Facultad 3

Arquitectura de software para el prototipo del videojuego Isla del Son

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Leandro González Lambert
Tutor: Ing. Emilio Enrique Cardero Alvarez
Co-tutor: MSc. Yarina Amoroso Fernández

La Habana, noviembre de 2023

Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título ***"Arquitectura de software para el prototipo de videojuego Isla del Son"*** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 29 días del mes de noviembre del año 2023.

Leandro González Lambert

_____

Firma del Autor

Ing. Emilio Enrique Cardero Álvarez

_____

Firma del Tutor

MSc. Yarina Amoroso Fernández

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Emilio Enrique Cardero Álvarez: graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2017.

Correo: eecardero@uci.cu

MSc. Yarina Amoroso Fernández: Máster en Ciencias Jurídicas, Mención Derecho Público, Universidad de Valencia. España. Profesora Auxiliar de la Universidad de Ciencias Informáticas con experiencia en la Educación Superior. Estudios de Doctorado "Aspectos éticos y jurídicos de la información digital". DEA, 2008. Universidad de Valencia. España. Experto del Comité UNESCO para la Preservación del Patrimonio Digital. Autora del Proyecto MENJUR, 1988. CONSULTA LEGAL: Sistema de distribución electrónica de información jurídica, 1992. Base de Datos del Diccionario Especializado de Derecho e Informática, 1994. ASESOR: “Estudio Comparado de las Constituciones Iberoamericanas”, "Historia Constitucional Cubana". Estudio terminológico de la Legislación Cubana, en coautoría con la Dra. Manuela Sassi, del Instituto de Lingüística Computacional de Pisa, Italia, 1996. Coautora del CD “Legislación Cubana Digital” 2000. Sitio WEB de la Gaceta Oficial de la República de Cuba 2001. Sitio de las Constituciones cubana 2003. Compilación de Derecho Notarial Cubano 2004. Modelo y Diseño del Sistema de Gestión del Registro del Estado Civil cubano 2004. Coautora de las publicaciones: “Derecho y Tecnologías de la Información”, Chile 2002. “Gobierno, Derecho y Tecnologías: las Actividades de los Poderes Públicos, THOMSON, 2006. “Construyendo el eG y la Sociedad del Conocimiento, UNESCO 2011. “Ciudadanas 2020, el Gobierno de la Información, Chile, 2011.

Correo: yarina@uci.cu.

AGRADECIMIENTOS

Agradecer a todos las personas que hicieron posible esta investigación. Al tribunal, a los tutores y el oponente por su atención y la ayuda prestada. Agradecer a los profesores y profesionales que a lo largo de estos años ayudaron en mi formación como ingeniero. Mis compañeros de aula, en especial Antonio, con quien he compartido toda esta etapa universitaria, muchos momentos de estudio y juegos, grandes vivencias que hoy son grandes anécdotas. Agradecer a mis familiares, en especial mis padres, siempre han sido el apoyo base para poder seguir adelante, de ellos aprendo día a día y hoy con el final de este proyecto compruebo una de sus grandes enseñanzas: “Eres capaz de hacer cualquier cosa siempre que te lo propongas.” Quiero agradecer además a mi abuela, que siempre me acobija cuando lo necesito, gracias por ser comprensiva, sé que este proyecto ha consumido mucho mi tiempo. Agradecer a los amigos, algunos tan especiales, que a la primera llamada de necesidad acuden de inmediato, gracias a todos los miembros del Clan Happy, en especial Alejandro. Agradecer los momentos de ocio con grandes compañeros y amigos, que noche a noche nos hemos convertido en una gran familia, que salva el mundo cada semana, en especial a Luis Angel. Agradecer a todos los presentes por el interés y respeto con el que han participado en la exposición de estos resultados.

DEDICATORIA

Dedicado a Mariano Lambert Matos, mi abuelo, quien hizo posible esta aventura. Donde sea que estés, quiero que sepas, que “La batalla está ganada”.

RESUMEN

La arquitectura de software se encarga de prevenir los problemas que pueden surgir en un proceso de desarrollo de software, estableciendo una serie de características para el desarrollo de un producto informático. Esta investigación tiene como objetivo principal diseñar una arquitectura de software para el videojuego “Isla del Son”, que integra algunos minijuegos ya desarrollados y los servicios de la plataforma Cosmox en un solo producto. Para ello se hace una investigación de los distintos estilos y patrones arquitectónicos existentes que más se apegan a las necesidades de la investigación y se determinan las distintas vistas posibles para la arquitectura. Una vez diseñada la arquitectura se desarrolla el prototipo de software bajo dicha arquitectura a través del cual se valida la solución propuesta empleando técnicas de evaluación basada en prototipos, escenarios en conjunto con el método de Análisis de Acuerdos de Arquitectura de Software (ATAM). Mediante esta técnica se pudieron determinar las fortalezas y debilidades de la arquitectura.

PALABRAS CLAVE

Arquitectura, Cosmox, videojuegos

ABSTRACT

Software architecture is responsible for preventing problems that may arise in a software development process, establishing a series of characteristics for the development of a computer product. The main objective of this research is to design a software architecture for the video game “Isla del Son”, which integrates some minigames already developed and the services of the Cosmox platform in a single product. To do this, an investigation is carried out into the different existing architectural styles and patterns that best fit the needs of the research and the different possible views for the architecture are determined. Once the architecture is designed, the software prototype is developed under said architecture through which the proposed solution is validated using evaluation techniques based on prototypes, scenarios in conjunction with the Software Architecture Agreement Analysis (ATAM) method. Using this technique, the strengths and weaknesses of the architecture could be determined.

KEYWORDS

Architecture, Cosmox, videogames

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El ser humano siempre ha sentido la necesidad de comunicarse, buscar saber, obtener información. Es por ello que los grandes saltos evolutivos de la humanidad tienen como hito la instauración de nuevos instrumentos de comunicación marcado en los últimos años por las grandes transformaciones producidas por la tecnología. Las tecnologías, con su desarrollo, han llegado al avance de ser interactivas con las propias personas que la consumen. La Real Academia Española (RAE)[1] tiene cuatro definiciones distintas para la palabra tecnología, la que más se adapta al tema a tratar es el conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico (RAE, 2023). Asimismo, también define el término interactivo como el adjetivo que califica a un sustantivo que permita la interacción, especialmente entre una computadora o un programa informático y su usuario (RAE, 2023).

De acuerdo a lo planteado anteriormente, se puede definir la tecnología interactiva como la relación humano-computadora, donde ambos entablan una comunicación a través de reconocimiento de gestos, multimedia 3D, realidad virtual, trabajo de apoyo computarizado y soporte y lenguaje natural; utilizando varios códigos como presentación de la información, por ejemplo: imágenes, videos y texto (Olivas, s.f.).

La Universidad de las Ciencias Informáticas (UCI) cuenta con un Centro de Tecnologías Interactivas (VERTEX) adscrito a la Facultad 4, que desarrolla productos y servicios informáticos asociados a los Entornos Interactivos 3D con un alto valor agregado, resultado de un ciclo completo de Investigación + Desarrollo + innovación (I+D+i). Entre las funciones de este centro se encuentra (Universidad de las Ciencias Informáticas, s.f.):

- Participación activa en la informatización del país desarrollando soluciones que requieran de la Interacción con Entornos Virtuales 3D.

[1] La Real Academia Española es una institución con personalidad jurídica que tiene como misión principal velar por que los cambios que experimente la lengua española en su constante adaptación a las necesidades de sus habitantes no quiebren la esencial unidad que mantienen en todo el ámbito hispánico. (RAE, 2021)

- Consolidación de los productos y servicios ofrecidos como rubros exportables, que cuenten con una alta competitividad y estén basados en principios de software y reusabilidad.
- Formación de personal altamente calificado en la investigación y desarrollo de soluciones con un alto sentido de la profesionalidad, competitividad y compromiso en la transformación de la sociedad.

Por las funciones del Centro VERTEX, la UCI, se encuentra comprometida en una propuesta del Ministerio de Educación Superior en conjunto con el Ministerio de Cultura. La cual surge en el mes de marzo del 2023, cuando la embajada de Cuba ante la UNESCO[2], presenta el expediente de candidatura del son como Patrimonio Cultural Inmaterial de la Humanidad[3], resultado de un proceso de transculturación con las culturas africanas, española y aborígenes (Esquivel, 2022). Cuba es constantemente bombardeada por culturas foráneas muy atractivas que desvían el interés de la población cubana de su cultura autóctona, con una mayor influencia en las jóvenes generaciones. Para apoyar la premisa de la embajada cubana, los Ministerios de Educación Superior y Cultura han propuesto a la UCI el desarrollo de un elemento con tecnología interactiva que promueva la cultura del son en el país.

Para cualquier tipo de producto que desee desarrollarse, se necesita un patrón que permita a los desarrolladores comunicarse entre sí, una estructura común que permita comprender en qué se basa el sistema que se está desarrollando. Por ello, surge la necesidad de escribir software, disciplina que ha evolucionado hasta convertirse en una profesión que no sólo se ocupa de crear programas informáticos, sino de optimizar su calidad basada en la estructura. Esta necesidad surge debido a que esta rama de la informática selecciona y diseña con base en requerimientos y restricciones (Rollings & Morris, 1999).

Hacia finales de los años ochenta y principios de los noventa, comienza a gestarse de forma más clara la idea de que las aplicaciones tienen una morfología, una estructura. Los autores

[2] La UNESCO es la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura, organismo dedicado a conseguir el establecimiento de la paz mediante la cooperación internacional en los ámbitos de la educación, la ciencia, la cultura y la comunicación e información. (UNESCO, s.f.)

[3] El patrimonio cultural inmaterial se refiere a las prácticas, expresiones, saberes o técnicas transmitidos por las comunidades de generación en generación.

Perry y Wolf publican en 1992 el artículo "*Foundations for the Study of Software Architecture*", lo que sería el punto de partida para la Arquitectura de Software tal como la conocemos y son quienes comienzan a utilizar este término (Perry & Wolf, 1992). Actualmente, se define a la arquitectura de software como quien determina de forma abstracta los componentes de un sistema, las interfaces y la comunicación entre ellos, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan soluciones a problemas de manera eficiente (Pressman, 2002).

De acuerdo a la propuesta ministerial realizada al centro, se puede plantear la siguiente **situación problemática**:

- Existen distintos componentes que son algunos videojuegos que estudiantes de la UCI vinculados al Centro VERTEX han desarrollado, a partir de ahora definidos como minijuegos, para diferenciar cuando la investigación haga referencia a estos videojuegos en específico.
- El Centro de VERTEX ha desarrollado una plataforma denominada Cosmox para construir recursos informáticos como un establecimiento de Ranking y Autenticación, sin embargo, no existe una arquitectura definida para integrar distintos componentes que permita una interrelación con la plataforma de Cosmox.

Por la **situación problemática** antes expuesta, se propone realizar la investigación a partir del siguiente **problema a resolver**: ¿Cómo se pueden integrar los minijuegos en conjunto con la plataforma de Cosmox?

Se define como **objeto de estudio**: Diseño de arquitectura de software.

A raíz del problema de investigación se define como **objetivo general:** Diseñar una arquitectura de software que permita integrar videojuegos y consuma los servicios de la plataforma Cosmox.

Partiendo del objetivo general expuesto, el **campo de acción** de esta investigación es:

Diseño de Arquitectura de Videojuegos.

Para darle cumplimiento al objetivo antes planteado se proponen las siguientes **tareas de investigación**:

1. Analizar las arquitecturas de software que integren componentes y utilicen recursos externos para identificar elementos a reutilizar.

2. Analizar las arquitecturas de software para los videojuegos.
3. Seleccionar las características de las arquitecturas analizadas para la arquitectura que se pretende diseñar.
4. Diseñar la arquitectura de software para una tecnología interactiva integrada por varios componentes y la integración de los servicios de la plataforma Cosmox.
5. Implementar un software como prototipo funcional que permita validar la arquitectura propuesta.
6. Integrar los servicios de la plataforma Cosmox en el prototipo desarrollado.
7. Validar la propuesta arquitectónica.

Como **métodos científicos de la investigación** se emplearon los siguientes:

Como métodos **teóricos**:

- **Histórico – Lógico:** Se analizó la evolución y las tendencias de las actuales arquitecturas de software.
- **Analítico – Sintético:** Se extrajo y se analizó la información de las principales arquitecturas de software utilizadas en la UCI.
- **Modelación:** Se modelaron los diagramas correspondientes a la metodología de software que se pretende desarrollar.

Como métodos **empíricos**:

- **Consulta bibliográfica:** Se consultaron distintas fuentes bibliográficas durante la investigación.
- **Pruebas:** Se realizaron para comprobar que la propuesta elaborada satisface el objetivo presentado, así como su correcto funcionamiento.
- **Entrevista:** Estableció la comunicación con los clientes y la información acerca de los requisitos del videojuego.

