Rick and Morty Game

Proyecto Final – Programación I

José David García Pineda - 192349

Michell Juliana Pérez Gómez – 192354

Euder Julián Pacheco Ascanio – 192356

Geraldine García Torrado - 192390

Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña

Ingeniería de Sistemas

Prof. Jesús Eduardo Guerrero Rodríguez

06 de diciembre del 2024

Introducción

La nueva era del entretenimiento digital ha dado lugar a la creación de videojuegos que fusionan la estrategia, la trama y los personajes icónicos, ofreciendo a los usuarios experiencias profundamente inmersivas. En este contexto, surge la idea de desarrollar un juego de batallas ambientado en el universo de Rick and Morty. Este videojuego propone enfrentamientos por turnos entre diversas versiones alternativas de los protagonistas, cada una equipada con habilidades únicas que reflejan fielmente sus personalidades y rasgos característicos mostrados en la serie.

El principal desafío consiste en diseñar y desarrollar un sistema que mantenga un equilibrio perfecto entre el estilo del juego y la originalidad de los personajes, permitiendo a los jugadores vivir una simulación estratégica y dinámica. Para lograr este objetivo, el proyecto se estructura con una arquitectura multicapa, donde cada capa asumirá responsabilidades bien definidas: la gestión de la lógica del juego, la persistencia de los datos de los personajes, y la provisión de una interfaz gráfica intuitiva para el usuario.

Este enfoque no solo asegura una grata experiencia de juego, sino que también garantiza que cada detalle, desde la mecánica de combate hasta las interacciones de los personajes, esté cuidadosamente alineado con el espíritu y la esencia del contexto planteado.

Objetivos

General:

- Diseñar y desarrollar un videojuego basado en el universo de Rick y Morty, con un sistema de batallas por turnos, implementando buenas prácticas de programación para garantizar un código limpio, escalable y eficiente.

Específicos:

- Usar los principios de la programación orientada a objetos (POO), que posibiliten el entendimiento, la ampliación, modificación y actualización del juego.

- Diseñar y crear un sistema de combate intuitivo, entendible y estratégico, que permita tomar decisiones tácticas a los usuarios.

- Incorporar un sistema de persistencia de datos eficiente que facilite la actualización y expansión de los datos usados en el software.

- Implementar una arquitectura multicapa que garantice la modularidad y mantenimiento del software.

- Generar una experiencia accesible y entretenida para los usuarios a través de una interfaz gráfica simple.

- Integrar ajustes y mejoras basadas en la retroalimentación de usuarios para aumentar su satisfacción.

Alcance del proyecto.

El alcance inicial del proyecto se centra en crear un juego funcional con una base sólida para combates por turnos. Este juego permitirá a los jugadores controlar versiones alternativas de los personajes, cada uno con habilidades únicas y características específicas que reflejan sus personalidades. Sin embargo, el diseño modular permite una evolución continua del sistema. En el futuro, se podrían incluir:

- Más personajes y habilidades únicas.
- Nuevos modos de juego, como batallas en equipo o torneos.
- Integración multijugador local o en línea.
- Efectos visuales y sonoros para mejorar la inmersión.
- Aseguramiento de la integridad y seguridad de los datos durante el juego.

Este proyecto no solo aspira a proporcionar un entretenimiento al usuario final, sino también a establecer un estándar de excelencia en el diseño y desarrollo de software. Al demostrar cómo aplicar de manera efectiva los principios de la programación orientada a objetos y las mejores prácticas de ingeniería de software.

Descripción de los módulos desarrollados.

El proyecto al desarrollarse bajo varias capas o módulos con responsabilidades específicas y que trabajan en conjunto para ofrecer una experiencia coherente y funcional, se divide de la siguiente manera:

1. **Módulo de la Lógica del Negocio:**

   Este módulo se responsabiliza de la definición de las reglas del juego, del control de las interacciones entre los personajes y las acciones que puede realizar cada uno durante una ronda o partida. Además, se encarga del flujo del juego, el sistema de turnos y el establecimiento de las condiciones para ganar o perder la batalla. Los componentes de esta capa son:

   - Clase principal o base **Character**:
     - Esta clase funciona como la plantilla principal para la creación de los personajes del juego. En ella se definen atributos generales como nombre, salud, poder y defensa y se inicializan con el constructor.
     - Se trata de una clase abstracta ya que en ella hay métodos con y sin implementación que definen las acciones a realizar en cada batalla.
     - Contiene los métodos comunes: attack, defend, heal, useSpecialAbility (método abstracto para habilidades únicas). Cada uno con las responsabilidades y funcionalidades asignadas.
     - En ella se aplican principios de POO como abstracción en los métodos abstractos y encapsulamiento al usar modificadores de acceso junto a getters y setters.

