**NodeJS, WebApp, WebQTServer**

Sistemas Operativos

Kenneth Tavares 237976

**websocket/public/**

**Descripción**: Carpeta principal que contiene todos los recursos accesibles públicamente. **Propósito**: Almacenar los archivos que serán servidos directamente al cliente. **Contenido**: Archivos CSS, JavaScript, imágenes y código fuente.

**websocket/public/css/**

**Descripción**: Carpeta que contiene archivos de estilo CSS.

**Propósito**: Organizar los estilos visuales de la aplicación.

**Contenido**: Archivos de estilo CSS para diferentes componentes de la interfaz.

**websocket/public/img/ (vacío)**

**Descripción**: Carpeta destinada a imágenes.

**Propósito**: Almacenar recursos gráficos como iconos, logos, etc.

**Contenido**: Actualmente vacío.

**websocket/public/js/**

**Descripción**: Carpeta que contiene las librerías y archivos necesarios

**Propósito**: Almacenar scripts y bibliotecas JavaScript utilizados por la aplicación.

**Contenido**: Múltiples archivos JavaScript incluyendo bibliotecas y código personalizado.

**websocket/public/src/**

**Descripción**: Carpeta que contiene código fuente.

**Propósito**: Almacenar archivos de código fuente específicos para la aplicación.

**Contenido**: Archivos de código fuente para diferentes componentes de la aplicación.

**websocket/views/**

**Descripción**: Carpeta que contiene las plantillas EJS (Embedded JavaScript) para las vistas del proyecto.

**Propósito**: Almacenar las plantillas que generan el HTML dinámico enviado al cliente. **Contenido**: Archivos EJS para las diferentes partes de la interfaz de usuario.

**websocket/public/styles.css**

**Propósito**: Define el estilo visual básico de la interfaz de usuario.

**websocket/views/content.ejs**

**Propósito**: Definir el contenido principal de la página, incluyendo el cliente WebSocket y el área de visualización 3D.

**websocket/views/footer.ejs**

**Propósito**: Definir el pie de página y cargar los scripts necesarios para el funcionamiento de la aplicación.

**websocket/views/header.ejs**

**Propósito**: Definir el encabezado de la página, cargar estilos CSS e inicializar el cuerpo de la página.

**websocket/views/index.ejs**

**Propósito**: Archivo principal que incluye todas las demás plantillas EJS para formar la página completa.

**websocket/public/src/dojoInclude.js**

**Propósito**: Carga todos los módulos y componentes necesarios del framework Dojo Toolkit para la construcción de la interfaz de usuario interactiva.

**websocket/public/src/echoWebSckt.js**

**Propósito**: Establece y gestiona la conexión WebSocket con el servidor en el puerto 3349, permitiendo la comunicación en tiempo real.

**Funciones o métodos**:

* debug(message): Muestra mensajes de depuración en un área de texto.
* sendMessage(): Envía el mensaje escrito en el campo de entrada al servidor WebSocket.
* initWebSocket(): Inicializa la conexión WebSocket con el servidor.
* stopWebSocket(): Cierra la conexión WebSocket activa.
* checkSocket(): Verifica el estado actual de la conexión WebSocket.

**websocket/public/src/scripts.js**

**Propósito**: Generar un gráfico 3D interactivo basado en una función matemática personalizada.

**Funciones o métodos**:

* custom(x, y): Función matemática personalizada que define la forma del gráfico 3D.
* drawVisualization(): Función principal que dibuja el gráfico 3D.

**websocket/public/src/socketService.js**

**Propósito**: Procesar los mensajes recibidos desde el servidor y ejecutar las acciones correspondientes.

**Funciones o métodos**:

* socketService(message): Función principal que procesa los mensajes recibidos por WebSocket.
* dateTime(data): Actualiza la fecha y hora en la interfaz.

**websocket/public/js/vis3d.js**

**Descripción**: Archivo JavaScript que contiene la biblioteca de visualización 3D.

**Propósito**: Proporcionar funcionalidades para la creación, manipulación y renderizado de gráficos en 3D.

**Contenido**: Código de la biblioteca de visualización 3D que implementa las clases y métodos necesarios para generar gráficos tridimensionales interactivos.