La investigación se ha estructurado con una introducción, tres capítulos y conclusiones, los capítulos han sido distribuidos de la siguiente forma:

- **Capítulo 1. Fundamentos y referentes teórico-metodológicos sobre la arquitectura de software para videojuegos:** En este capítulo se realizan las investigaciones de las distintas arquitecturas de software, características, ventajas y

desventajas de las distintas formas de estructuración y vistas de las arquitecturas de software en general y arquitecturas de videojuegos en particular.

- **Capítulo 2. Diseño de la propuesta de arquitectura de software para la integración de videojuegos y servicios de la plataforma Cosmox:** En este capítulo se diseña una propuesta para dar solución a la situación problemática de esta investigación. En la solución se diseña una arquitectura de software para la integración de videojuegos y los servicios de la plataforma Cosmox.
- **Capítulo 3. Validación de la propuesta de Arquitectura de Software para videojuegos y servicios de la plataforma Cosmox:** En este capítulo se exponen algunos conceptos asociados a cómo se evalúa una arquitectura de software, los métodos de cómo evaluar la arquitectura y se aplican las técnicas de pruebas a un prototipo de software desarrollado sobre esa arquitectura.

CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE LA ARQUITECTURA DE SOFTWARE PARA VIDEOJUEGOS

En este capítulo se realizan las investigaciones de las distintas arquitecturas de software, características, ventajas y desventajas de las distintas formas de estructuración y vistas de las arquitecturas de software en general y arquitecturas de videojuegos en particular.

1.1 Arquitectura de Software

De acuerdo con el *Software Engineering Institute* (SEI)[4], la Arquitectura de Software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos (Bass, Clements, & Kazman, 1997). El Instituto de Ingeniería Eléctrica y Electrónica IEEE[5] propone: "La Arquitectura de Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución" (IEEE Computer Society, 2000). Garlan y otros autores, por su parte, refieren "…es el nivel del diseño del software donde se definen la estructura y propiedades globales del sistema…" "… se centra en aquellos aspectos del diseño y desarrollo que no pueden tratarse de forma adecuada dentro de los módulos que forman el sistema" (Garlan & Perry, 1995).

La arquitectura de software tiene que ver con el diseño y la implementación de estructuras de software a alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales. Es una solución a las dificultades que con el tiempo se han ido descubriendo, desarrollando formas y guías generales, con base a las cuales se puedan resolver esos problemas (Cruz, 2017).

[4] Instituto de Ingeniería de Software, líder en ingeniería de software y ciberseguridad, investigan problemas complejos de ingeniería de software, ciberseguridad e ingeniería de Inteligencia Artificial; crean y prueban tecnologías innovadoras y la transacción de las soluciones maduras a la práctica (Universidad Carnegie Mellon, 2023).

[5] El Instituto de Ingenieros Eléctricos y Electrónicos es la mayor organización profesional técnica del mundo, que agrupa profesionales que se dedican al avance en la innovación tecnológica y a la excelencia en beneficio de la humanidad (IEEE, 2018).

Con la ausencia de la arquitectura de software en el proceso lógico del desarrollo de software, es posible que diversos elementos del sistema contengan deficiencias en su estructura y puedan ocasionarse problemas, tales como (Hofmeister, Krutchen, Nord, Obbink, & Ran, 2007):

- Falla en cumplir con la calidad necesaria.
- Falla en intentar soportar las necesidades de negocio.
- Falta de compromiso con el usuario.

1.2 Estilos de la Arquitectura de Software

Los estilos arquitectónicos describen una clase de arquitectura o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran rápidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez identificados estos estilos, es lógico y natural pensar en reutilizarlos en situaciones semejantes que se presenten en el futuro (Reynoso, Investigación, Publicaciones y Cursos de Antropología, 2004).

Estos estilos, son un conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer el sistema junto con las restricciones de la forma en que la composición se lleve a cabo. Los componentes, se pueden distinguir por la naturaleza de su computación. Los tipos de componentes también se pueden distinguir conforme a su forma de empaquetado, de acuerdo con las formas en que interactúan con otros componentes (Shaw & Clements, 1996).

El estilo arquitectónico es una descripción del patrón de los datos y la interacción de control entre componentes, ligada a una descripción informal de los beneficios e inconvenientes aparejados por el uso del estilo. Son artefactos de ingeniería importantes porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos. Ofrecen evidencia basada en la experiencia sobre la forma en que se ha utilizado históricamente cada clase, junto con razonamiento cualitativo para explicar por qué cada clase tiene esas propiedades específicas (Klein & Kazman, 1999).

Todas las características que se definen en una arquitectura de software, forman parte de un estilo, proporcionando una visión general de cómo se organiza el sistema. De estos estilos surgen los patrones arquitectónicos que proporcionan una solución detallada de cómo implementar el estilo arquitectónico.

1.3 Patrones de la Arquitectura de Software

Un patrón en la arquitectura de software es la conceptualización de una solución genérica y reutilizable, aplicable a un problema de diseño de software en un contexto determinado, satisfaciendo las necesidades del negocio. Son la representación de las buenas prácticas y estructuras de diseño probadas, de modo que puedan reutilizarse. El objetivo es reutilizar las experiencias y conocimientos arquitectónicos que han dado buenos resultados en el pasado (Ramirez, 2023).

Su elaboración para una solución concreta puede verse como un proceso de selección, adaptación y combinación de patrones. El arquitecto de software de un proyecto debe decidir cómo reutilizar un patrón, hacer los ajustes necesarios al contexto específico, a las restricciones del problema, y a la solución que está diseñado (Ramirez, 2023).

A medida que los videojuegos han evolucionado, estos se han beneficiado de la aparición de herramientas que facilitan su desarrollo, automatizando determinadas tareas y ocultando la complejidad inherente a muchos procesos de bajo nivel. Las arquitecturas de software hacen la vida más sencilla a los desarrolladores de videojuegos, ya que (Ernest):

- El proceso de diseño de cualquier aplicación de software suele ser iterativo y en diferentes etapas de forma que se vaya refinando con el tiempo.
- La etapa de diseño es una tarea crítica en el ciclo de vida del software.
- Al construir aplicaciones similares se presentan situaciones recurrentes y que se asemejan a situaciones pasadas.

Básicamente cuando un juego contiene parte de su lógica no resulta práctico reutilizarla para otro juego, ya que implicaría modificar el código fuente sustancialmente. Sin embargo, si dicha lógica o comportamiento no está definido a nivel de código, sino, por ejemplo, mediante una serie de reglas específicas a través de distintos archivos, entonces la reutilización sí es posible, ya que optimiza el tiempo de desarrollo (Ernest).

No existe un número fijo de patrones arquitectónicos de software, ya que estos continúan evolucionando con el tiempo a medida que surgen nuevas tecnologías y enfoques. Cada uno tiene sus propias fortalezas y debilidades.

1.3.1 Arquitectura en capas

Es una arquitectura de una organización jerárquica, tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Los conectores se definen mediante los protocolos que determinan las formas de la interacción. Las restricciones topológicas pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con sus capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma. Es suponer que, si esta exigencia se relaja, el estilo deja de ser puro y pierde algo de su capacidad heurística (Garlan & Perry, Advances in Software Engineering and Knowledge Engineering Vol 2, 1993). Es decir, se pierde la posibilidad de reemplazar totalmente una capa sin afectar las restantes, disminuyendo la flexibilidad del conjunto y complicando su mantenimiento.

Un ejemplo de esta arquitectura en el desarrollo de videojuegos fue publicado por Ricardo Emmanuel Gutiérrez Hernández, Francisco Álvarez Rodríguez y Jaime Muñoz, de la Universidad Autónoma de Aguascalientes en México, bajo el nombre de “Arquitectura para Videojuegos Serios con Aspectos Culturales” que consiste en una arquitectura con 5 capas (Gutiérrez-Hernández, Álvarez, & Muñoz-Arteaga):

- La capa contexto cultural contiene la información del usuario y recursos de juego donde se agrupan todos los datos de usuario y archivos multimedia para aplicarlos en el videojuego.
- La capa de aplicación contiene la especificación de los elementos genéricos del juego, en la base a la lógica y los recursos disponibles, respondiendo y proporcionando una interfaz para administrar eventos y retroalimentación al usuario.
- La capa de objetos del juego se integran los módulos y objetos responsables de ofrecer las características propias del juego, gestionando los recursos lógicos y las vistas entre los componentes y paquetes del videojuego para un escenario.
- La capa de escenario integra las clases relativas al correcto funcionamiento interno del videojuego como los objetos, multimedia, reglas de interacción, así como las vistas consideradas en la clase general del objeto, proporcionando la simulación de la variedad de elementos de objetos y personajes de videojuego.

- La capa de interfaz de usuario muestra el conjunto de escenarios, objetos del videojuego y se procesan los datos de entrada del usuario para realizar las diferentes interacciones en la historia o niveles de juego del contexto cultural del usuario/jugador.

Una estructura similar podría aplicarse para la solución del proyecto, de esta forma se pueden aprovechar la ventaja del modularidad, permitiendo las siguientes ventajas:

- Separar cada una de las capas en módulos que les permita funcionar de forma independiente, donde estos módulos serían los minijuegos y los servicios de la plataforma Cosmox.
- Este estilo permite una gran escalabilidad, ya que, en un futuro se pueden agregar más videojuegos al software final o incluso modificar o eliminar los minijuegos integrados inicialmente, ya que cada capa tiene su propia responsabilidad.
- Ofrece una mayor seguridad, ya que permite separar los datos críticos contenidos en la plataforma Cosmox y la lógica de la interfaz del usuario u otros componentes, aumentando la seguridad al no permitir que ciertas capas interactúen con otras.

Sin embargo, cada capa solo puede interactuar con sus adyacentes, esto supone complicaciones en el diseño de la arquitectura, ya que implica que su estructuración debe ser bien precisa para que el flujo de información sea efectivo.

1.3.2 Arquitectura basada en componentes

Un componente de software es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas (Clemens, 2002). Las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución. En cuanto a las restricciones, puede admitirse que una interfaz sea implementada por múltiples componentes. Usualmente, los estados de un componente no son accesibles desde el exterior (Clemens, 2002).

Esta forma de desarrollar software suele centrarse en la construcción de sistemas a partir de componentes reutilizables, de los cuales se crea una dependencia. Utilizando este modelo, si algún componente falla, puede afectar la funcionalidad de todo el sistema, estos componentes

pueden cambiar con el tiempo, lo que puede afectar la compatibilidad del sistema (Binnie, 2023).

Si se piensa en los minijuegos a integrar junto con la plataforma Cosmox como componentes independientes podría ser la arquitectura perfecta para hacer efectiva su integración.

1.3.3 Arquitectura para videojuegos integrada con Cosmox

La selección de un buen diseño de la arquitectura de software en un videojuego, permitirá su (Morales, Nava, Fernández, & Rey):

- Reusabilidad.
- Extensibilidad.
- Manejabilidad.
- Reducción de tiempos de producción.
- Reducción de gastos de recursos.

La arquitectura basada en componentes permite utilizar cada uno de los minijuegos y a la plataforma Cosmox como componentes de su estilo arquitectónico, por ende, permitirá la comunicación entre ellos a través de una interfaz principal que se encargará del tráfico de los recursos de cada componente de manera individual. De esta forma, se define como patrón arquitectónico a utilizar en esta investigación la **arquitectura basada en componentes.**

1.4 Vistas de la Arquitectura de Software

Las vistas de una arquitectura de software es un término que se refiere a cómo se presenta o se visualizan las arquitecturas en un sistema de software. Según Pressman, una arquitectura de software puede tener múltiples “vistas” o perspectivas, cada una de las cuales se centran en ciertos aspectos del sistema. Cada vista proporciona una representación diferente del sistema para abordar diferentes preocupaciones o intereses (Pressman, 2002).

Las arquitecturas de software se enfrentan a la abstracción, descomposición y composición usando estilos y patrones. Para describir una arquitectura de software se usa un modelo compuesto de múltiples vistas o perspectivas. Con el fin de resumir estos modelos, el creador de la metodología RUP propone cinco vistas (Kruchten, 1999):

- La vista lógica como objeto de diseño.

- La vista de procesos como captura de la concurrencia y sincronización como los aspectos del diseño.
- La vista física como el mapeo de la distribución del software en el hardware.
- La vista de desarrollo como la organización estática del software en su entorno de desarrollo.
- Los escenarios como los ensambladores de los elementos mostrados en las vistas anteriores, la descripción de una arquitectura puede ser organizada alrededor de las vistas lógicas de procesos, física y de desarrollo, mostrándose en los escenarios. La arquitectura, de hecho, evoluciona a partir de estos escenarios.