- Subclases **Rick y Morty**:
  - Estas son clases hijas que heredan o extienden de la clase Character.
  - Ambas implementan o sobrescriben el método para las habilidades especiales useSpecialAbility(), de esta manera: Rick, duplica el daño infligido al oponente; Morty, cura un porcentaje de su salud máxima.
  - En esta clase se pueden encontrar los principios de POO de herencia tanto de métodos como atributos y el polimorfismo en la sobreescritura del método.

- Clase **Game**:
  - Es la clase que maneja la lógica central del juego, permitiendo la interacción con el usuario. Establece la manera de seleccionar personajes, la alternancia de turnos, procesamiento de acciones y verificaciones de las condiciones de victoria.
  - Incluye métodos para: Cargar los personajes disponibles, mostrar opciones disponibles al jugador (showOptions). Ejecutar acciones seleccionadas (executeAction).
  - Controlar el flujo del juego (determinación del turno inicial, alternancia de turnos y finalización de la partida).
  - Su diseño modular y manejo adecuado de excepciones aseguran la robustez y la mantenibilidad del código.

2. Módulo de Persistencia (Base de Datos):

Se maneja un archivo JavaScript Object Notation (JSON) que funciona como una base de datos para el videojuego de batallas de Rick and Morty, proporcionando una lista detallada de los personajes disponibles para los jugadores. El archivo está organizado en dos categorías principales: Ricks y Mortys. Cada categoría contiene una lista de personajes, cada uno con atributos específicos.

Estructura general del JSON:

- Ricks: Esta sección incluye varios personajes diferentes de Rick, cada uno con sus atributos de name, health y power. Estos atributos definen la capacidad de combate y resistencia de cada Rick en el juego.
- Mortys: Similar a la sección de Ricks, esta parte contiene una lista de personajes de Morty, cada uno con atributos de name, health y power.

En el juego se utiliza para cargar la información de los personajes al inicio. Proporciona los datos necesarios para:

- Mostrar la lista de personajes disponibles para que los jugadores seleccionen.
- Definir las estadísticas de combate de cada personaje durante las batallas.
- Mantener una estructura organizada y fácil de actualizar.

3. **Módulo de Acceso a Datos:**

   Este módulo se encarga de la lectura y carga de los datos de los personajes desde un archivo JSON, transformándolos en objetos Java que pueden ser utilizados en el juego. Contiene:

   - Clase **DatabaseHandle**r: Es esencial para cargar los personajes desde un archivo JSON y convertirlos en objetos utilizables dentro del juego. Utiliza la biblioteca Gson para facilitar esta conversión, asegurando que los datos estén correctamente estructurados y disponibles para ser usados en la lógica del juego.
   - Define el método loadCharacters, que: Lee el archivo de persistencia (characters.json) y convierte los datos en listas de objetos Rick y Morty organizados en la clase anidada Characters.
   - Proporciona métodos de acceso a las listas de personajes para su uso en la capa lógica del negocio.

4. **Módulo de Presentación (Interfaz Gráfica):**

   Inicialmente, se planteó desarrollar la interfaz gráfica del proyecto utilizando Android Studio, ya que ofrecía un entorno robusto para crear aplicaciones móviles modernas con una experiencia de usuario más avanzada. Sin embargo, debido a problemas de ejecución, se decidió migrar a Java Swing, una tecnología más sencilla y rápida de implementar para interfaces gráficas en aplicaciones de escritorio. Esta

transición permitió enfocarse en las funcionalidades principales del juego, asegurando una solución funcional y portátil que cumpliera con los requisitos del proyecto sin comprometer la jugabilidad ni la interacción del usuario.

