Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Estructura de Datos
Sección P
Ing. Mario Bautista
Aux. Marco Galindo

PROYECTO

Objetivos

- Que el estudiante adquiera los conocimientos necesarios acerca de la aplicabilidad y la implementación de las Estructuras de Datos en una aplicación en ambiente comercial.
- Que el estudiante aprenda y aplique los conceptos referentes a punteros y memoria dinámica en la solución de problemas.
- Definir algoritmos de búsqueda y recorridos para las estructuras creadas.
- Que el estudiante adquiera habilidad en la abstracción de los problemas para desarrollar soluciones óptimas al implementar las estructuras de datos.

Descripción

Se le solicita al estudiante crear un juego que sea como el juego Battleship, pero mejorado. El proyecto será realizado con el uso de diferentes tecnologías como lo son los servicios web y Android.

Descripción del Juego

Este juego está basado en el juego de mesa, lanzado en 1987, conocido como Battleship. El objetivo del juego es hundir la flota entera del oponente antes de que el oponente destruya la del jugador.

Tableros

Cada jugador maneja dos tableros divididos en casillas. Cada tablero representa una zona diferente: la propia y la contraria. Este tablero puede tener hasta 4 niveles, por lo tanto será un cubo disperso. En uno de los tableros, el jugador coloca sus naves y registra los «tiros» del

oponente; en el otro, se registran los tiros propios haciendo diferencia entre los tiros acertados y los que no.

Naves

Al comenzar, cada jugador coloca sus naves en el primer tablero, de forma secreta, invisible al oponente.
Existen 4 diferentes tipos de naves:

- Satelite
- Avion
- Barco
- Submarino

Cada una de las naves puede ocupar los siguientes espacios:

- Satelite 1 espacio
- Avion 3 espacios horizontales y 4 verticales o 2 espacios horizontales y 2 verticales en forma de T
- Barco 1, 2 o 3 espacios en forma horizontal o vertical
- Submarino 1, 2 o 3 espacios en forma horizontal o vertical

Al hablar de horizontal o vertical, se toma como referencia ver el tablero en vista de planta.

Desarrollo del juego

Al iniciar el juego se debe definir, del lado del servidor, los parámetros del juego. Estos son:

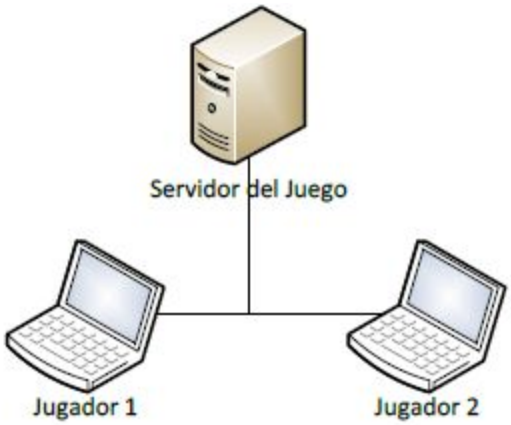
- Numeros de naves por nivel
- Tipo de disparo
    - Uno por uno
    - Rafaga (tiros consecutivos por turno)
- Las dimensiones del tablero serán dinámicas y serán definidas tomando en cuenta las posiciones de las naves.
- Variante
    - Normal
        - El juego termina hasta que haya un ganador
    - Tiempo
        - El juego termina a un tiempo definido, para escoger el ganador, al final del tiempo, se contabiliza la cantidad de aciertos que tenga cada uno. Si los dos tienen la misma cantidad de aciertos, se declara un empate.

Una vez todas las naves han sido posicionadas, se inicia una serie de rondas. En cada ronda, cada jugador en su turno «dispara» hacia la flota de su oponente indicando una posición (las coordenadas de una casilla), la que registra en el segundo tablero. Si esa posición es ocupada por parte de una nave contraria, se marcará la casilla en los dos tableros como tocada, si todavía quedan partes de la nave sin dañar, o hundida si con ese disparo la nave ha quedado totalmente destruida (esto es, si la acertada es la última de las casillas que conforman la nave que quedaba por acertar). Si la posición indicada no corresponde a una parte de alguna nave, se guardará el registro del disparo.

En la interfaz, se debe mostrar con una X los tiros errados y con un O los tiros acertados, además de una bitácora que muestre las incidencias del juego. El diseño de la misma queda a discreción del estudiante, pero debe mostrar todas y cada una de las opciones pedidas.