Un estudio realizado en el procesamiento de imágenes, sistemas de comunicaciones, control e instrumentación, para conocer las estructuras de mayor importancia en su arquitectura, dio origen a cuatro vistas dentro de las cuales se describen las estructuras principales del sistema desde una perspectiva particular (González):

- La vista conceptual para describir el sistema en términos de elementos principales de diseño y las relaciones entre estos, dentro de un dominio determinado. Esta es una vista independiente de las decisiones de implementación y enfatiza en los protocolos de interacción entre los elementos.
- La vista de módulos para capturar la descomposición funcional y las capas del sistema, el sistema es descompuesto lógicamente en subsistemas, módulos y unidades abstractas, donde cada capa representa las distintas interfaces de comunicación permitidas entre los módulos.
- La vista de ejecución para describir la estructura dinámica del sistema en términos de sus elementos en tiempo de ejecución, para modelar las tareas operativas del sistema, procesos, mecanismos de comunicación y asignación de recursos, los aspectos principales que considera esta vista son el desempeño y el entorno de ejecución.
- La vista de código para organizar el código fuente en directorios, archivos y bibliotecas. Algunos de los aspectos que se incluyen son, los lenguajes de programación a utilizar, herramientas de desarrollo, la administración de la configuración y de la estructura y organización del proyecto.

Estas vistas son las que se pretenden utilizar para diseñar la arquitectura de esta investigación, ya que, en el desarrollo de los videojuegos, la descripción a través de una especificación de requisitos y casos de uso se hace complejo, ya que genera una gran cantidad de diagramas de casos de uso para detallar los procesos, descartando la opción de Kruchten. El último estudio descrito se ajusta más al desarrollo de tecnologías interactivas que no necesiten una documentación tan precisa para la definición de sus funcionalidades a través de casos de uso, sino el uso de las funcionalidades conceptuales, como en los videojuegos.

Conclusiones del capítulo

De acuerdo a la investigación de las distintas vistas arquitectónicas que existen, los estilos y patrones arquitectónicos para el desarrollo de software, teniendo en cuenta que se pretende diseñar una arquitectura de software que integre los minijuegos y utilice los servicios de Ranking y Autenticación de la plataforma Cosmox, se ha definido de entre los patrones candidatos a la **arquitectura basada en componentes**.

CAPÍTULO 2: DISEÑO DE LA PROPUESTA DE ARQUITECTURA DE SOFTWARE PARA LA INTEGRACIÓN DE VIDEOJUEGOS Y SERVICIOS DE LA PLATAFORMA COSMOX

En este capítulo se diseña una propuesta para dar solución a la situación problemática de esta investigación. En la solución se diseña una arquitectura de software para la integración de videojuegos y los servicios de la plataforma Cosmox.

2.1 Descripción de la propuesta solución

Como resultado para esta investigación se diseña una arquitectura de software que integra los distintos minijuegos adaptados para el videojuego principal y los servicios de Ranking y Autenticación de la plataforma Cosmox. Dicha arquitectura trata a los videojuegos y los servicios como componentes que se integran entre sí a través de interfaces con las que interactúan.

Esta propuesta de arquitectura, que utiliza un patrón de una arquitectura basada en componentes, consta con distintas vistas que determinarán una mayor comprensión de su estructura y funcionamiento.

2.2 Vistas Arquitectónicas

Definidas las siguientes cuatro vistas principales para esta arquitectura en el capítulo anterior.

2.2.1 Vista Conceptual

Esta vista describirá el sistema en elementos del diseño y las relaciones entre ellos. Es una vista independiente de las decisiones de implementación, enfocándose en la interacción entre los distintos elementos.

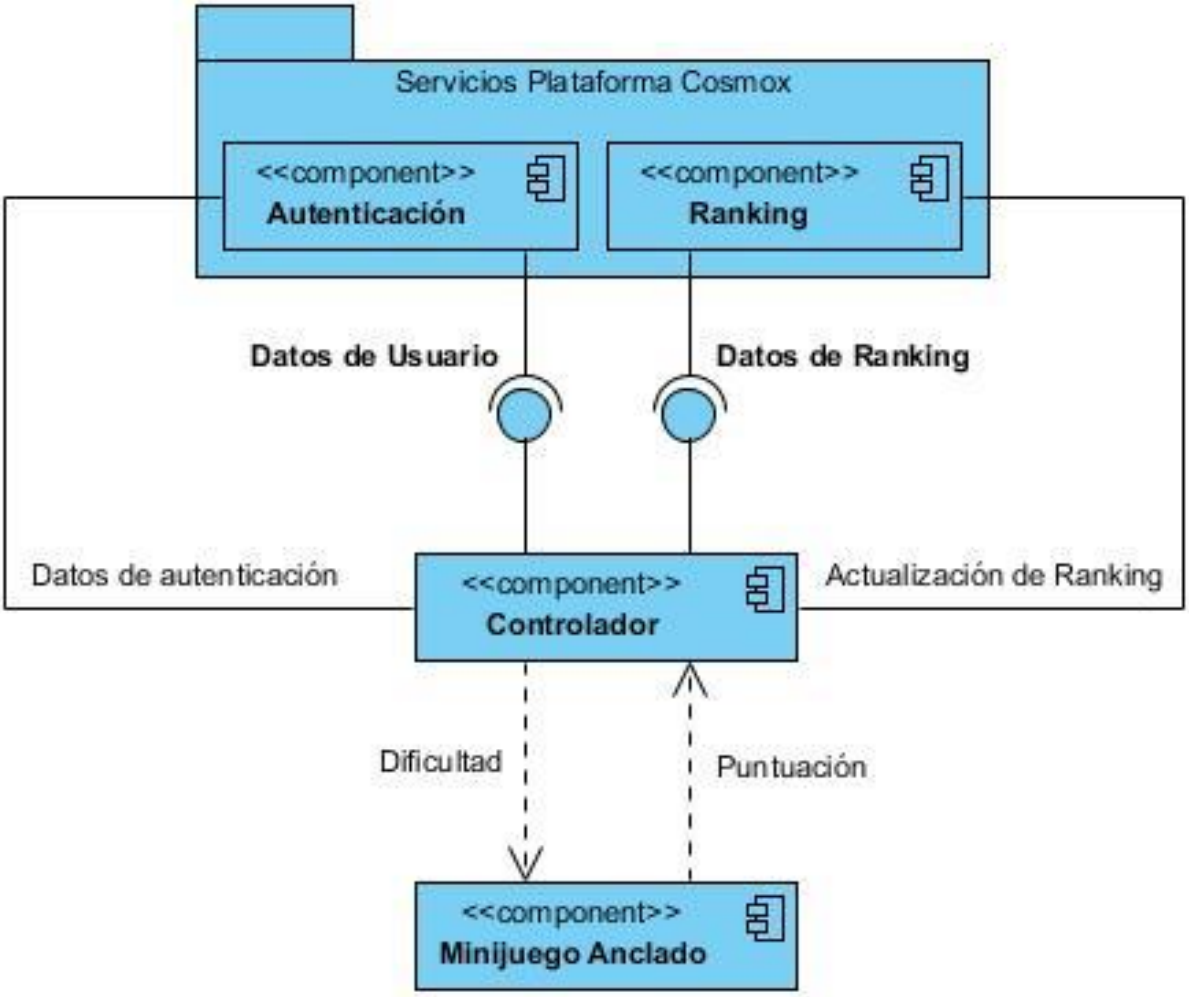

Figura 1 Diagrama de Componentes de la Vista Conceptual de la Arquitectura de Software.

Estructura conceptual de la Arquitectura de Software:

- Tiene un componente principal, definido como **Controlador**, es el encargado de establecer la comunicación entre todos los componentes. Gestiona los datos recibidos por los servicios de la plataforma Cosmox e interactúa con ellos para enriquecer los datos. Se encarga de establecer la dificultad del minijuego antes de comenzar y una vez finalizada la partida recibirá la puntuación obtenida por el usuario.
- Cuenta con un paquete que engloba y ofrece los datos del usuario los componentes que representan los servicios que ofrece la plataforma Cosmox:
    - El servicio de **Autenticación** que ofrece el servicio de contraste de datos de información de un usuario a través de la interfaz **Datos de Usuario**, para una vez contrastados enviar al Controlador los datos del usuario correspondiente. Es

importante recalcar que este servicio requiere un registro previo en la web de la plataforma Cosmox.

    - El servicio de **Ranking** que ofrece el servicio de los datos de Ranking de la plataforma a través de la interfaz **Datos de Ranking**, estos datos son actualizados a partir de los datos que se reciben del componente **Controlador**.

- El componente **Minijuego Anclado** representa cualquiera de los minijuegos anclados u otro videojuego que pueda ser anclado en versiones posteriores. Recibe los datos de la dificultad con la que debe comenzar la partida y una vez finalizada la partida enviará la puntuación obtenida por el usuario al componente **Controlador**.

2.2.2 Vista de Módulos

La vista de los módulos captura la descomposición funcional y las capas del sistema, en un sistema descompuesto lógicamente en subsistemas, donde las capas representan las distintas interfaces de comunicación permitidas entre los módulos (González). Cuenta con tres capas principales, donde cada una encierra a los módulos que establecen las funciones principales relacionadas a la representación de su parte lógica.

- La **Capa de Servicios** engloba los servicios que la arquitectura consume, en este caso los servicios ofrecidos por la plataforma Cosmox de Autenticación y Ranking.
- La **Capa de Interfaces** representa los módulos de las distintas interfaces con las que el usuario tendrá interacción:
    - La interfaz **Principal** donde se manipulan los datos generales y se accede a los distintos módulos de la capa interfaces.
    - La interfaz de **Autenticación** que se encarga de la gestión de los datos de información de usuario.
    - La interfaz de **Ranking** que gestiona todo el proceso de la puntuación recibida de los minijuegos, así como la actualización y visualización de los mismos.
    - La interfaz de **Selección de Minijuego** que se encarga de llevar a cabo el proceso de ejecución directa del minijuego seleccionado por el usuario.
- La **Capa de Minijuegos** contiene todos los videojuegos integrados al sistema.

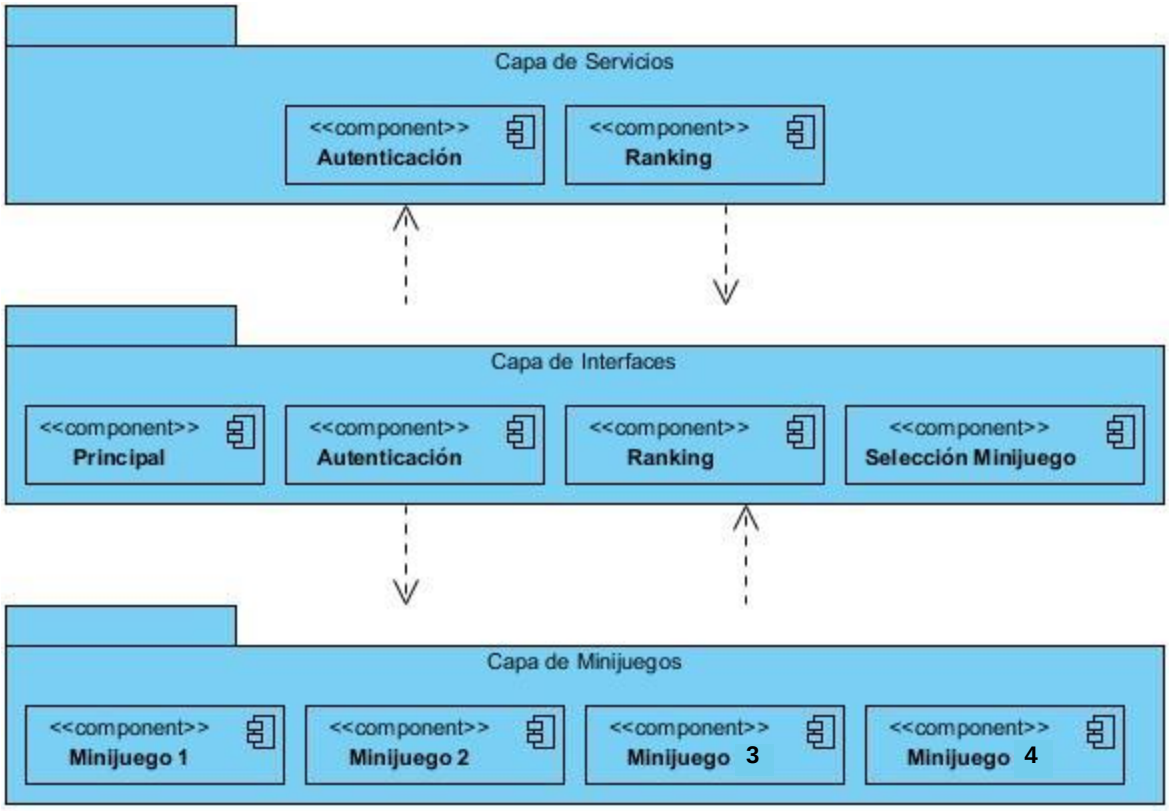

Figura 2 Diagrama de Componentes de la Vista Modular de la Arquitectura de Software.

2.2.3 Vista de Ejecución

La vista de ejecución describe la estructura dinámica del sistema en el momento que se está ejecutando, describe los elementos de su ejecución y modela las tareas operativas del sistema, los procesos que genera, mecanismos de comunicación y asignación de recursos, mostrando el desempeño del sistema en un entorno de ejecución (González).

