Breve descripción del problema.

El objetivo de este proyecto es realizar una simulación del juego tradicional memoria que se trata de un numero de cartas que se encuentra boca abajo cada jugador puede levantar 2 cartas, si al revisar las dos cartas son iguales el jugador actual gana un punto y así continua hasta que no quede ni una carta, gana el que al final tenga mas cartas.

El detalle de este juego es que se tiene que hacer una conexión cliente-servidor donde el cliente lo único que hace es mostrar los datos en pantalla, el servidor es quien tiene toda la lógica de si son iguales, si se suman puntos, etc.

Otro detalle es que nos están diciendo que los recursos de memoria son escasos por lo que solo podemos tener en memoria solo 20 cartas escritas y en total son 64, por lo que tenemos que implementar un algoritmo de paginación para que el resto de las cartas se guarden en disco

Diagrama de clases.

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Descripción de las estructuras de datos desarrolladas.

Arreglo:

Este se implemento para el caso del servidor en cada espacio guardar el puntero al objeta donde esta la información de cada carta que esta en memoria

Matriz

La matriz de puede considerar que es un arreglo que en cada posición tiene otro arreglo y en este caso se uso en el cliente para guardar el puntero a los botones de cada carta que se muestra en pantalla

Patron de diseño singleton:

Este patrón se uso en varias clases como la de matriz en memoria y matriz en disco, este patrón se uso para que en vez de estar pasar una instancia de como parámetro a otra clase mejor en cada clase se puede llamar directamente, porque el patron singleton solo nos permite acceder a una sola instancia, por lo que aunque se llama al constructor varias veces siempre va a retornar la misma instancia.

Descripción del funcionamiento del algoritmo de paginación

Para este algoritmo se crearon dos clases “Matriz\_memoria” y “Matriz\_disco” que por errores de conceptos lo que hace cada una esta inverso “Matriz\_memoria” trabaja la información de disco y “Matriz\_disco” trabaja la información en memoria. Pasemos a explicar cada una.

“Matriz\_memoria”:

Esta tiene como atributo un arreglo de string donde están todos posibles tipos de cartas

Tiene una función “**generar\_matriz**” que hace una matriz de 8x8 con los tipos de cartas revueltos y guarda esa matriz en un txt.

Tiene otra función **“demetarjeta(posicion)”** que lo que hace es leer el txt y de vuelve la información que está en la posición dada.

Tiene una última función “**cambieestadotrajeta(posicion)”** que lo que hace estado cambiar el estado de que si ya se encontró su pareja.

“Matriz\_disco”

Esta tiene un atributo “**espacio\_en\_tarjetas**” que lo que me da es el índice el arreglo donde voy a cargar una carta

Un atributo “**ptr\_tarjetascargadas**” que es un arreglo donde se guardan los punteros de los objetos de cartas que están en memoria

Tiene como atributo una instancia de “**Matriz\_memoria**” para llamar a sus funciones.

Tiene la función **“deme\_la\_imagen(posicion)”** que lo que hace es ver si la carta esta cargada llamada a la función “estacarda(posicion)”, si lo esta nada mas retorna la imagen asociada a la posición dada. en caso de no ser así hay dos escenarios, si el espacio que le toca esta ocupado se llama a la función de “Matriz\_memoria->cambiaestadotarjeta(posicion)” si la carta es vuelta, y después se llama a la función ”cargarTarjeta(,posición, “Matriz\_memoria->demetarjeta(posicion))”, en el escenario de que la posición este vacia nada mas se llama ”cargarTarjeta(,posición, “Matriz\_memoria->demetarjeta(posicion))”,

La función “**estacargada(posicion)**”lo que hace es recorrer el arreglo ptr\_tarjetascargadas” si no lo esta retorna un “NO” si esta retorna la posición en la que se encuentra.

La función”**cargartarjeta(posición,informacion)”**  lo que hace es crear un objeto “tarjetas\_cargadas” con toda la información y guardarla en el arreglo “ptr\_tarjetascargadas”

La ultima función es **“demetipo(posicion)”**  que hace lo mismo que deme\_la\_imagen pero retorna el tipo de carta

Protocolo de comunicación entre cliente y servidor

El protocolo que se uso es de envíos de string donde al recibir un mensaje este envia el mensaje a una clase llamada “lectura\_mensaje” donde se lee el mensaje y esta llama a funciones segun lo que entro

f. Problemas encontrados en forma de bugs de github: En esta sección se detalla cualquier

problema que no se ha podido solucionar en el trabajo.

El único problema que no se ha podido arreglar es que el envió de imágenes no se ha podido realizar por medio de base64 por lo que envía en binario.

Jira

Captura de pantalla de computadora

Descripción generada automáticamenteUna captura de pantalla de una computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamenteUna captura de pantalla de una computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamenteUna captura de pantalla de una computadora

Descripción generada automáticamenteUna captura de pantalla de una computadora

Descripción generada automáticamente

Link: <https://deylan6.atlassian.net/jira/software/projects/PJDM/boards/1/roadmap?shared=&atlOrigin=eyJpIjoiNzUzNzhiZDVlYmUxNGM2MThhY2Q0YTFmODRlNTY4OWQiLCJwIjoiaiJ9>

Github

Link: <https://github.com/deylan16/Proyecto-1-Datos-2-.git>

}