Implementación

El juego tendrá que ser implementado mediante un servidor (Python) y un programa cliente que también será web (JSP). Por tanto, es necesario que la configuración se realice en red, como el siguiente diagrama:



Para desarrollar este diagrama de red, se utilizarán máquinas virtuales. El orden y configuración de las mismas queda a criterio del estudiante.

Servidor del Juego

Este equipo tendrá el servidor del juego, así como las estructuras necesarias para el desarrollo del juego. Se debe proveer una página de administración. Debe tener una pantalla de login para validar el ingreso de un usuario administrador. Solamente este usuario puede acceder a esta parte del sistema.

Dentro de la funcionalidad, debe tener lo siguiente:

1era Fase

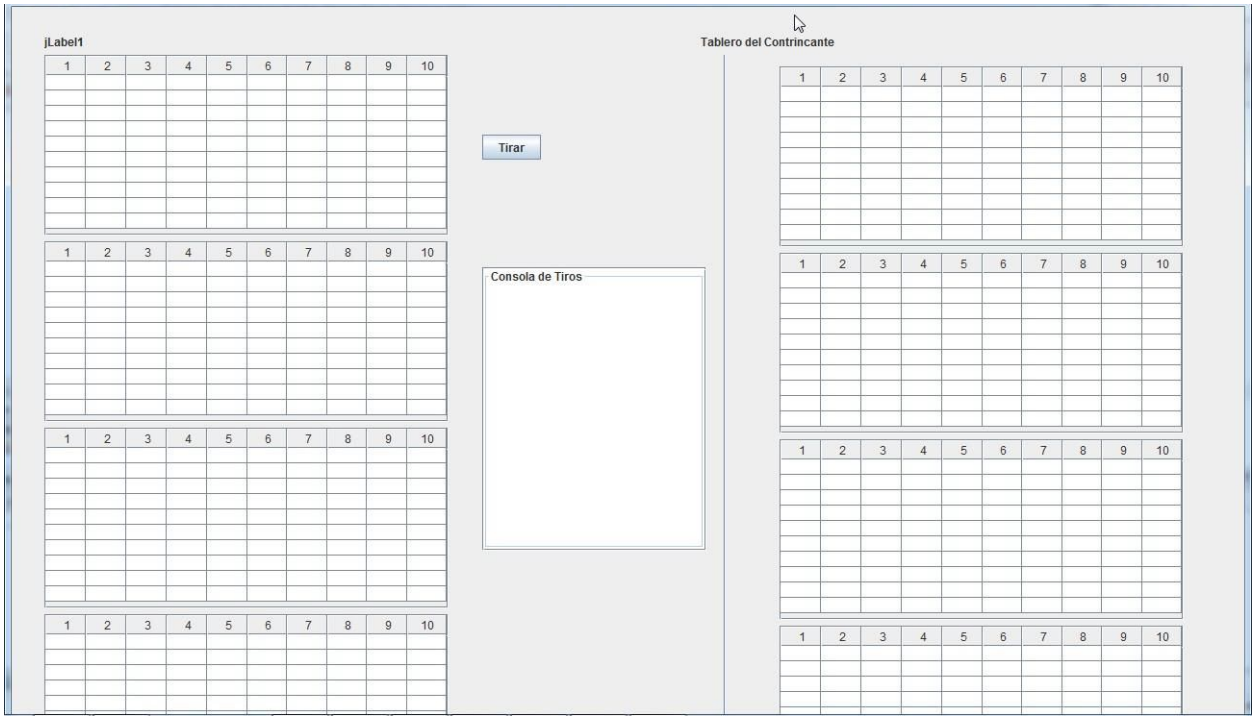
- ABC de usuarios
- ABC de juego
    - Se dan 2 usuarios y se crean estadisticas del juego
- Carga de datos
- Ver reportes

2da Fase

- ABC de contactos
- ABC de movimientos (juego activo)

Cliente

En la aplicación tipo cliente se debe de crear una interfaz de usuario con los elementos necesarios para el juego. Se recomienda el uso de Bootstrap para la interfaz de usuario. Podría ser algo similar a esto:



Se debe tener una consola que informe en donde caen los tiros, si se acertaron o no, así como el progreso del tiempo, si esa variante es la escogida.