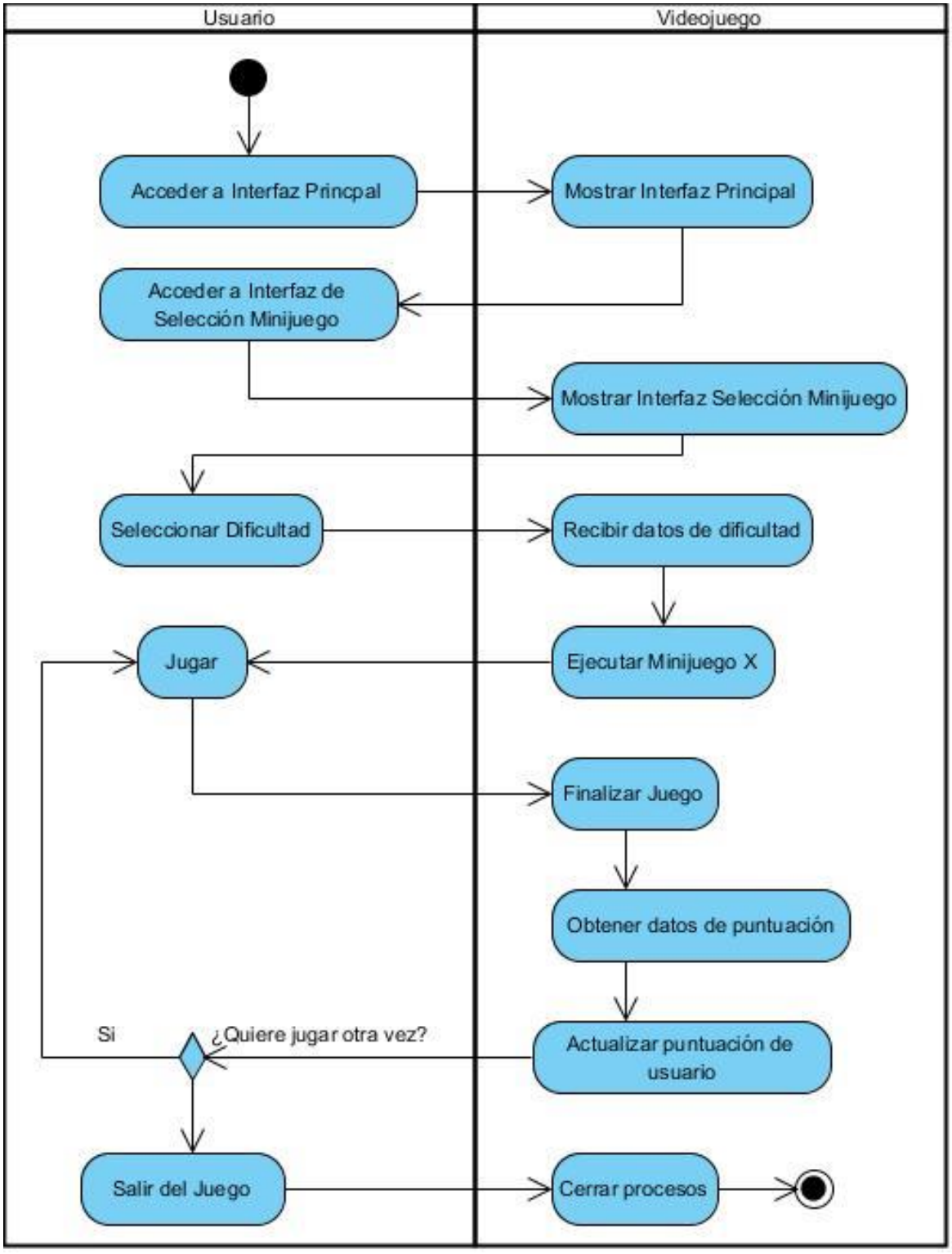

Figura 3 Diagrama de la Vista de Ejecución de la Arquitectura de Software en el Flujo de Ejecución de Minijuego.

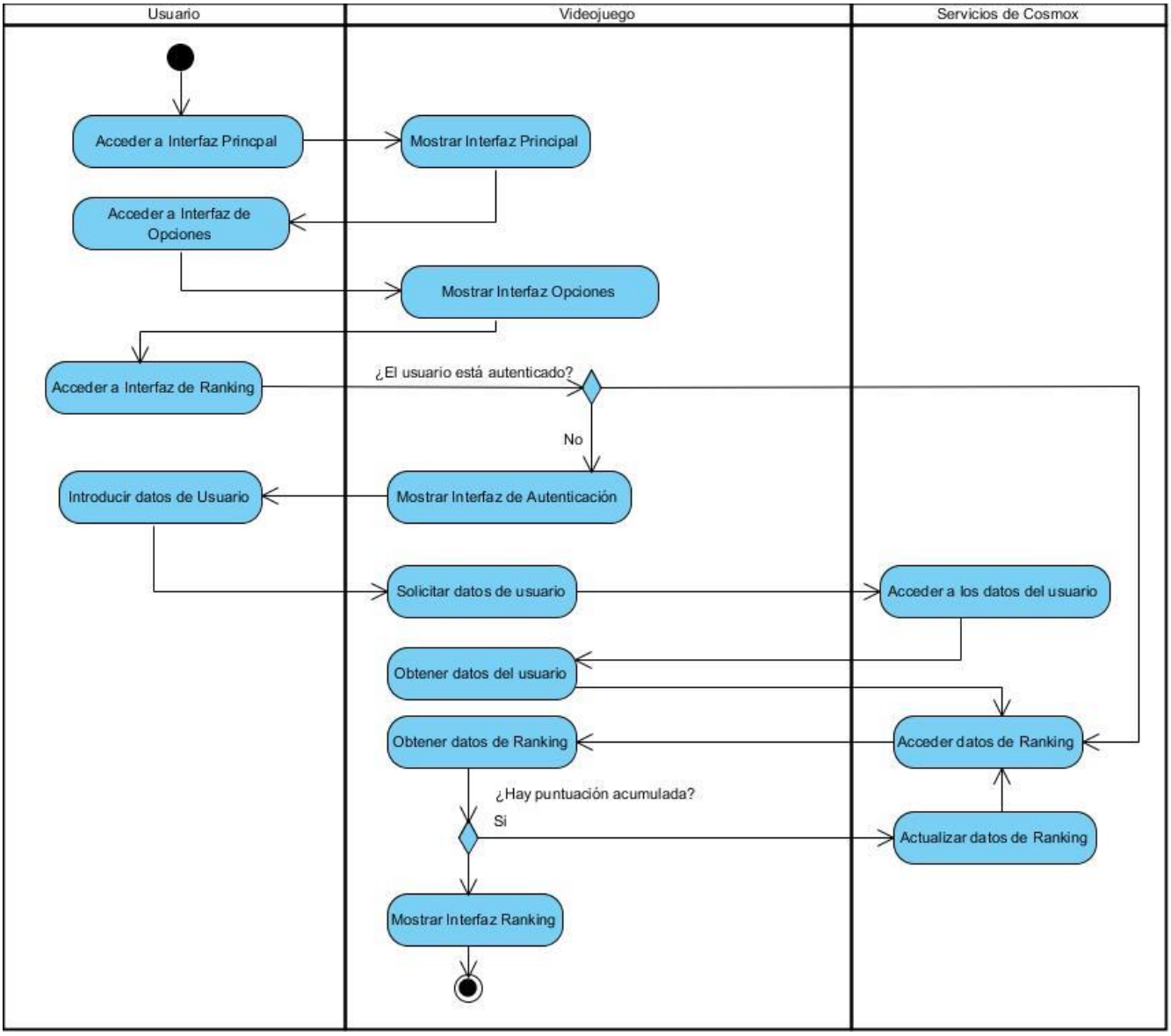

Figura 4 Diagrama de la Vista de Ejecución de la Arquitectura de Software en el Flujo de Ranking y Autenticación.

Existen tres flujos de ejecución principales en la vista de ejecución de la arquitectura de software:

- **Flujo de Ejecución de Minijuego:** El usuario inicia accediendo a la interfaz principal, desde ahí accede a la interfaz de selección de minijuegos, para una vez seleccionado el minijuego envíe los datos de dificultad u otros atributos necesarios para cargar la instancia del minijuego. Al ejecutar el minijuego el usuario podrá jugar su partida que

una vez finalizada emitirá los datos de puntuación que se almacenaran en un archivo momentáneamente y luego regresar a la interfaz principal.

- **Flujo de Autenticación:** El usuario inicia al acceder a la interfaz principal, para luego acceder a la interfaz de autenticación, allí ingresa sus datos para autenticarse y al enviarlo se consumen los servicios de la plataforma Cosmox de autenticación, contrastando si las credenciales ingresadas corresponden a algún usuario y son correctas. Si no son correctas, se direccionará a la interfaz principal, en cambio, si son correctas la autenticación del usuario se mantendrá activa y comenzará el **Flujo de Ranking**.
- **Flujo de Ranking:** El usuario solo puede acceder a este flujo en caso de culminado satisfactoriamente un **Flujo de Autenticación**. Al acceder el usuario a la interfaz de ranking, el sistema comprueba si hay puntuación acumulada por el **Flujo de Ejecución de Minijuego**. Si hay puntuación acumulada se actualiza la puntuación del ranking a través del consumo de los servicios de la plataforma Cosmox de ranking, para luego mostrar los datos actualizados. Si no hay puntuación acumulada, consumen dichos servicios para únicamente mostrar la puntuación en la interfaz de Ranking.

2.2.4 Vista de Código

La vista de código organiza el código fuente, los directorios, archivos y bibliotecas. Algunos de los aspectos que se incluyen son la administración de la configuración, la estructura y organización del proyecto.

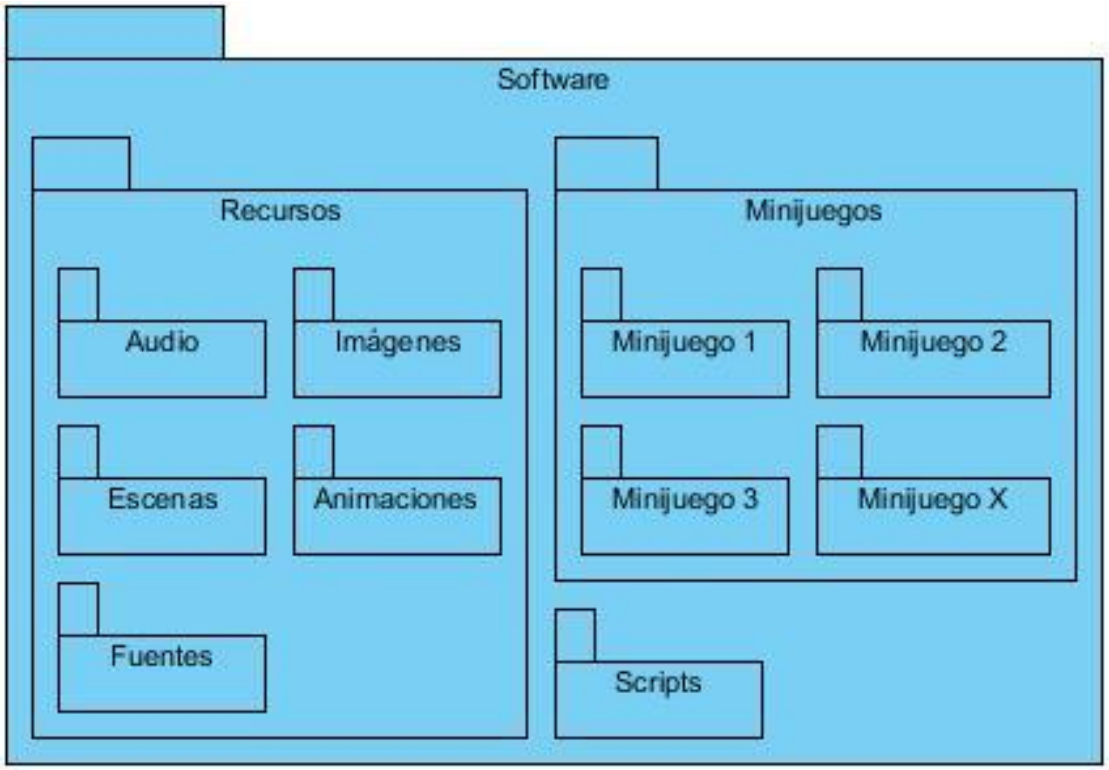

Figura 5 Diagrama de Paquetes de la Vista de Código en la Arquitectura de Software.

El paquete de software es la que contendrá la aplicación final, en su interior contiene una distribución de carpetas con una serie de elementos que establecerán una relación entre ellos:

- El paquete de **Recursos** contiene todos los recursos de multimedia que necesita el software para su funcionamiento:
    - El paquete de **Audio** almacena todos los audios, sonidos y músicas utilizadas en el software.
    - El paquete de **Imágenes** almacena todas las imágenes utilizadas en el software.
    - El paquete de **Escenas** contiene todas las escenas que conforman el software.
    - El paquete de **Animaciones** contiene las animaciones que se utilizan para un mayor dinamismo del software.
    - El paquete de **Fuentes** contiene todas las fuentes tipográficas utilizadas en el software.
- El paquete de **Minijuegos** contiene todos los minijuegos que se anclarán al software y generarán las puntuaciones de los usuarios.