Este módulo se enfoca en proporcionar una experiencia interactiva al usuario a través de una interfaz sencilla pero funcional. A continuación, se detalla su descripción:

- Pantalla de selección de personajes: Un panel que permite al jugador seleccionar un personaje Rick y un Morty de una lista desplegable. Incluye un botón para iniciar la batalla, mostrando un diseño visual limpio con bordes y colores destacados.
- Pantalla de Batalla: Contiene los siguientes elementos:
    - Panel de Estadísticas: Muestra las estadísticas de salud de Rick y Morty mediante etiquetas (JLabel) y barras de progreso (JProgressBar).
    - Registro de Eventos: Un área de texto (JTextArea) donde se registran los eventos de la batalla en tiempo real.
    - Acciones del Jugador: Botones (JButton) que permiten al jugador realizar acciones como atacar, defenderse, curarse o usar una habilidad especial.
    - Indicador de Turno: Una etiqueta que señala de quién es el turno en cada ronda.
- Transiciones entre Pantallas: Uso de un CardLayout para alternar entre las pantallas de selección de personajes y batalla de manera fluida.

- Gestión de Eventos: Los botones están vinculados a listeners (ActionListener) para ejecutar las acciones del juego. Las transiciones entre pantallas y la actualización de estadísticas se gestionan dinámicamente.
- Las barras de progreso visualizan el estado de salud de los personajes y cambian dinámicamente según el daño recibido.
- Mantiene un control lógico sobre los turnos del jugador y actualiza el indicador correspondiente.
- Los personajes se cargan desde un DatabaseHandler, integrando este módulo con la base de datos del proyecto.

La estructura organizada y modular del software permite una fácil extensión y mantenimiento, lo cual es crucial para el desarrollo eficiente y efectivo del juego.

Tecnologías empleadas y justificación.

En el desarrollo del juego de batallas, se emplearon diversas tecnologías que abarcan programación, manejo de datos y diseño de interfaces gráficas. A continuación, se describen las principales tecnologías utilizadas:

1. Lenguaje de Programación: Java.

Es el lenguaje principal para implementar la lógica del negocio, la interacción con datos y la interfaz gráfica con Java Swing. Sus características de POO facilitan la implementación de clases jerárquicas como Character, Rick y Morty. Ofrece herramientas robustas para manejar flujos de control, excepciones y colecciones necesarias para la lógica del juego.

Justificación: Conocimiento del lenguaje. Popularidad y soporte extenso. Portabilidad, ya que el código puede ejecutarse en diversas plataformas gracias a la Máquina Virtual Java (JVM).

2. **Librerías Nativas de Java:**
    - java.util.List y ArrayList: Para gestionar las listas de personajes.
    - java.util.Random: Para determinar de forma aleatoria el turno inicial.
    - java.io: Para la carga y lectura de archivos JSON en el sistema de persistencia.

Justificación: Son ligeras, eficientes y están incluidas en el JDK, evitando dependencias externas innecesarias.

3. **JSON (JavaScript Object Notation):**

Formato de archivo utilizado para almacenar los datos de los personajes (characters.json).

Justificación: Ligero, fácil de leer y escribir, y ampliamente compatible con diversas tecnologías. Ideal para datos estructurados que requieren actualizaciones frecuentes o expansión futura.

4. **Biblioteca Gson:**

Manejo del archivo characters.json en el módulo de persistencia. Convierte datos en formato JSON a objetos Java, permitiendo trabajar de forma natural con listas de personajes.

Justificación: Simplifica la deserialización de datos (Paso de JSON a objetos). Ideal para proyectos con archivos JSON que requieren alta compatibilidad y facilidad de uso.

5. **Java Swing:**

   Se utilizó como la principal biblioteca para construir la interfaz gráfica de escritorio. Swing proporciona componentes como botones, paneles, etiquetas y barras de progreso, que se personalizaron para adaptar la estética y funcionalidad del juego. Esta elección permitió crear una interfaz sencilla pero funcional en un entorno multiplataforma.

   Justificación: Java Swing fue seleccionado por su capacidad de ofrecer una solución práctica y eficiente para las necesidades del proyecto, priorizando la funcionalidad, la compatibilidad y la facilidad de desarrollo en un entorno de escritorio.

6. **Modelo de Arquitectura Multicapa:**

   División en capas de lógica de negocio, persistencia de datos y presentación gráfica.

   Justificación: Promueve la separación de responsabilidades y permite mejorar o reemplazar partes del sistema sin afectar otras capas.

7. **Control de Versiones con Git:**

Gestión del código fuente a través de un sistema de control de versiones distribuido.

Justificación: Es la herramienta estándar en desarrollo de software colaborativo. Asegura el historial del proyecto y la integración continua.