Debe realizarse un login en cada cliente, tomando como base los usuarios en el árbol.

Cliente Android

Este cliente solo se pedira en la segunda fase. Este cliente tendrá un login (igual con el árbol de usuarios) y podrá realizar lo siguiente:

- Crear código QR
- Ver todos reportes respectivos de la segunda fase

Estructuras de Datos

Usuarios

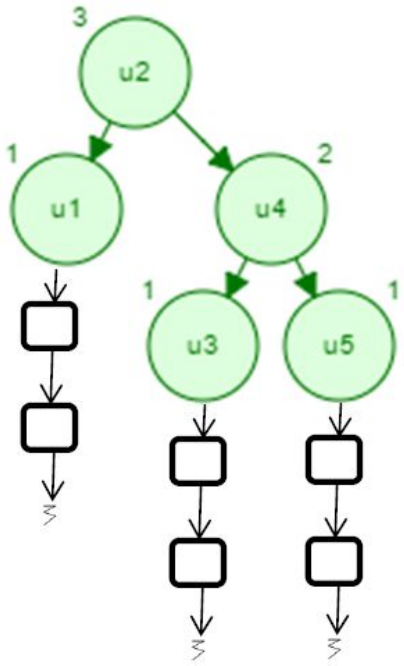
Los usuarios se manejaran con un **árbol binario de búsqueda** por nombre de usuario. Del usuario se tienen que guardar los siguientes datos:

- Nombre de usuario (debe de ser único)
- Contraseña
- Una variable que indica si este está conectado o no
- Apuntador a lista de juegos

Lista de juegos

Es una lista doblemente enlazada que contiene los siguientes datos del juego:

- Nombre de usuario del oponente
- Tiros realizados
- Tiros acertados
- Tiros fallados
- Una bandera que indica si gano o perdio la partida
- Daño recibido



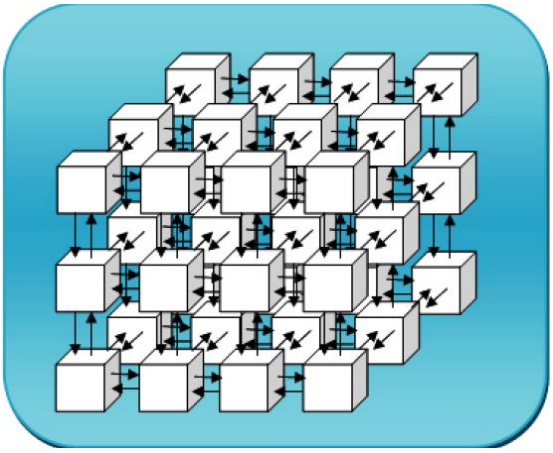
Tablero de juego

El tablero tendrá la forma de una cubo disperso y se manejará una matriz por jugador.

La malla principal de la matriz (la que tiene los encabezados) es la malla que maneja los barcos. De allí se apuntará a las siguientes dimensiones:

- Satelites
- Aviones
- Barcos
- Submarinos

En cada nodo se agregará una bandera que indicará si ya ha sido impactado y además otra bandera que indicará si ya fue hundido.



Historial de Juego

Esta sección es de la fase 2. El historial consiste en guardar cada uno de los tiros realizados en un juego, en un árbol B. Los datos a guardar son los siguientes:

- Coordenada X
- Coordenada Y
- Tipo tiro (ráfaga, uno a uno)
- Resultado (fallo, golpe, eliminación del objetivo)
    - Tipo de nave golpeada, si fuera el caso
- Emisor
- Receptor
- Fecha
- Tiempo restante
    - Si fuera el caso
- Número de tiro

Antes de iniciar el juego, debe preguntarse cuál es el parámetro por el que se debe ordenar el árbol B. Se debe tomar en cuenta que el mismo puede repetirse. Si se diera el caso, el segundo criterio de ordenamiento es el número de tiro.