- El paquete de **Scripts** es el que da la funcionalidad al software, contiene todos los archivos de código del sistema. Además, en este paquete es donde se almacenan los dos archivos de código más importantes para la comunicación de los minijuegos con el servicio de Ranking de la plataforma Cosmox:
    - **Archivo de dificultad:** Este archivo es generado en dependencia de la progresión del usuario en el software, definirá la dificultad con la que debe cargarse el juego, su contenido no va más allá de un número entero que definirá la dificultad de la instancia del minijuego cargado, a mayor sea el número, mayor será la dificultad. Su escritura se llevará a cabo en el momento en que el usuario selecciona el minijuego que desea jugar y su lectura será llevada a cabo al iniciar la instancia del minijuego seleccionado. Este archivo es denominado como **dificultad**.
    - **Archivo de puntuación:** Este archivo es generado al completar una partida de un minijuego, almacena la cantidad de puntos acumulados en la partida. Siempre contiene un número natural, que es la cantidad de puntos que hay que pasar desde la capa de los minijuegos a la capa de las interfaces. Su escritura ocurre al finalizar la partida, antes de acabar la instancia del minijuego y al transferir los datos a la puntuación entre las interfaces se realiza un proceso de lectura antes para conocer el dato y luego se restablece el valor del archivo a 0. Que su contenido sea 0 indica que no hay puntos que traspasar de la capa de minijuegos a la capa interfaces. Tiene otra instancia de lectura y/o escritura en el momento de la autenticación correcta del usuario, ahí se comprobará el valor y se transferirán los datos si su valor es distinto a 0, ya que un usuario puede jugar los minijuegos y acumular puntuación en este archivo sí que se encuentre autenticado. Este archivo es denominado como **puntos**.

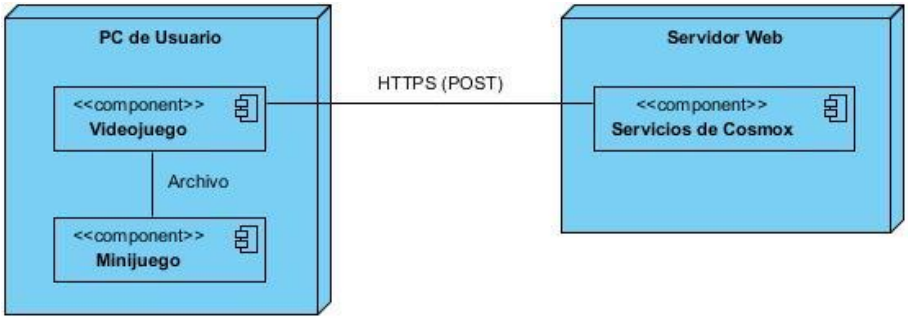

Figura 6 Diagrama de Despliegue de la Arquitectura

El videojuego, contenido en el pc del usuario se comunica con los distintos a través del sistema de archivos de la plataforma mediante la distribución descrita en el paquete de scripts. La comunicación con los servicios de Cosmox será a través del servidor web en el que se encuentra mediante conexión segura HTTPS con el envío de información a través del método POST.

Conclusiones del capítulo

La arquitectura diseñada en este capítulo está caracterizada por la integración de los videojuegos y los servicios que ofrece la plataforma Cosmox de Ranking y Autenticación como componentes que se integran como un todo. Se definen cuatro vistas que estructuran la arquitectura, la definen conceptualmente, mediante módulos que se encuentran integrados en tres capas distintas, la forma en la que se ejecutan los distintos procesos de software y la estructuración de los archivos de código.

CAPÍTULO 3: VALIDACIÓN DE LA PROPUESTA DE ARQUITECTURA DE SOFTWARE PARA VIDEOJUEGOS Y SERVICIOS DE LA PLATAFORMA COSMOX

En este capítulo se exponen algunos conceptos asociados a cómo se evalúa una arquitectura de software, los métodos de cómo evaluar la arquitectura y se aplican las técnicas de pruebas a un prototipo de software desarrollado sobre esa arquitectura.

3.1 Evaluación de la arquitectura de software

La arquitectura de software tiene un gran impacto en la calidad de un sistema, por lo que se hace necesario evaluarla para determinar su potencial para alcanzar los atributos de calidad requeridos. Aunque es importante destacar que la evaluación no define si una arquitectura es buena o no, simplemente expresa donde se encuentran los riesgos y fortalezas de la misma (Cruz, 2017).

En principio, se deben establecer los parámetros que se quieren evaluar, de esta forma es posible establecer la base para la evaluación, así como establecer cuándo se debe establecer la evaluación. Para ello, es posible evaluar la arquitectura en cualquier fase del desarrollo, existiendo dos variantes principales que definen cuándo realizar la evaluación: evaluación temprana y evaluación tardía (Clements, Kazman, & Klein, 2002).

Para realizar la evaluación temprana no es necesario que la arquitectura esté totalmente especificada, esta puede realizarse desde las fases tempranas del diseño y a lo largo del proceso de desarrollo, lo que permite tomar decisiones arquitectónicas producto a una evaluación en función de los atributos de calidad esperados. La evaluación tardía se realiza cuando la arquitectura ya está establecida y la implementación se ha culminado, realizar la evaluación en este momento se considera muy útil, pues se puede observar el cumplimiento de los atributos de calidad asociados al sistema y su comportamiento de forma general (Clements, Kazman, & Klein, 2002).

3.2 Técnicas de evaluación de arquitectura de software

Las técnicas utilizadas para la evaluación de atributos de calidad requieren grandes esfuerzos por parte del ingeniero de software para crear especificaciones y predicciones. Estas técnicas requieren información del sistema a desarrollar que no está disponible durante el diseño arquitectónico, sino al principio del diseño detallado del sistema (Software engineering, 2001).

En vista de que el interés es tomar decisiones de tipo arquitectónico en las fases tempranas del desarrollo, son necesarias técnicas que requieran poca información detallada y pueden conducir a resultados relativamente precisos. Las técnicas existentes en la actualidad para evaluar arquitecturas permiten hacer una evaluación cuantitativa sobre los atributos de calidad a nivel arquitectónico, pero se tienen pocos medios para predecir el máximo (o mínimo) teórico para las arquitecturas de software. Sin embargo, debido al costo de realizar este tipo de evaluación, en muchos casos los arquitectos de software evalúan cualitativamente para decidir entre las alternativas de diseño (Bosh, 2000).

3.2.1 Evaluación basada en escenarios

Un escenario es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema con este. Consta de tres partes: el estímulo, el contexto y la respuesta. El estímulo es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema. Esto puede incluir la ejecución de tareas, cambios en el sistema, ejecución de pruebas, entre otros. El contexto describe qué sucede en el sistema al momento del estímulo. La respuesta describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Este último elemento es el que permite establecer cuál es el atributo de calidad asociado (Clements, Kazman, & Klein, 2002).

3.2.2 Evaluación basada en experiencia

En muchas ocasiones los arquitectos e ingenieros de software otorgan valiosas ideas que resultan de utilidad para la evasión de decisiones erradas de diseño, estas ideas se basan en factores subjetivos (Bosh, 2000) y están respaldadas por una línea lógica de razonamiento, que se puede adquirir por el trabajo realizado en proyectos similares. Por tanto, el principal instrumento de evaluación con que cuenta esta técnica es precisamente la intuición y experiencia con que cuentan los arquitectos y demás miembros del equipo de desarrollo. Existen dos tipos de evaluación basada en experiencia: la evaluación informal, que es realizada por los arquitectos de software durante el proceso de diseño, y la realizada por equipos externos de evaluación de arquitecturas.

3.2.3 Evaluación basada en prototipo

Esta técnica consiste en implementar una parte de la arquitectura de software y ejecutarla en el contexto del sistema. Para su uso se necesita mayor información sobre el desarrollo y disponibilidad de hardware, y los elementos que constituyen el contexto del sistema de software. Mediante esta técnica se obtiene un resultado de evaluación con mayor exactitud (Bosh, 2000).

El prototipo de un videojuego es un método para probar un concepto del juego, en este caso su arquitectura, comprobar si la idea puede llevarse a la práctica antes de invertir muchos recursos en él. También se utiliza para perfeccionar ciertos mecanismos que pudiesen resultar hacerlo más interesante y atractivo a los jugadores (Pérez, 2015).

Al iniciarse un proyecto de videojuego, con los prototipos se pueden conocer lo antes posible diversos problemas de interfaz o mecanismos que puedan solucionarse fácilmente. La creación de prototipos de juegos debe utilizarse como un generador de ideas y una exploración alternativa que permita ganar más valor en el proceso de investigación, definición, concepción y prueba; explorando y experimentando ideas, problemas y oportunidades. El prototipo del videojuego además genera confianza a los usuarios y los inversores, ya que les permite tener el acceso más directo a las ideas del proyecto.

El prototipo del videojuego que se desarrolla necesita integrar los distintos minijuegos y los servicios de Ranking y Autenticación de la plataforma Cosmox mediante una arquitectura de software que permita integrarlos a través de una interfaz principal. En el desarrollo de los prototipos también se utilizan las herramientas de software y metodologías de desarrollo correspondientes que pudiesen derivar a las versiones estables.

3.3 Métodos de evaluación de arquitectura de software

Independiente a las técnicas de la evaluación de las arquitecturas, existen también métodos de evaluación arquitectónica, evalúan el potencial del diseño arquitectónico para alcanzar los niveles deseados en cuanto a los requisitos de calidad (Clements, Kazman, & Klein, 2002)., estos métodos se ven asistidos de las técnicas de evaluación arquitectónica analizadas anteriormente. Actualmente existen diversos métodos para realizar pruebas a la arquitectura de software, cada uno con características específicas, escoger un método de evaluación

requiere tener bien definidos los atributos que se desean evaluar. Algunos de los métodos de evaluación son:

- El **Método de Análisis de Arquitectura de Software (SAAM)** como uno de los primeros métodos en ser ampliamente difundidos y documentados. Creado originalmente para el análisis de la modificabilidad de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar de forma rápida atributos de calidad, tales como la portabilidad, escalabilidad e integralidad (Clements, Kazman, & Klein, 2002).
- El **Método de Revisión Intermedio de Diseño (ARID)** es conveniente de realizar para la evaluación de diseños parciales en las etapas tempranas del desarrollo. Gira en torno a la calidad y completitud de la documentación y la suficiencia, ajuste y conveniencia de los servicios que provee el diseño propuesto (Clements, Kazman, & Klein, 2002).
- El **Método de Análisis de Acuerdos de Arquitectura de Software (ATAM)** está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación de SAAM integralidad (Clements, Kazman, & Klein, 2002). El método se concentra en la identificación de los estilos arquitectónicos o enfoques arquitectónicos utilizados y ayuda a los involucrados en el proyecto a entender las consecuencias de las decisiones arquitectónicas tomadas con respecto a los atributos de calidad. Su desarrollo se basa nueve pasos agrupados en cuatro fases.

3.4 Evaluación de la arquitectura propuesta

Después del análisis de las diferentes técnicas y métodos de evaluación de arquitecturas, se seleccionan, para evaluar la arquitectura propuesta, la técnica de evaluación basada en prototipo y en escenarios en conjunto con el método de evaluación ATAM.

3.5 Metodología, Herramientas y Tecnologías para el desarrollo del prototipo

Para poner en práctica lo que se el desarrollo del prototipo del videojuego como solución al problema que aborda esta investigación, se pretenden utilizar las siguientes herramientas.

3.5.1 Metodología del desarrollo de software

Para el desarrollo de este prototipo funcional se utiliza la metodología para el desarrollo de software ágil Extreme Programming (XP), que desarrolla y gestiona proyectos con eficiencia,

flexibilidad y control basada en la retroalimentación. Esta metodología está basada en 5 fases (Pressman, 2002):

1. **Planificación:** va de acuerdo con las historias de usuario, se priorizan y descomponen en mini versiones, luego la planificación se va revisando cada dos semanas aproximadamente, después de las iteraciones, para obtener un software útil, funcional, listo para las pruebas y su lanzamiento.
2. **Diseño:** se trabaja en un código sencillo, realizando lo mínimo necesario para que funcione, obteniendo un prototipo.
3. **Codificación:** se realiza la codificación del software de forma organizada y planificada.
4. **Pruebas:** son automáticas y continuas, son la clave para proyectos a corto plazo. Incluso el cliente puede hacer pruebas, proponer pruebas nuevas y validarlas.
5. **Lanzamiento:** luego de los éxitos de la fase de pruebas ajustadas a los requisitos del cliente se genera el software útil para incorporarlo al mercado.

