TALLER PROGRAMACION AVANZADA: ANALISIS Y DISEÑO “DECKBULDER”

Integrantes: Juan Almonte ,Sean Castillo.

Correo Electronico:

juan.almonte@alumnos.ucn.cl,
sean.castillo@alumnos.ucn.cl.

Rut:22.168.387-0,20.542.721-K.

Paralelo:C1.

Profesor: Tomas Reiman.

Fecha:12 de abril del 2024.

Indice

Introducción: Los Juegos de cartas

Los juegos de cartas representan un pasatiempo vivido con fervor en todo el mundo, transportando a sus jugadores a una variedad de experiencias. Desde simples momentos de diversión hasta dimensiones más complejas y estratégicas, estos juegos cautivan a una amplia gama de entusiastas. En ellos, la estrategia y las posibilidades se entrelazan, desafiando la imaginación de los estrategas los cuales se desafían constantemente para innovar en su estilo de juego.

Magic: The Gathering, concebido por Richard Garfield, ha dejado una huella indeleble en el paisaje de los juegos de estrategia. Surgió de la pasión de Garfield por los juegos de cartas, fusionando elementos de estrategia, fantasía y coleccionismo. A partir de dos proyectos anteriores fallidos, Garfield rescató las ideas principales que dieron vida a Magic. En "SafeCracker", se exploraba la construcción de una caja fuerte para intentar abrir las de los adversarios, ofreciendo un vistazo primigenio al concepto de mazos. Mientras que en "Five Colors", se destacaban las funcionalidades y efectos únicos de cada carta, capaces de cambiar radicalmente el curso de una partida. Esta amalgama de conceptos dio lugar a un juego innovador que, con su sistema evolutivo, ha logrado permear en todos los estratos de la sociedad.

Del Bosquejo al Código: Comparando Procesos en Diseño y Programación

El camino hacia el éxito sin precedentes de Magic: The Gathering no comenzó con el juego definido que conocemos hoy en día. Para lograr que el juego se destacara como algo único, fue necesario un exhaustivo proceso de análisis del producto, estudio del entorno, establecimiento de bases y definición de las diversas funciones que un juego de esta envergadura debía poseer. Este proceso, fundamental para el éxito, llevó a Magic aproximadamente tres meses para conceptualizar sus bases y dar forma a su versión inicial, conocida como "Alpha".

El primer prototipo de Magic se materializó mediante cartas de recorte y una amplia gama de mecánicas de juego. Richard Garfield experimentó con distintos tipos de cartas, habilidades y reglas con el fin de encontrar el equilibrio perfecto. Tras múltiples iteraciones finalmente alcanzaron una versión del juego que tenía el potencial necesario para triunfar.

El desarrollo de software generalmente sigue un proceso sistemático que podemos comparar con el caso presentado, constando de varias fases, entre las cuales el análisis y diseño son fundamentales para crear un software útil, exitoso y duradero.

La etapa de análisis en el ciclo de vida del software es fundamental para comprender plenamente las necesidades y requisitos del sistema. Implica una comunicación estrecha entre desarrolladores y clientes para identificar con precisión las funcionalidades

esenciales. El objetivo es crear una especificación detallada de los requerimientos del sistema, que abarque funciones, restricciones y otros aspectos relevantes.

Durante la fase de diseño del ciclo de vida del software, se exploran y evalúan diversas opciones de implementación para el sistema en construcción. Este proceso implica tomar decisiones cruciales sobre la arquitectura y la estructura general del software. Dado que el diseño es una etapa compleja, se sugiere realizarlo de manera iterativa. Esto significa que se pueden llevar a cabo múltiples iteraciones para refinar y mejorar la solución, garantizando así un diseño óptimo y adaptado a las necesidades del proyecto.