**websocket/public/js/dojo-release1.17.3/**

**Descripción**: Carpeta que contiene el framework Dojo Toolkit en su versión 1.17.3.

**Propósito**: Proporcionar un conjunto completo de herramientas JavaScript para el desarrollo de aplicaciones web.

**Contenido**: Framework Dojo completo, incluyendo múltiples módulos como dojo (núcleo), dijit (widgets UI), dojox (extensiones) y utils (herramientas de desarrollo). Contiene código JavaScript, CSS, imágenes y otros recursos necesarios para el funcionamiento del framework.

**websocketserver/**

**Descripción**: Carpeta que contiene el código del servidor WebSocket.

**Propósito**: Almacenar el código fuente del servidor que maneja las conexiones WebSocket y proporciona datos al cliente.

**Contenido**: Archivos de cabecera (.h), archivos de implementación (.cpp), archivos de proyecto Qt y otros recursos necesarios para el servidor.

**websocketserver/build/**

**Descripción**: Carpeta generada por el sistema de compilación. **Propósito**: Almacenar archivos generados durante la compilación del proyecto. **Contenido**: Archivos objeto (.o), archivo ejecutable, archivos de configuración y otros archivos generados por el compilador y el sistema de construcción.

**websocketserver/allHeaders.h**

**Propósito**: Proporcionar un archivo de cabecera común que incluye todas las bibliotecas estándar y de Qt necesarias para el proyecto.

**Contenido**:

* Inclusiones de bibliotecas estándar de C++ (iostream, tuple, vector, math, cmath)
* Inclusiones de clases Qt para trabajar con JSON (QJsonArray, QJsonDocument, QJsonObject, QJsonParseError, QJsonValue)
* Declaración del uso del espacio de nombres estándar

**websocketserver/bouncing.h**

**Propósito**: Declarar la clase Bouncing que implementa la simulación física de un objeto rebotando.

**Funciones y métodos**:

* Bouncing(): Constructor de la clase.
* simulation(float xi, float t, float e): Simula el movimiento de un objeto en un punto dado del tiempo y devuelve las coordenadas.
* ball(float xi, float t, float e): Calcula si la pelota está rebotando y sus coordenadas.
* simulation(): Genera las coordenadas completas para toda la simulación.

**Atributos**:

* v0: Velocidad inicial.
* theta: Ángulo de lanzamiento.
* g: Aceleración de la gravedad.
* x\_0, y\_0: Posición inicial.
* dt: Incremento de tiempo para la simulación.

**websocketserver/echoserver.h**

**Propósito**: Definir la clase EchoServer que implementa el servidor WebSocket que maneja las conexiones con los clientes.

**Funciones y métodos**:

* EchoServer(quint16 port, bool debug, QObject \*parent): Constructor que inicializa el servidor en un puerto específico.
* ~EchoServer(): Destructor de la clase.
* onNewConnection(): Maneja nuevas conexiones de clientes.
* processTextMessage(QString message): Procesa mensajes de texto recibidos.
* processBinaryMessage(QByteArray message): Procesa mensajes binarios recibidos.
* socketDisconnected(): Maneja desconexiones de clientes.
* processTimer(): Procesa eventos del temporizador.

**Atributos**:

* timer: Temporizador para enviar actualizaciones periódicas.
* pClient: Puntero al cliente WebSocket activo.
* m\_pWebSocketServer: Puntero al servidor WebSocket.
* m\_clients: Lista de clientes conectados.
* m\_threads: Lista de hilos de ejecución para los clientes.
* m\_debug: Bandera que indica si se debe mostrar información de depuración.

**websocketserver/wsthread.h**

**Propósito**: Definir la clase WsThread que implementa un hilo de ejecución separado para cada cliente WebSocket.

**Funciones y métodos**:

* WsThread(QWebSocket \*pSocket): Constructor que inicializa el hilo con un socket específico.
* run(): Método principal del hilo que se ejecuta cuando se inicia.
* processTextMessage(QString message): Procesa mensajes de texto recibidos en este hilo.
* socketDisconnected(): Maneja la desconexión del cliente asignado a este hilo.
* timerSlot(): Maneja eventos del temporizador en este hilo.

**Atributos**:

* threadConnected: Bandera que indica si el hilo está conectado.
* wsTimer: Puntero al temporizador asociado con este hilo.

**websocketserver/wstimer.h**

**Propósito**: Definir la clase WsTimer que implementa un temporizador para enviar actualizaciones periódicas a los clientes WebSocket.