3.5.2 Lenguaje Unificado de Modelado (UML)

Es el lenguaje estándar para escribir diseños de software, puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo. Está pensado para utilizarse en todos los métodos de desarrollo, etapas del ciclo de vida de un software, dominios de aplicación y medios. Es un sistema de notaciones destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Es el estándar mundial que utilizan los desarrolladores, autores y proveedores de Herramientas para Ingeniería de Software Asistida por Computación (CASE) (Pressman, 2002).

3.5.3 Herramienta CASE: Visual Paradigm

Es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, soporta el ciclo de vida completo del proceso de desarrollo de software a través de la representación de todo tipo de diagramas, diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los

requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software (Visual Paradigm, 2022).

3.5.4 Motor de Videojuegos: Unity 3D

Es un motor gráfico 3D empaquetado como una herramienta para crear videojuegos, aplicaciones interactivas, visualizaciones y animaciones 3D. Es el centro de la línea de producción, con un completo editor visual para crear videojuegos. El contenido del juego es construido desde el editor y el gameplay[6] se programa usando un lenguaje archivos de código (Unity 3D, 2023). Además, varios de los desarrolladores del Centro VERTEX tienen un vasto conocimiento en este motor y los minijuegos que se pretenden integrar han sido desarrollados en él.

3.5.5 Lenguaje de Programación: C#

C# (en su versión 7.0) es uno de los lenguajes de programación que emplea Unity 3D para compilar los archivos de código, es orientado a objetos, lo cual facilita el trabajo ya que en esta herramienta todo componente o elemento del videojuego es un objeto o clase. Al empezar a programar, se pueden definir una o más clases dentro de un mismo espacio de nombres. Soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo (Seco, 2001).

3.5.6 Entorno de Desarrollo Integrado (IDE): Visual Studio 2022

IDE de Visual Studio es una plataforma de lanzamiento creativa que puede utilizar para editar, depurar y compilar código y, finalmente, publicar una aplicación. Además del editor y depurador estándar que ofrecen la mayoría de IDE, Visual Studio incluye compiladores, herramientas de completado de código, diseñadores gráficos y muchas más funciones para mejorar el proceso de desarrollo de software. Visual Studio es el IDE más rápido para la productividad. Tenga como destino cualquier plataforma o dispositivo. Compile cualquier tipo de aplicación. Trabaje en equipo. Diagnostique y detenga problemas antes de que ocurran. Esto hace que sus procesos diarios sean más flexibles y adaptables (Microsoft, 2003).

---

[6] El término gameplay hace referencia a la manera en la que el jugador interactúa con el juego o a la manera en la que el juego interactúa con el jugador.

3.6 Desarrollo del prototipo de videojuego

El objetivo del prototipo del videojuego es validar el funcionamiento de la arquitectura de software diseñada en el capítulo anterior. Su estilo artístico enaltecerá la cultura cubana, principalmente el género musical del son, sus figuras principales y algunos de los elementos asociados al baile, la danza, la gastronomía y la música. Se quiere dar aporte al compromiso que tiene la UCI con los ministerios de educación superior y cultura para el producto que promocione la cultura del son, así como las Casas Llamo al Son, y sus productos asociados, con la implementación de los minijuegos.

La modalidad de juego es mayormente individual, con algunos de sus minijuegos cooperativos como el minijuego de trivia “Sabios del Son”. El juego tiene un alcance para jugadores de 8 años en adelante.

3.6.1 Diseño del prototipo de videojuego

El juego cuenta con una estética toon, tanto para personajes y fondos. Se emplea una paleta de colores llamativos semejantes a los que se aprecian en el paisaje campestre cubano. Las ventanas y botones tienen bordes redondeados. Los elementos del menú deben tener un aspecto vintage (estética de los años 1930 a 1950).

La fuente utilizada para el videojuego es: Made Mountain (MadeType, 2023).

El videojuego iniciará con un menú principal, desde el cual se puede acceder a la pantalla del mapa de cuba. Dicha pantalla mostrará las distintas secciones (provincias) del juego. El mapa de cuba será verde contrastando con el azul del mar, la paleta de colores será viva del trópico. El HUD[7] tendrá un aspecto de madera barnizada vetada como algunos de los instrumentos de cuerda que se ven en el mismo juego. Cada una de las provincias tendrá una respectiva pantalla de Casa Llamo al Son y cada casa una Biblioteca Musical, que contiene breves descripciones de figuras históricas, instrumentos y elementos asociados.

Las reglas de un videojuego son las que definen qué el usuario puede hacer y que no, en este caso:

[7] En el ámbito de los videojuegos, se le llama HUD (Heads-UP-Display) al conjunto de iconos, números, mapas, etc. Que durante el juego provee información sobre el estado de la partida y/o personaje, como por ejemplo vida restante, ubicación, munición, objetos en uso, etc (DeVuego, 2023).

- El jugador supera el juego cuando desbloquea todas las casas “Llamo al Son”.
- El jugador supera la fase de cada provincia desbloqueando todos los elementos asociados en la “Casa Llamo al Son”.

3.6.2 Distribución de los niveles

El juego cuenta con 70 niveles en total, cada provincia (15 en total) cuenta con 4 minijuegos para cada una, variando la dificultad progresivamente. Cada provincia además cuenta con una leyenda del son cubano que guía al jugador por su recorrido.

Provincia	**Dificultad**	**Leyenda**
Guantánamo	Fácil	Elio Revé
Santiago de Cuba	Fácil	Compay Segundo y Cándido Fabre
Holguín	Fácil	Guayabero
Granma	Medio	Sindo Garay, Pachi Naranjo
Las Tunas	Medio	Familia Valera Miranda, Barbarito Diez
Camagüey	Medio	Adalberto Álbarez, Tiburon Morales
Ciego de Ávila	Medio	Bando Rojo y Azul
Sancti Spíritus	Medio	<<sin definir>>
Cienfuegos	Difícil	Benny More
Villa Clara	Difícil	<<sin definir>>
Matanzas	Difícil	Arsenio Rodrígue y Pérez Prado
Mayabeque	Difícil	A lo mejor no
Pinar del Río	Difícil	Polo Montañés
La Habana	Difícil	Ignacio Piñeiro, Juan Formel y Miguelito Valdés
Isla de la Juventud		Mongo Rives

Tabla 1 Representación de los niveles del videojuego

Diagramas de Estados

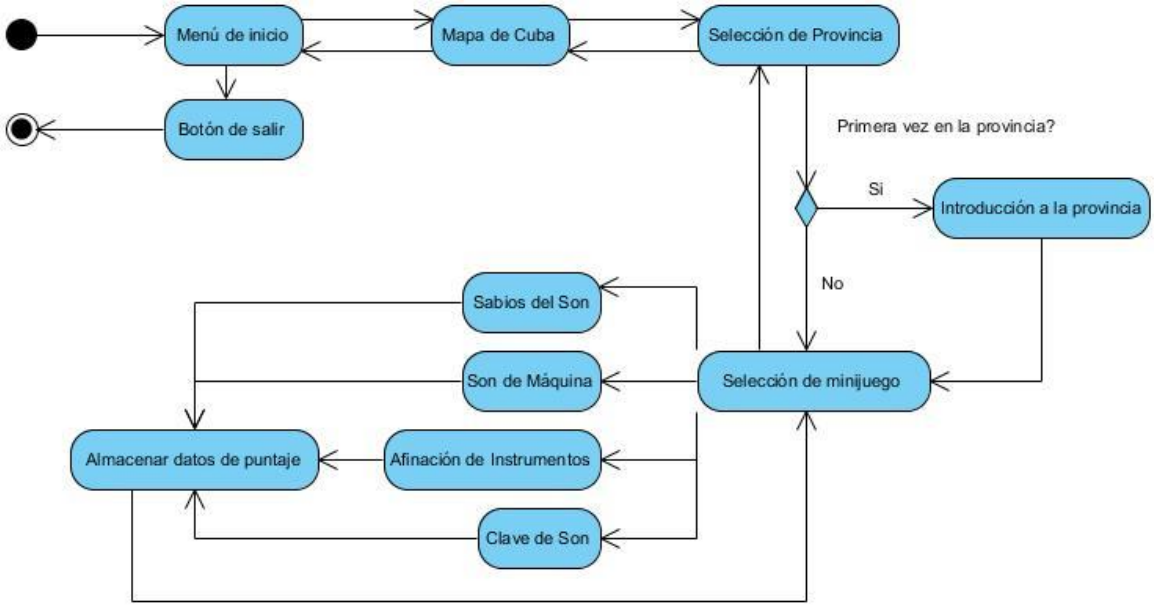

Figura 7 Diagrama de Estados de la "Selección de Minijuegos".

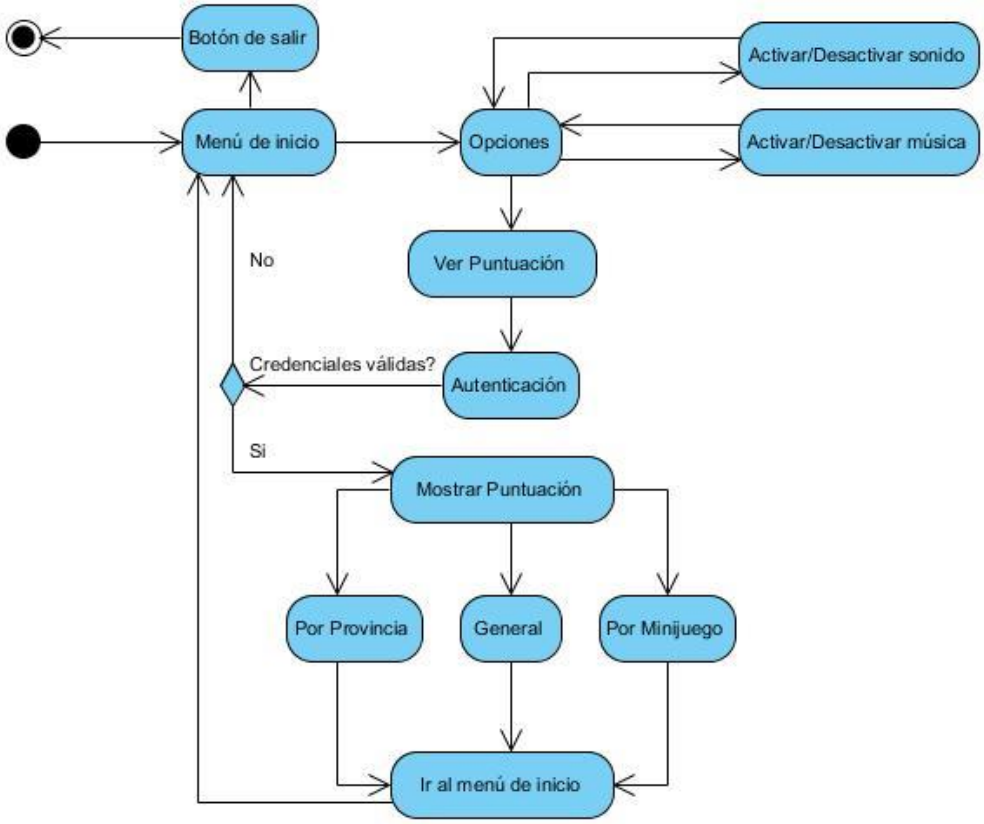

Figura 8 Diagrama de Estado de “Mostrar Puntuación”.

Diagrama de Clases

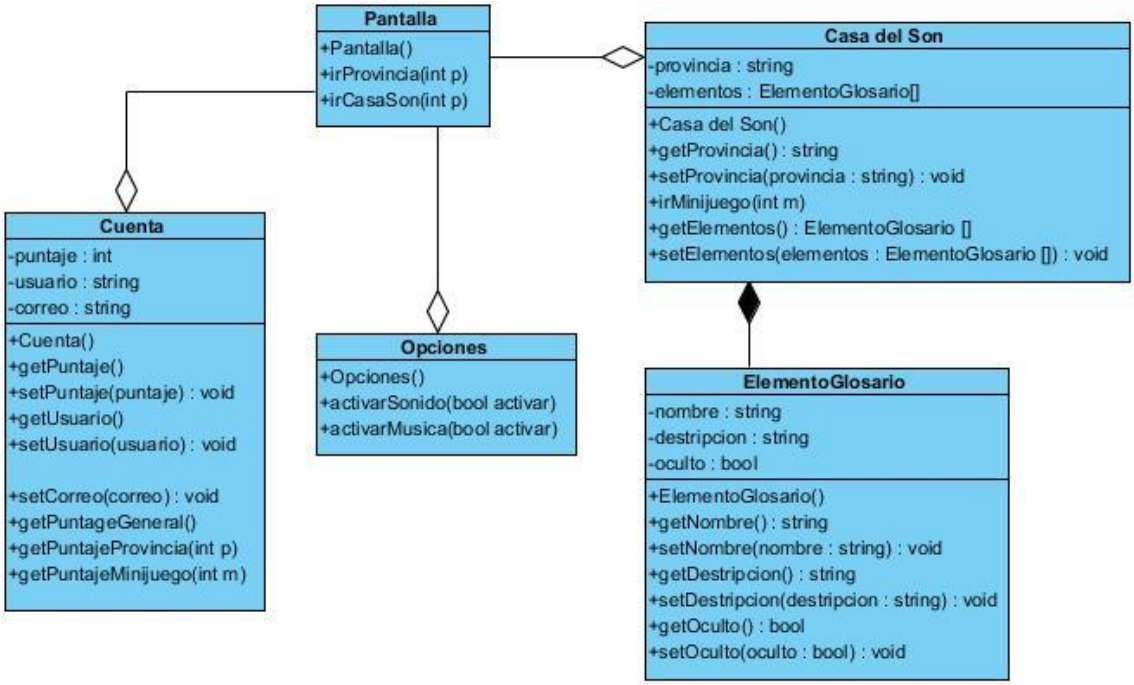

Figura 9 Diagrama de Clases.