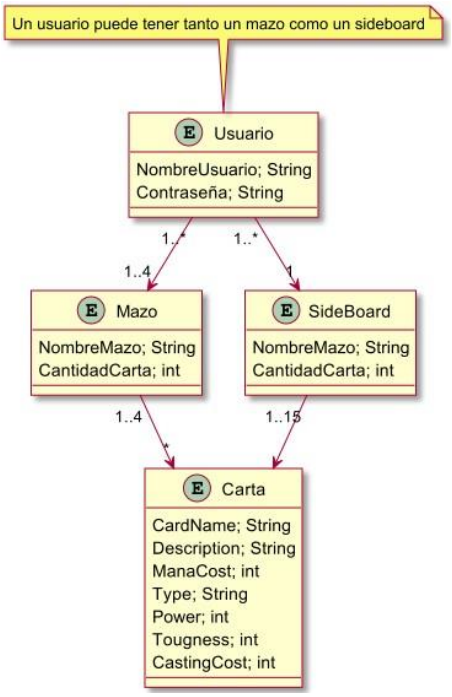
Como se puede apreciar la creación de las cartas magic es, aunque de manera primitiva se usaron conceptos similares a los que los programadores emplean con el fin de realizar el mejor juego posible.

Problemática Actual:

Un entusiasta fanático del “magic” nos ha encargado diseñar un software que a través de una base de datos pueda crear múltiples opciones de mazos para ello describiremos como implementaremos la problemática presentada

Modelo de Dominio

Un modelo de dominio es una representación visual de los conceptos y objetos interconectados del mundo real, que permite una comprensión clara y concisa de la estructura de un sistema. Estos diagramas son herramientas poderosas que facilitan la comprensión de las relaciones entre los diferentes componentes de un proyecto o sistema, lo que ayuda en la toma de decisiones informadas.



Este es un diagrama de clases que representa la relación entre diferentes entidades en un sistema de gestión de mazos de cartas. Las principales entidades son:

Usuario: Representa a un usuario del sistema. Tiene atributos como NombreUsuario y Contraseña de tipo String.

Mazo: Representa un mazo de cartas construido por el usuario. Tiene atributos como NombreMazo de tipo String y CantidadCarta de tipo int.

SideBoard: Similar al Mazo, pero representa un conjunto adicional de cartas que el usuario puede utilizar para modificar su mazo principal.

Carta: Representa una carta individual. Tiene atributos como CardName, Description y Type de tipo String, ManaCost, Power, Toughness y CastingCost de tipo int.

Las relaciones entre estas entidades son las siguientes:

Un Usuario puede tener un Mazo y un SideBoard.

Un Mazo y un SideBoard están compuestos por múltiples Cartas.

La relación entre Mazo y Carta, así como entre SideBoard y Carta, es de composición (1..*), lo que indica que un Mazo o SideBoard puede contener una o más Cartas.

Contratos

Los contratos son acuerdos formales entre las partes involucradas en el desarrollo de software que especifican las responsabilidades, expectativas y restricciones de cada una de ellas en relación con las operaciones o funciones del sistema.

Se mostraran los contratos realizado para la realización del programa:

1. LecturaCsv()

Descripción Lee un archivo CSV para proporcionar datos a otros métodos del programa.

Pre-condiciones: Existencia de un archivo CSV legible.

Post-condiciones: Los datos leídos coinciden con los requeridos por los métodos.

2. LecturaArchivo()

Descripción Lee un archivo TXT para proporcionar datos a otros métodos del programa.

Pre-condiciones Existencia de un archivo TXT legible.

Post-condiciones: Los datos leídos coinciden con los requeridos por los métodos.

3. RegistrarUsuario(String nombreUsuario, String contrasenia)

Descripción: Registra un nuevo usuario para iniciar sesión en el sistema.

Pre-condiciones: Datos de inicio de sesión coherentes con lo solicitado por el programa.

Post-condiciones: El nuevo usuario se añade a la lista de usuarios registrados.

4. ConstruirMazo()

Descripción: Crea un nuevo mazo con cartas solicitadas por el usuario.

Pre-condiciones: Datos coherentes de entrada y lista de mazos no llena.

Post-condiciones: Se agrega un nuevo mazo a la lista de mazos disponibles.

5. ModificarMazo()

Descripción: Modifica un mazo existente con nuevas cartas solicitadas por el usuario.

Pre-condiciones: Existencia del mazo en la lista y la lista no está vacía.

Post-condiciones: Los datos del mazo se modifican según las nuevas cartas especificadas.

6. ExportarMazo()

Descripción: Crea un archivo TXT que contiene los datos del mazo creado por el usuario.

Pre-condiciones: La lista de mazos no debe estar vacía y debe existir al menos un mazo.