**Funciones y métodos**:

* WsTimer(QWebSocket \*tSocket, QObject \*parent): Constructor que inicializa el temporizador con un socket específico.
* stop(): Detiene el temporizador.
* timerSlot(): Método que se ejecuta cada vez que el temporizador envía una señal.

**Atributos**:

* timer: Puntero al temporizador Qt.
* pSocket: Puntero al socket WebSocket asociado con este temporizador.

**websocketserver/main.cpp**

**Propósito**: Punto de entrada principal de la aplicación que configura y ejecuta el servidor WebSocket.

**Funciones y métodos**:

* main(int argc, char \*argv[]): Función principal que inicializa la aplicación, configura las opciones de línea de comandos y ejecuta el servidor WebSocket. **Atributos**:
* parser: Analizador de comandos para las opciones de línea de comandos.
* debug: Opción para habilitar la salida de depuración.
* port: Puerto en el que escuchará el servidor (predeterminado: 3349).

**websocketserver/wsthread.cpp**

**Propósito**: Implementa la clase WsThread que maneja conexiones WebSocket individuales en hilos separados.

**Funciones y métodos**:

* WsThread(QWebSocket \*pSocket): Constructor que inicializa el hilo con un socket WebSocket.
* run(): Método principal del hilo que se ejecuta mientras está conectado.
* processTextMessage(QString message): Procesa mensajes de texto recibidos y los devuelve al cliente.
* socketDisconnected(): Maneja la desconexión del cliente WebSocket.
* timerSlot(): Función preparada para eventos temporizados (actualmente vacía). **Atributos**:
* threadConnected: Indica si el hilo está conectado.
* wsTimer: Temporizador para enviar actualizaciones periódicas.

**websocketserver/wstimer.cpp**

**Propósito**: Implementa un temporizador que envía la fecha y hora actual al cliente WebSocket a intervalos regulares.

**Funciones y métodos**:

* WsTimer(QWebSocket \*tSocket, QObject \*parent): Constructor que configura el temporizador.
* stop(): Detiene el temporizador.
* timerSlot(): Función que se ejecuta en cada intervalo del temporizador para enviar la fecha y hora actual.

**Atributos**:

* timer: Objeto QTimer que maneja la temporización.
* pSocket: Socket WebSocket al que se envían los datos.

**websocketserver/bouncing.cpp**

**Propósito**: Implementa la clase Bouncing que simula el movimiento físico de un objeto rebotando.

**Funciones y métodos**:

* Bouncing(): Constructor de la clase que inicializa los parámetros físicos.
* simulation(float xi, float t, float e): Calcula las coordenadas (x, y) en un tiempo específico.
* ball(float xi, float t, float e): Verifica si la pelota está en el aire y calcula sus coordenadas.
* simulation(): Genera las coordenadas completas para toda la simulación de rebotes. **Atributos**:
* v0: Velocidad inicial (30.0).
* theta: Ángulo de lanzamiento (70 grados en radianes).
* g: Aceleración de la gravedad (9.81).
* x\_0, y\_0: Posición inicial (0.0, 0.0).
* dt: Incremento de tiempo para la simulación (0.01).

**websocketserver/echoserver.cpp**

**Propósito**: Implementa el servidor WebSocket que acepta conexiones y crea hilos para manejar cada cliente.

**Funciones y métodos**:

* EchoServer(quint16 port, bool debug, QObject \*parent): Constructor que inicializa y configura el servidor WebSocket.
* ~EchoServer(): Destructor que limpia los recursos del servidor.
* onNewConnection(): Maneja nuevas conexiones entrantes creando un hilo para cada una.
* processTextMessage(QString message): Procesa mensajes de texto recibidos.
* processBinaryMessage(QByteArray message): Procesa mensajes binarios recibidos.
* socketDisconnected(): Maneja la desconexión de clientes.
* processTimer(): Envía la fecha y hora actual a los clientes cuando se activa el temporizador. **Atributos**:
* m\_pWebSocketServer: Servidor WebSocket principal.
* m\_clients: Lista de clientes conectados.
* m\_debug: Indica si el modo de depuración está habilitado.
* pClient: Cliente WebSocket actual.
* timer: Temporizador para enviar actualizaciones periódicas.