3.6.3 Especificación de Eventos

La especificación de los eventos tiene como objetivo el identificar los eventos que conforman el videojuego, sus propiedades y organización arquitectónicas. Es la sección que describe esencialmente lo que el usuario puede hacer y cómo puede hacerlo.

Descripción Textual de Eventos

Objetivo	Cargar al minijuego	
Actores	Jugador	
Resumen	Cargar el minijuego	
Complejidad	Baja	
Prioridad	Alta	
Precondiciones	-	
Postcondiciones	-	
Flujo de eventos		
Actor		**Sistema**
	Accede a la interfaz “Mapa de Cuba”	
		Muestra la interfaz “Mapa de Cuba”
	Selecciona la dificultad del Minijuego, definido como la provincia del país en el mapa.	
		Muestra el contenido de la provincia seleccionada
	Selecciona el Minijuego a jugar.	
		Cargar el minijuego
Prototipo elemental de interfaz gráfica de usuario		
		

Tabla 2 Descripción textual del evento Selección de Minijuegos.

Objetivo	Autenticar usuario	
Actores	Jugador	
Resumen	Cargar el minijuego	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario debe tener una cuenta en la plataforma Cosmox.	
Postcondiciones	-	
Flujo de eventos		
Actor		**Sistema**
	Accede a la interfaz “Opciones”	
		Muestra la interfaz “Opciones"
	Accede a la interfaz “Ranking”	
		Muestra la interfaz “Autenticar a Ranking”
	Introducir datos	
		Cargar los datos de usuario de la plataforma Cosmox
Prototipo elemental de interfaz gráfica de usuario		

Tabla 3 Descripción textual del evento Autenticar usuario.

Objetivo	Ver Ranking	
Actores	Jugador	
Resumen	Ver el Ranking	
Complejidad	Baja	
Prioridad	Media	
Precondiciones	El usuario debe estar autenticado.	
Postcondiciones	-	
Flujo de eventos		
Actor		**Sistema**
	Accede a la interfaz “Ver Ranking”	
		Cargar los datos de ranking de la plataforma Cosmox
Prototipo elemental de interfaz gráfica de usuario		

Tabla 4 Descripción textual del evento Ver Ranking.

Objetivo	Ver Glosario de Términos	
Actores	Jugador	
Resumen	Ver el Glosario de Términos	
Complejidad	Media	
Prioridad	Baja	
Precondiciones	-	
Postcondiciones	-	
Flujo de eventos		
Actor		**Sistema**
	Accede a la interfaz “Ver Glosario de Términos”	
		Muestra los datos del Glosario de Términos
Prototipo elemental de interfaz gráfica de usuario		

Tabla 5 Descripción textual del evento Ver Glosario de Términos.

3.7 Árbol de utilidad

El árbol de utilidad presenta la evaluación mediante el método de ATAM en conjunto con las técnicas basadas en escenarios y prototipos, son los más relevantes a comprobar sobre el dominio de aplicación desarrollando videojuegos, dado que aporta las interpretaciones consistentes, pertinentes, facilitando el análisis de su comportamiento desde el núcleo arquitectónico hasta los horizontes de la arquitectura relacionados con las funcionalidades más básicas del prototipo funcional.

Los atributos que puede manejar un árbol de utilidades para la arquitectura de software de un videojuego dependen de los objetivos y las características del mismo, así como de los requisitos no funcionales que se quieren lograr con la arquitectura. En este caso se medirán los siguientes atributos para determinar las fortalezas y debilidades de la arquitectura:

- **Disponibilidad:** se refiere a la capacidad del videojuego de estar operativo y accesible cuando se requiere, ya que se trata de un juego en línea o multijugador.
- **Rendimiento:** se refiere a la velocidad y la eficiencia con la que el videojuego realiza sus funciones, ofreciendo una experiencia fluida y satisfactoria al usuario. Se puede medir por el tiempo de respuesta, el uso de recursos y la capacidad de procesamiento.
- **Seguridad:** se refiere a la protección del videojuego frente a amenazas externas o internas que puedan comprometer su integridad, confidencialidad o disponibilidad. Se puede medir por el nivel de riesgo, la frecuencia y la severidad de los ataques, y las medidas de prevención, detección y recuperación.
- **Escalabilidad:** se refiere a la capacidad del videojuego de soportar un aumento en el número o en las demandas de los usuarios, sin perder calidad o rendimiento. Se puede medir por el número máximo de usuarios concurrentes, el tiempo máximo de espera o el uso máximo de recursos.
- **Interoperabilidad:** se refiere a la capacidad del videojuego de comunicarse e intercambiar información con otros sistemas o plataformas, manteniendo su funcionalidad y calidad. Se puede medir por el número, el tipo y el formato de los sistemas o plataformas con los que se puede interactuar.
- **Portabilidad:** se refiere a la capacidad del videojuego de funcionar en diferentes entornos o dispositivos, sin requerir modificaciones o adaptaciones. Se puede medir por

el número, el tipo y las características de los entornos o dispositivos en los que se puede ejecutar.

3.8 Resultados de la validación de la propuesta arquitectónica

Luego de realizar las distintas pruebas de validación, siguiendo los atributos expuestos en el árbol de utilidades a la propuesta arquitectónica se han obtenido los siguientes resultados:

3.8.1 Disponibilidad

Las pruebas de disponibilidad han determinado que, el software está estrechamente vinculado a la plataforma Cosmox, y la disponibilidad de los servicios relacionados con ella es total dependencia de la misma, sin embargo, no todas las funcionalidades se ven obstruidas, ya que gracias a la arquitectura, los juegos siempre van a estar disponibles, es decir, serán en todo momento jugables para el usuario y sus estadísticas de puntuación no se verán perjudicadas, ya que aunque no se encuentre autenticado en el sistema, el software va acumulando la puntuación hasta que el usuario se autentique a la plataforma y se puedan consumir los servicios de Ranking.

3.8.2 Rendimiento

Cada minijuego que se integre a la aplicación, lleva sus propias pruebas de rendimiento, las pruebas de rendimiento se han aplicado al prototipo funcional base donde se integró la arquitectura. Una de las herramientas para las pruebas realizadas orientadas al rendimiento fue el Monitor de Recursos que ofrece el Administrador de Tareas de Windows 11, este software integrado al sistema operativo permite conocer datos específicos de las distintas tareas que se encuentran funcionando en el sistema operativo.

Una vez iniciada la aplicación, consume la cifra de 141 Mb de memoria en el sistema, esta cifra aumenta 3 Mb más una vez el usuario se ha autenticado en la aplicación. La aplicación en su estado de ejecución consta con 53 subprocesos y un porcentaje de un 5% de uso del proceso.

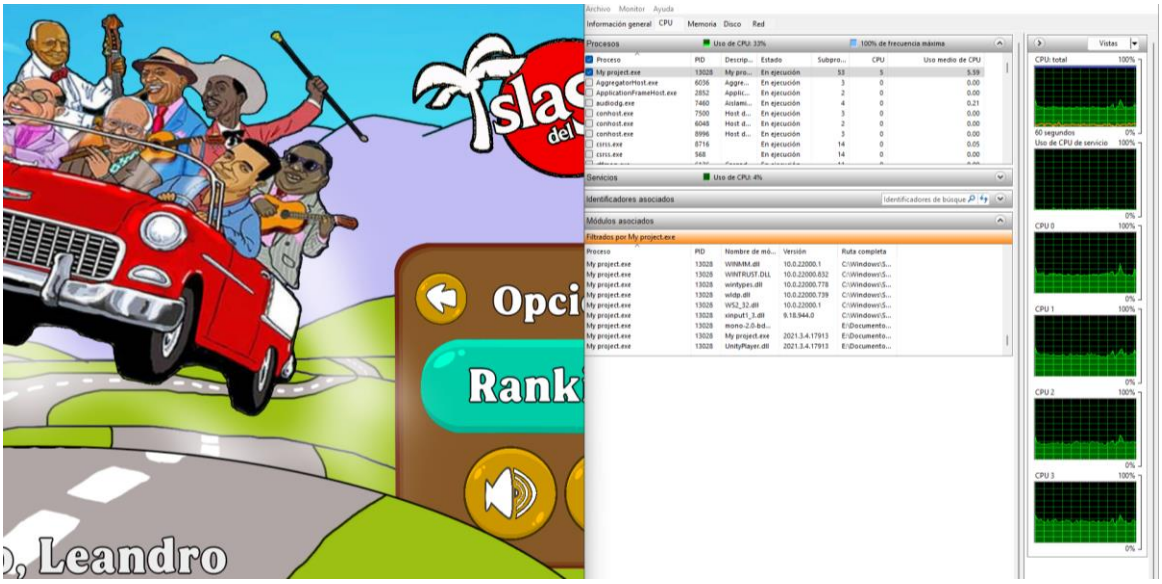

Figura 10 Datos del Monitor de Recursos durante la ejecución del prototipo.

El tiempo de respuesta al cargar los distintos escenarios, que en este caso son los minijuegos anclados oscila entre los 0,01 y 0,10 segundos. Este tiempo de respuesta aumenta en dependencia de la complejidad del minijuego que se esté cargando, podría determinarse que, por parte de la aplicación base que se encarga de controlar la arquitectura, el tiempo de respuesta entre escenarios es despreciable. Se obtienen iguales resultados al interactuar con el módulo de los servicios, donde el tiempo de respuesta oscila entre los 0,01 y 0,10 segundos sumados al tiempo de respuesta que ofrecen los servicios de la plataforma Cosmox.

3.8.3 Seguridad

Este videojuego tiene dos factores vulnerables a través de los cuales se pueden alterar los datos en su tráfico. Una de las vías vulnerables es al enviar y recibir los datos que consumen los servicios de la plataforma Cosmox, estos datos son protegidos mediante el envío a través de la vía web por método POST, método que envía parámetros en la solicitud HTTP para el servidor.

La otra vía vulnerable es la transferencia de datos al incursionar en el módulo de los minijuegos, donde se envían y reciben los archivos de datos de dificultad y puntuación respectivamente. Estos archivos al generarse utilizan la librería System.Security.Criptography que proporciona Microsoft, que proporciona servicios criptográficos, incluida la codificación y

decodificación seguro de datos, así como muchas otras operaciones, como hashing, generación de números aleatorios y autenticación de mensajes (Microsoft Ignite, 2023).

3.8.4 Escalabilidad

El prototipo funcional de esta arquitectura incluye 4 minijuegos que han sido integrados para ser jugados por el usuario, si bien el prototipo no ofrece una buena escalabilidad a menos que reciba un rediseño gráfico, la arquitectura sí lo es. No existe un límite en la escalabilidad que afecte el rendimiento de la arquitectura. El límite de la integración de componentes radica en la capacidad de almacenamiento de la que disponga el usuario en el dispositivo tecnológico en el cuál funciona el software donde se ha implementado la arquitectura resultante de esta investigación.

3.8.5 Interoperabilidad

La interoperabilidad de la arquitectura se tiene en cuenta con el acceso a dos de las capas de la vista modular, la capa de los servicios y la capa de los minijuegos, para integrar más componentes de este tipo.

Si la arquitectura necesitase realizar la integración de otros tipos de servicios, al ser una arquitectura basada en componentes, basta con añadir el consumo de dichos servicios al módulo de los servicios y realizar la implementación correspondiente en el código fuente de la aplicación para la operabilidad de este servicio.

En cuanto a la integración de otros minijuegos, cualquier minijuego puede ser integrado a la arquitectura, siempre y cuando cumpla con la gestión de archivos de los datos necesarios del paquete de Scripts, que se muestra en el diagrama de paquetes de la Vista de Código. En síntesis, si un minijuego pretende integrarse a la arquitectura del prototipo basta con que obtenga los datos de dificultad al iniciar y genere los datos de la puntuación al finalizar en los archivos descritos en el paquete de Scripts del epígrafe 2.2.4 Vista de Código de esta investigación.