Post-condiciones: Se crea un nuevo archivo TXT con los datos del mazo.

7. AgregarSliboard()

Descripción: Crea un nuevo "Sliboard" (un mazo más pequeño) con cartas solicitadas por el usuario.

Pre-condiciones: Datos coherentes de entrada y la lista de mazos no está llena.

Post-condiciones: Se agrega un nuevo "Sliboard" a la lista de mazos disponibles.

8. ModificarSliboard()

Descripción: Modifica un "Sliboard" existente con nuevas cartas solicitadas por el usuario.

Pre-condiciones: Existencia del "Sliboard" en la lista de mazos.

Post-condiciones: Se realiza la modificación del "Sliboard" según las nuevas cartas especificadas.

9. ExportarSliboad()

Descripción: Crea un archivo TXT que contiene los datos del "Sliboard" creado por el usuario.

Pre-condiciones: La lista de mazos debe contener al menos un "Sliboard".

Post-condiciones: Se crea un nuevo archivo TXT con los datos del "Sliboard".

10. ImprimirCarta(String nombre)

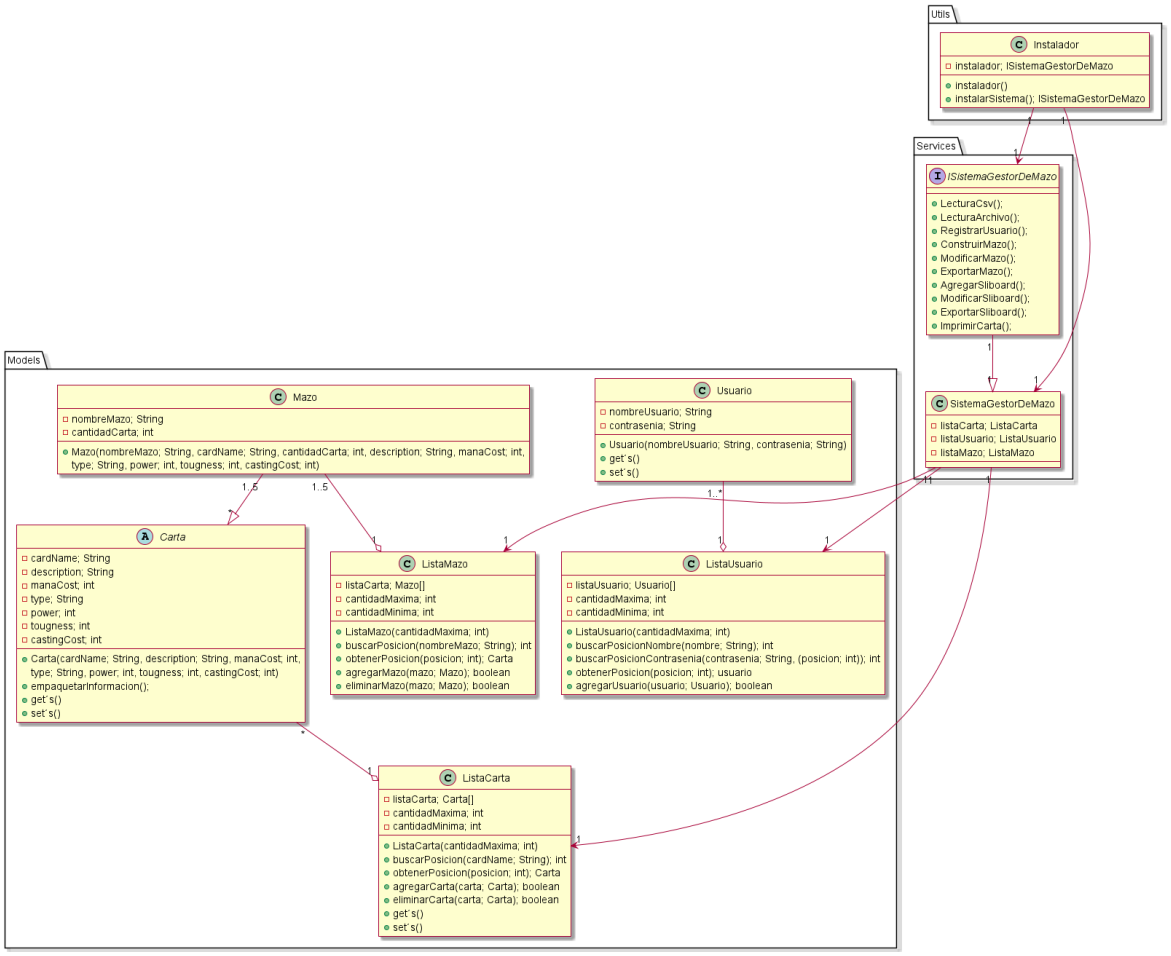
Descripción: Imprime los datos de una carta específica solicitada por el usuario mediante su nombre.

