en la mayoría de computadoras, un teclado que lo más normal posible.

**Vista de Implementación:**

La aplicación está basada en un modelo de tres capas, la capa de presentación, lógica del negocio y capa de persistencia, todas estas capas están presentes y unidas lógicamente en una sola computadora, cada una de las capas fueron programadas y configuradas por una persona diferente, durante la configuración y desarrollo se conto con un sistema de administración de cambios (SVN) y una computadora central que contenía los archivos necesarios d configuración, en pocas palabras se trabajo bajo un modelo de desarrollo centralizado.

**Tamaño y Rendimiento**

La arquitectura de software soporta todos los requerimientos de rendimiento establecidos en el documento de especificaciones suplementarias. El cumplimiento de los requerimientos es logrado a través de una arquitectura simple de una sola maquina. Esta computadora ejecuta todos los procesos necesarios para su buen funcionamiento, utilizando un mínimo de espacio en disco y en memoria, esto se logra ejecutando la mayor carga de la aplicación.

* La computadora que tendrá instalada la aplicación deberá tener capacidad de poder soportar aplicaciones java, usando el JRE mas reciente.
* Se debe de tener como mínimo 20MB de disco Duro, y no más de 32 MB de RAM
* De preferencia, poder contar unos paquetes o librerías de desarrollo java, en caso de que la aplicación detecte que sea necesario instalar alguna librería de aplicación.

**Calidad**

La arquitectura de software seleccionada soporta todos los requerimientos de calidad establecidos en el documento de especificaciones suplementarias

* La aplicación será compatible con los distintos sistemas operativos que en la actualidad son muy usados, Linux y Windows
* La interfaz de usuario será diseñada para proveer una fácil utilización para que usuarios inexpertos no necesiten capacitación exhaustiva para su utilización
* El sistema proveerá de una completa ayuda local, esta ayuda mostrara tutoriales paso a paso de cómo utilizar las distintas funcionalidades del sistema
* El sistema será portable tanto dentro como para clientes Windows como Linux o Unix