3.8.6 Portabilidad

A pesar de que el prototipo de la arquitectura fue desarrollado para el sistema operativo Windows de ordenador personal. Pueden explorarse otras plataformas, sin riesgo a que la arquitectura falle, ya que cualquier plataforma que tenga un sistema operativo que permita la

gestión de archivos podrá adaptarse a la arquitectura fácilmente. Una característica a tener en cuenta para mantener los niveles de seguridad de la arquitectura es que en el prototipo desarrollado se utilizan librerías del propio sistema operativo Windows para realizar el cifrado de archivos, por lo que si se utiliza alguna otra plataforma que no sea compatible con estas librerías es necesario investigar otras alternativas.

3.9 Riesgos y Fortalezas de la Arquitectura propuesta

De acuerdo a los resultados obtenidos de las pruebas realizadas al prototipo funcional en el que se integran la arquitectura desarrollada en esta investigación, se llega a la conclusión de los siguientes riesgos y fortalezas:

Fortalezas:

- La disponibilidad del software bajo esta arquitectura es permanente. Los servicios que se consumen en la red son utilizados una vez que el usuario sea conectado a la red y el proveedor de los servicios esté activo, pero el objetivo de acceder a los módulos de los distintos minijuegos siempre está disponible.
- El prototipo desarrollado tiene muy bajo coste de recursos, por lo que es altamente probable que el resto de aplicaciones desarrolladas bajo esta arquitectura tengan un bajo consumo de recursos independientemente de la tecnología utilizada en su desarrollo.
- La arquitectura permite la integración de componentes que ofrezcan servicios como los de Autenticación y Ranking que ofrece la plataforma Coxmox consumidos por el prototipo de prueba.
- Cualquier minijuego puede ser integrado mientras cumpla con la consumo y generación de archivos definida en la Vista de Código del epígrafe 2.2.4.

Riesgos:

- Un software desarrollado bajo esta arquitectura debe ser compatible con una plataforma que integre un sistema de gestión de archivos y a su vez tenga métodos para el cifrado y protección del contenido de los mismos. A menos que se vincule una tecnología externa para el cifrado del contenido de los archivos.
- El límite de minijuegos que se puede integrar a la arquitectura es en dependencia del objetivo que se quiere lograr, la operabilidad de la arquitectura no se ve afectada, sin

embargo, mientras más componentes se integren mayor almacenamiento necesitará un usuario para utilizar el software desarrollado.

Conclusiones del capítulo

Realizadas las pruebas para comprobar la eficacia y eficiencia de la arquitectura diseñada en el capítulo 2, se concluye que cumple con el objetivo de la investigación del diseño de una arquitectura de software que permite la integración de videojuegos y consume los servicios de la plataforma Cosmos. Se logra definir las distintas fortalezas y debilidades para que los desarrolladores de software que la utilicen comprendan las vulnerabilidades de su software.

CONCLUSIONES

Con la realización de esta investigación, se definió y se validó una arquitectura de software para la integración de videojuegos y los servicios de la plataforma Cosmox, probada sobre un prototipo funcional, que permite conocer sus fortalezas y debilidades para los desarrolladores que decidan integrarla. De esta forma se dio cumplimiento al objetivo propuesto al inicio de la investigación, además:

- Se determinó qué vistas arquitectónicas son fundamentales para el desarrollo de una arquitectura que se centre en la integración de videojuegos y servicios, principalmente los servicios que ofrece la plataforma Coxmox, como componentes.
- Se diseñó una arquitectura basada en componentes, ya que las características de estilo de este patrón arquitectónico definieron que cada uno de los elementos a integrar (los minijuegos y los servicios de la plataforma Cosmox) podían ser tratados como componentes.
- Se definió como integrar los servicios de Ranking y Autenticación que ofrece la plataforma Cosmox en una arquitectura orientada a la integración de componentes y que ventajas ofrece la utilización de este servicio a la arquitectura.
- Las pruebas realizadas, mediante el método ATAM y basadas en escenarios y prototipo, arrojaron la presencia de las fortalezas y riesgos en el prototipo, que fueron registrados.

RECOMENDACIONES

La arquitectura diseñada como resultado de la investigación es extensible para otros tipos de proyectos que pretendan integrar distintos componentes y servicios como uno solo, siempre teniendo en cuenta las características de la misma, las vistas desplegadas en su diseño y las cualidades que deben tener los componentes que se pretenden integrar para que su composición sea funcional. Ya que dicha arquitectura integra servicios, sería recomendable en la investigación y/o desarrollo de un servicio que permita consumir componentes almacenados en la internet que permitan ser consumidos a modo de servicios, de esta forma se mitigaría el riesgo que está relacionado con su escalabilidad.

REFERENCIAS BIBLIOGRÁFICAS

Aucejo, E. (23 de abril de 2019). *Minijuego | ¿Qué significa Minijuego? | Definición de minijuego.* Obtenido de Geekno: https://www.geekno.com/glosario/minijuego

Bass, L., Clements, P., & Kazman, R. (1997). *Software Architecture in Practice.*

Binnie. (2023). *Desventajas del modelo basado en componentes - Abrirarchivos*. Obtenido de Abrirarchivos TI y tecnología: https://abrirarchivos.info/tema/desventajas-del-modelo-basado-en-componentes/

Bosh, J. (2000). *Design & Use of Software Architectures.* Addison-Wesley.

Cazalla, C. E. (2014). *Diseño y desarrollo de un protoripo básico de un videojuego plataformas en 2D.*

Cervera, I. (12 de abril de 2019). *Gameplay | ¿Qué significa gameplay?* Obtenido de Geeko: https://www.geekno.com/glosario/gameplay

Chen, C. (2023). *Qué son las TIC (Definición, Características y Ejemplos) - Significados*. Obtenido de Significados: https://www.significados.com/tic/

Clemens, A. S. (2002). *Component software: Beyond Object-Oriented programming.*

Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating software architectures: methods and case.* Boston: Wesley, Addison.

Collins, P. (30 de enero de 2021). *Prototipado Rápido de Juegos: ¿Por qué son Importantes en el Desarrollo de Juegos? | Starloop Studios.* Obtenido de Starloop Studios: https://starloopstudios.com/prototipado-rapido-de-juegos-por-que-son-importantes-en-el-desarrollo-de-juegos

Cruz, A. A. (2017). *Arquitectura de software para videojuegos desarrollados sobre el motor de juego Unity 3D.* julio.

DeVuego. (27 de Marzo de 2023). *HUD | Definici'on en GameDic.* Obtenido de Diccionario online de términos sobre videojuegos y cultura gamer: https://www.devuego.es/gamerdic/termino/hud

Ernest, A. (s.f.). *Fundamentals of Game Design.*

Esquivel, Y. (8 de Junio de 2022). *Cuba presentará en Unesco candidatura del son como Patrimonio Cultural Inmaterial de la Humanidad*. Obtenido de Cubadebate: http://www.cubadebate.cu/noticias/2022/06/08/cuba-presentara-en-unesco-candidatura-del-son-como-patrimonio-cultural-inmaterial-de-la-humanidad/

EUROINNOVA. (2023). *Aplicaciones interactivas | Web Oficial Euroinnova*. Obtenido de EUROINNOVA International Online Education: https://www.euroinnova.edu.es/blog/aplicaciones-interactivas

Garlan, D., & Perry, D. (1995). *Special Issue on Software Architecture.*

Garlan, D., & Perry, D. E. (1993). *Advances in Software Engineering and Knowledge Engineering Vol 2.*

González, O. (s.f.). *Documentando la arquitectura de software.*

Gutiérrez-Hernández, R. E., Álvarez, F. J., & Muñoz-Arteaga, J. (s.f.). *Arquitectura de software para Juegos Serios con Aspectos Culturales: Caso de Estudio en un Videojuego para Fórumas Temperatura.*

Herández, A. P., Pérez, K. T., & Correa, O. M. (2017). Marco de trabajo ingenieril para el proceso de desarrollo de videojuegos. *Revista Antiqoqueña de las Ciencias Computacionales y la Igeniería de Software*.

Higuerey, E. (25 de Mayo de 2020). *5 tipos de contenidos interactivos y las ventajas de usarlos*. Obtenido de Rockcontent: https://rockcontent.com/es/blog/tipos-de-contenidos-interactivos/

Hofmeister, C., Krutchen, P., Nord, R., Obbink, H., & Ran, A. (2007). *Un modelo general de diseño de arquitectura de software derivado de cinco enfoques industriales.*

IEEE. (2018). *Quienes Somos - IEEE Sección España*. Obtenido de IEEE Sección España: https://ieeespain.org/quienes-somos/

IEEE Computer Society. (2000). *Recommended Practice for Architectural Description for Software-Intensive Systems.*

Klein, M., & Kazman, R. (1999). *Atributte-based architectural styles.* Carnegie Mellon University: Technical Report.

Kruchten, P. (1999). *The Rational Unified Process.*

Llamas, J. (1 de mayo de 2020). *Tipos de tecnología - Qué es, definición y concepto | 2023 | Economipedia*. Obtenido de Economipedia: https://economipedia.com/definiciones/tipos-de-tecnologia.html

MadeType. (2023). *Made Type on Behance.* Obtenido de Made Type: https://www.behance.net/gallery/160834231/MADE-Mountain-FREE-Font/modules/956323897

Microsoft. (2003). *Visual Studio: IDE y Editor de código para desarrolladores de software y Teams*. Obtenido de Microsoft | Visual Studio: https://visualstudio.microsoft.com/es/

Microsoft Ignite. (14-17 de Noviembre de 2023). *System.Security.Cryptography Namespace*. Obtenido de Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-7.0

Morales, G. A., Nava, C. E., Fernández, L. F., & Rey, M. A. (s.f.). *Procesos de desarrollo para videojuegos.*

Olivas, I. (s.f.). *Tecnología Interactiva express.adobe.com*. Obtenido de Tecnología Interactiva: https://express.adobe.com/page/hCgEB/

Pérez, L. R. (diciembre de 2015). *Diseño y desarrollo de un prototipo de un videojuego de aventuras con contenido creado preocedimentalmente.* Obtenido de accedacris.ulpgc.es: https://accedacris.ulpgc.es/bitstream/10553/15372/2/0717485_00000_0000.pdf

Perry, D. E., & Wolf, A. (Octubre de 1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*.

Porto, G. A. (17 de junio de 2021). *Videojuego - Qué es, tipos, definición y concepto*. Obtenido de Definición: https://definicion.de/videojuego/

Pressman, R. S. (2002). *Ingeniería de Software. Un enfoque práctico.* Madrid: McGraw-Hill.

RAE. (2021). *Estatutos y Reglamento de la Real Academia Española.* Obtenido de Real Academia Española: https://www.rae.es/sites/default/files/2021-02/Estatutos%20y%20reglamento_2014_19_2_2021.pdf

RAE. (2023). *interactivo, interactiva | Definición | Diccionario de la lengua española | RAE - ASALE*. Obtenido de Real Academia Española: https://dle.rae.es/interactivo

RAE. (2023). *tecnología | Definición | Diccionario de la lengua española | RAE - ASALE*. Obtenido de Real Academia Española: https://dle.rae.es/tecnolog%C3%ADa

Ramirez, E. R. (28 de febrero de 2023). *Arquitecto Empresarial & Software & Datos | Integrador de Aplicaciones (SOA & REST & Web Services & ETL) | Data Engineer | Scrum Master*.

Obtenido de Linkedin: https://es.linkedin.com/pulse/patrones-en-la-arquitectura-de-software-elmo-renato-castro-ramirez

Reynoso, C. (2004). *Investigación, Publicaciones y Cursos de Antropología.* Obtenido de Ciencia Cognitiva y Complejidad: carlosreynoso.com.ar/arquitectura-de-software/

Reynoso, C., & Kicillof, N. (2004). *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.*

Rick Kazman, M. K. (2002). *Evaluating software architectures: methods and case studies.* Addison-Wesley.

Rollings, A., & Morris, D. (1999). *Game Architecture and Desing.*

Seco, J. A. (2001). *El lenguaje de programación.*

Shaw, M., & Clements, P. (1996). A field guide to Boxology: Preliminary classification of architectural styles for software systems. *Computer Science Departament and Software Engineerinf Institute*.

Software engineering. (2001). *ISO/IEC 9126-1.*

UNESCO. (s.f.). *El mandato y la misión de la UNESCO en resumen*. Obtenido de UNESCO: https://www.unesco.org/es/brief

Unity 3D. (2023). *Plataforma de desarrollo en tiempo real de Unity | Motor de VR, AR, 3D y 2D*. Obtenido de Unity 3D: https://unity.com/es

Universidad Carnegie Mellon. (2023). *Instituto de Ingeniería de Software*. Obtenido de Carnegie Mellon University: https://www.sei.cmu.edu/

Universidad de las Ciencias Informáticas. (s.f.). *Centro de Tecnologías Interactivas | Universidad de las Ciencias Inform'aticas*. Obtenido de Universidad de las Ciencias Informáticas: https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-tecnologias-interactivas

Visual Paradigm. (2022). *Ideal Modeling & Diagramming Tool for Agile Twam Collaboration*. Obtenido de Visual Paradigm: https://www.visual-paradigm.com/