Contactos

Esta sección es de la fase 2 del proyecto. El usuario, además de tener su lista de juegos, se agregará un árbol AVL de contactos. Estos contactos pueden ser usuarios ya existentes en el sistema o usuarios que no existan. Si el usuario ya existe, el nodo AVL sólo apuntará al nodo del usuario ya existente, en caso contrario, se debe ingresar todos los datos en un nuevo nodo

AVL. Los datos son los mismos del usuario. Entonces, el contenido del nodo AVL sería el siguiente:

- Apuntador a usuario (si ya existe)
- Nickname
- Contraseña

Dispersión de usuarios

La dispersión de usuarios es exclusiva de la segunda fase del proyecto. Teniendo los usuarios en el árbol binario, se tomará todos y cada uno de ellos y se exportarán a una tabla de dispersión hash. Esta tabla hash tendrá los siguientes parámetros:

- Tamaño inicial: 47
- Porcentaje de ocupación máximo: 60%
- Porcentaje de ocupación mínimo: 30%
- Estrategia de resolución de colisiones: Exploración lineal
- Función de dispersión: mitad del cuadrado
- Valor a ingresar en la función de hash: el ascii acumulado del nombre de usuario

Código QR

Esta sección es de la fase 2. Esta aplicación nos debe de dar la opción de generar un código QR que contenga un JSON con todas las estadísticas de un juego que escojamos de la lista de juegos del usuario que haya iniciado sesión.

Tecnologias

Las tecnologías serán las siguientes:

- Primera fase
    - Servidor (Python con Flask)
    - Cliente (JSP)
- Segunda fase
    - Cliente (Android)
        - A través de este nuevo cliente se podrá ver la funcionalidad de la segunda fase, consumiendo los servicios web provistos por el servidor.
    - Servidor (Python con Flask)

Reportes

Todos los reportes serán en Graphviz.

- Primera fase
    - Tablero de juego: Muestra el tablero en su estado actual.
    - Tablero inicial: Muestra cómo estaba el tablero al ser iniciado el juego.

        - Tiros fallados: Muestra el tablero con solamente los disparos mal realizados.
        - Tiros acertados: Muestra el tablero con solamente los tiros acertados.
        - Usuarios: estructura completa con sus listas.
        - Árbol espejo de usuarios (sin sus listas)
        - Mostrar los datos del árbol de usuarios
            - Altura
            - Niveles
            - Cantidad de nodos hoja
            - Cantidad de nodos rama
        - Consultas
            - Top 10 de usuarios con más juegos ganados
            - Top 10 de usuarios con mayor porcentaje de acierto
- Segunda fase
    - Historial de tiros del juego (Árbol B)
    - Árbol de contactos de algún usuario requerido
    - Usuarios (Hash)
    - Consultas
        - Top 10 de usuario con más naves eliminadas
        - Top 10 de usuarios con más contactos
        - Mostrar juego con más tiros realizados para su finalización
        - Mostrar juego con menos tiros realizados para su finalización

Restricciones

Los reportes son esenciales para verificar si se trabajó correctamente la estructura, así que **sin reportes, no se calificará el proyecto**. Si no se puede mostrar el reporte de alguna estructura se anularán los puntos que tengan relación con la estructura, no solo los puntos del reporte.

Observaciones

- Las estructuras deben de ser realizadas por el estudiante. No se puede hacer uso de librerías para reemplazar estructuras del proyecto.
- Lenguaje a utilizar: Python, JSP y Java, cualquier IDE.
- Herramienta de reportes: Graphviz.
- Las gráficas deben ser abiertas por la aplicación y se tienen que hacer los arreglos para que se vean bien, no buscarlas en el sistema. Se penalizará con puntos si no se cumple este requerimiento.
- Proyectos con excepción NULL tendrán penalización de puntos por excepción.
- Se debe de crear un repositorio en GitHub con el nombre de ProyectoJunio2017_#carnet.

- Los documentos deben estar sincronizados don GitHub así mismo teniendo un historial de que trabajo en dicha plataforma
- Forma de entrega: vía GitHub
- Fecha de Entrega:
    - Primera fase: 25 de Junio del 2017 a las 11:59 P.M. (Última sincronización de GitHub).
    - Segunda fase: 15 de Julio del 2017 a las 11:59 P.M. (Última sincronización de GitHub).
- Para la entrega se debe de mandar un correo con el asunto [EDD]EntregaFase1_#Carnet y [EDD]EntregaFase2_#Carnet al correo 201314038@ingenieria.usac.edu.gt cuando se realice la última sincronización en la plataforma Github.
- **COPIAS SERÁN PENALIZADAS con una nota de 0.**
- **COPIAS SERÁN PENALIZADAS conforme al reglamento.**