Pre-condiciones: Los datos de lectura de archivos son coherentes y los valores no son negativos.

Post-condiciones: Se imprime por pantalla la información de la carta especificada.

Diagrama de clase

Un diagrama de clases es una representación visual de las clases, interfaces y sus relaciones en un sistema orientado a objetos. Las clases, fundamentales en la programación orientada a objetos, representan entidades del mundo real con atributos privados y métodos públicos. Las relaciones entre las clases se muestran con líneas que representan diferentes tipos de asociaciones, brindando una visión clara de la estructura del sistema y las relaciones entre sus componentes.



Este diagrama de clases representa el diseño de un sistema de gestión de mazos de cartas, diseñado específicamente para la construcción de mazos utilizando cartas de "Magic: The Gathering". Aquí hay una descripción detallada de las clases y su funcionalidad:

Usuario: Esta clase representa a un usuario del sistema, con atributos como nombreUsuario y contraseña, ambos de tipo String.

Mazo: Representa un mazo de cartas construido por el usuario. Tiene atributos como nombreMazo, de tipo String, y cartas (un conjunto de cartas que componen el mazo), que son instancias de la clase padre Carta.

Carta: Representa una carta individual, con atributos como cardName (nombre de la carta), manaCost (costo de maná), tipoCard (tipo de carta), texto (descripción), power (poder), toughness (resistencia) y Casting Cost (costo de lanzamiento), cada uno con su tipo de dato correspondiente.

ListaCartas: Esta clase es un arreglo de datos que almacena instancias de Carta. Proporciona métodos para agregar, obtener, eliminar y buscar cartas dentro de la lista.

ListaUsuario: Similar a ListaCartas, esta clase es un arreglo de datos que almacena instancias de Usuario. También ofrece métodos para agregar, eliminar y buscar usuarios.

SistemaGestionDeMazos: Contiene instancias de ListaUsuario, ListaCartas y ListaMazos, y proporciona métodos para inicializar el sistema a partir de archivos. Es crucial para facilitar la comunicación entre las diferentes clases y gestionar de manera efectiva sus funciones.

ISistemaGestionDeMazos: Esta interfaz contiene métodos para realizar operaciones relacionadas con la carga de la base de datos al sistema, la gestión de mazos y la gestión del dashboard.

Instalador: Crea una copia del SistemaGestionDeMazos para evitar que el usuario modifique o afecte datos importantes del programa. Además, interactúa con el usuario para realizar funciones específicas y operar las funcionalidades del programa de manera adecuada.

Conclusión

El presente trabajo abordó el diseño y la implementación de un sistema de gestión de mazos para el juego de cartas coleccionables "Magic: The Gathering". A través de un enfoque metódico, se analizó la problemática planteada y se desarrolló una solución integral que cumple con los requisitos establecidos.

Gracias al enfoque riguroso y al cumplimiento de las etapas fundamentales de análisis y diseño, se logró desarrollar un sistema robusto y eficiente para la gestión de mazos de cartas de "Magic: The Gathering". Este sistema permite a los usuarios registrarse, construir y modificar mazos personalizados, así como buscar y visualizar información detallada sobre las cartas disponibles.

En resumen, este trabajo demuestra la importancia de seguir una metodología sólida y estructurada en el desarrollo de software, lo que conduce a soluciones efectivas y escalables. El sistema desarrollado no solo cumple con los requisitos establecidos, sino que también sienta las bases para futuras mejoras y expansiones, asegurando su relevancia y utilidad a largo plazo en el mundo de los juegos de cartas coleccionables.