SDLC (Software Development Life Cycle)

El Ciclo de Vida del Desarrollo de Software para este proyecto, está descrito por cada una de las siguientes etapas:

1. **Planificación:** Durante esta etapa se establecieron varios objetivos esenciales para el correcto desarrollo del proyecto:
    - Identificar los requisitos iniciales del juego como mecánicas de combate, selección de personajes y actualización de estadísticas. Establecer cada una de las responsabilidades de los diferentes módulos que lo componen.
    - Definir cada uno de los roles y compromisos de cada uno de los integrantes del equipo, para lograr un trabajo equilibrado y colaborativo.
    - Seleccionar cada uno de las tecnologías y herramientas adecuadas.

    Roles:

    - Geraldine García Torrado: Módulo de Lógica del Juego.

    Responsabilidades: Desarrollar la lógica interna del juego, incluyendo las mecánicas de combate, manejo de turnos y actualización de estadísticas de personajes.

    - Euder Julián Pacheco Ascanio: Persistencia.

Responsabilidades: Implementar la base de datos por medio de un JSON.

- Michell Juliana Pérez Gómez: Módulo de Acceso a Datos.

Responsabilidades: Implementar la capa de persistencia, encargada de cargar y guardar los datos de los personajes desde el archivo JSON.

- José David García Pineda: Módulo de Interfaz Gráfica (UI/UX).

Responsabilidades: Diseñar y desarrollar la interfaz gráfica del usuario (GUI) en Android Studio.

2. **Análisis:** Durante esta fase, se definieron tanto los requisitos funcionales como los no funcionales del sistema y se identificaron posibles riesgos y estrategias para mitigarlos:

    - Definir los requisitos funcionales del juego como: Los jugadores deben poder seleccionar un personaje y realizar acciones como atacar, defender, curarse y usar habilidades especiales. La victoria se decide cuando la salud de uno de los personajes llega a cero. Una interfaz intuitiva, permitiendo a los jugadores navegar fácilmente por las opciones.
    - Requisitos no funcionales: Asegurar que el juego funcione de manera fluida y sin problemas, optimizando el uso de recursos. Carga eficiente de datos desde el archivo JSON.
    - Identificar posibles riesgos: Posibles errores al cargar datos desde el archivo JSON.

3. **Diseño:**

Esta etapa se centra en crear una estructura detallada del sistema que facilite su construcción y asegure que todos los componentes funcionen armoniosamente:

- Se establece el diseño del sistema mediante la arquitectura multicapa con una capa de presentación implementada con Java Swing, una capa lógica que tiene las reglas del juego y una capa de persistencia que maneja la base de datos para la carga y almacenamiento los personajes en las partidas.
- Se representan las relaciones entre cada uno de los componentes del sistema mediante diagramas UML, que ayudan a entender de una manera más clara la estructura y objetivos del proyecto.
- Se diseña un plan de integración que garantice que la lógica y los datos funcionen antes de integrarse con la UI.

4. **Implementación:**

Cada integrante desarrolla su módulo de forma paralela, usando todas las tecnologías descritas anteriormente, así:

- **Módulo de Lógica del Juego:** Desarrollo de las clases base (Character, Rick, Morty) y métodos principales como attack, heal y useSpecialAbility. Implementación de la clase Game para gestionar el flujo del combate.
  Responsable: Geraldine García Torrado.
- Modulo de Persistencia: Elaboración del JSON y documentación.
  Responsable: Euder Julián Pacheco Ascanio.

- **Módulo de Acceso a Datos:** Implementación de la clase DatabaseHandler para cargar datos desde el archivo JSON y mapearlos a objetos Java.

  Responsable: Michell Juliana Pérez Gómez.
- Módulo de Interfaz Gráfica: Diseño de la interfaz con Java Swing para la selección de personajes y el combate. Integración de eventos de botón con la lógica del juego.

  Responsable: José David García Pineda.

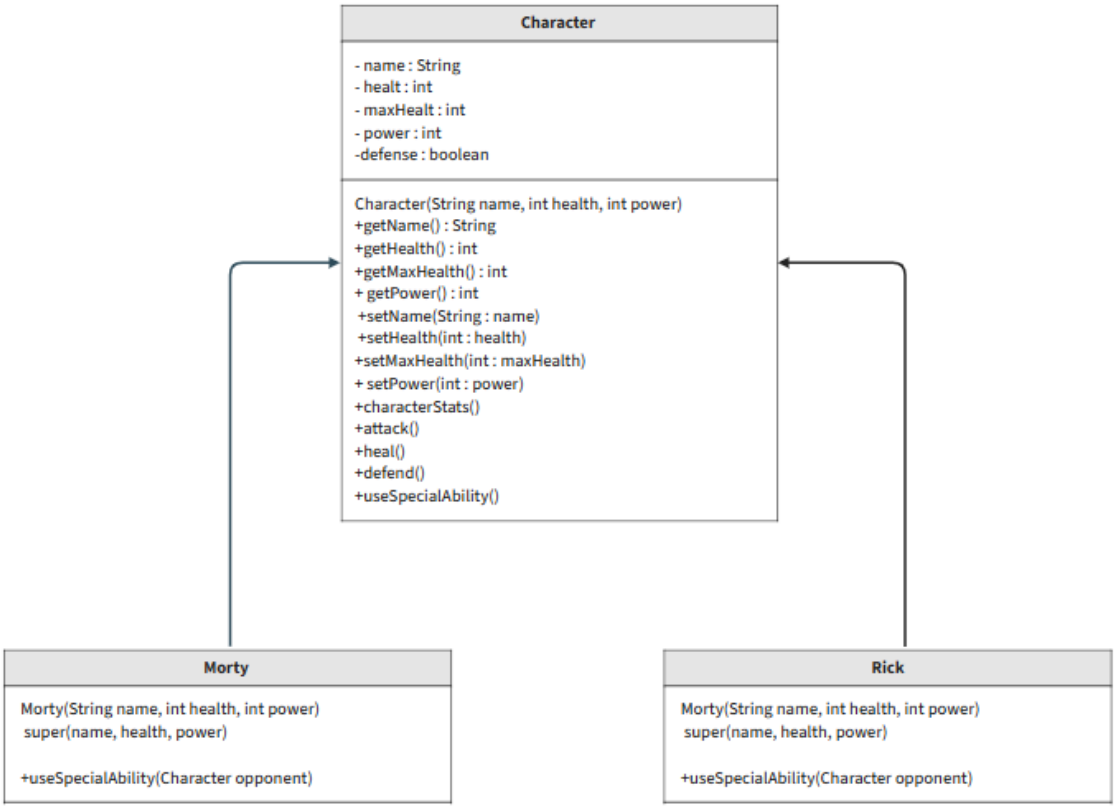
5. **Pruebas:** Esta fase asegura que el software funcione de manera óptima y cumpla con los requerimientos establecidos, para esto se aplican:
   - Pruebas Unitarias: Verificación individual de los métodos attack, heal y useSpecialAbility. Validación del comportamiento esperado del juego en distintos escenarios.
   - Pruebas de Integración: Comprobación de la interacción entre los módulos de lógica, datos y UI.
   - Pruebas de Usabilidad: Evaluación de la claridad y facilidad de uso de la interfaz gráfica.
   - Pruebas de Rendimiento: Verificación de la velocidad de carga de datos desde el archivo JSON.

6. **Mantenimiento:** La fase de mantenimiento es una etapa continua que asegura la calidad y la relevancia del videojuego a lo largo del tiempo. Se basa en:
   - Analizar, identificar y priorizar errores.

- Resolución de errores reportados por usuarios o detectados durante el despliegue.
- Actualización constante de funcionalidades y optimización de las mismas.
- Actualización de la documentación técnica.
- Planificar y desarrollar nuevas características basadas en la retroalimentación de los usuarios.
- Mantener un registro detallado de todos los cambios realizados.

Diagramas UML

1. **Diagrama de clases:** Este diagrama describe las clases que interactúan en el software, una clase base llamada Character y las clases hijas Morty y Rick. En cada una se encuentran los atributos junto a sus métodos.



2. Diagrama de casos de uso:

El diagrama de casos de uso representa las interacciones entre el jugador y el sistema del juego Rick and Morty (Representado dentro del cuadrado). Muestra las funcionalidades principales disponibles para el jugador, como Iniciar Batalla, Seleccionar Personaje, Atacar, Defenderse, Curarse, Usar ataque especial y Reiniciar Batalla. Las relaciones como include y extend indican dependencias o extensiones entre las funcionalidades.