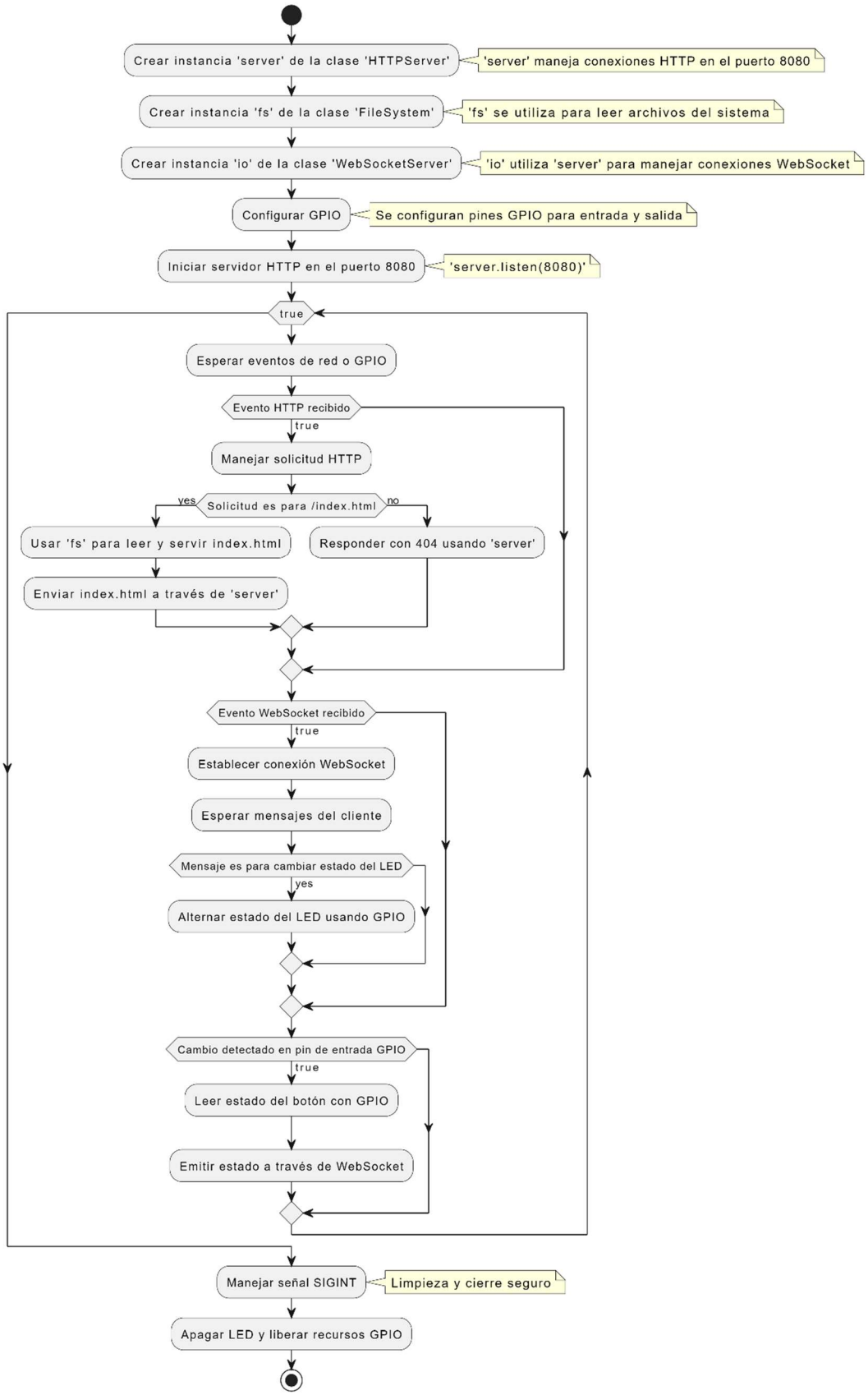
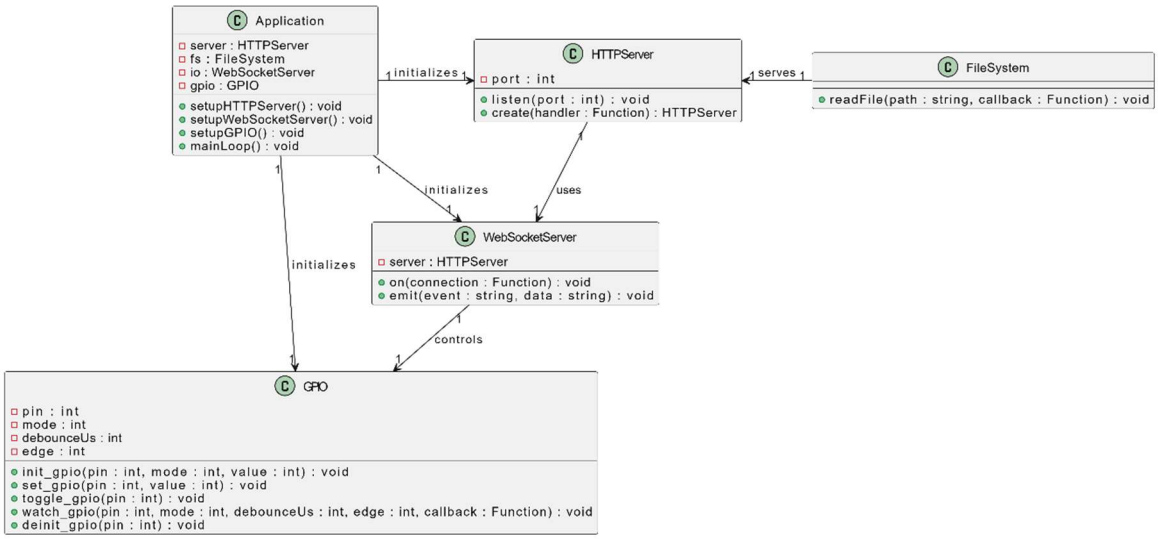
Nodetest5

```
// Crea un servidor HTTP que escucha en el puerto 8080
const http = require('http').createServer(handler);

// Utiliza el módulo del sistema de archivos para leer y servir archivos HTML
var fs = require('fs');

// Establece una conexión WebSocket sobre el servidor HTTP
var io = require('socket.io')(http);

// Configura los pines GPIO para control de entrada y salida
const gpio = require("@iiot2k/gpiox");

// Función para manejar solicitudes HTTP y servir la página web
function handler (req, res) {
  fs.readFile(__dirname + '/public/index.html', function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}

// Gestiona eventos de WebSocket para controlar el LED y leer el estado del botón
io.sockets.on('connection', function (socket) {
  // Monitorea el estado del botón y emite cambios a los clientes
  console.log("conectado");
  var state = gpio.watch_gpio(INPUT_PIN, gpio.GPIO_MODE_INPUT_PULLDOWN, DEBOUNCE_US, gpio.GPIO_EDGE_BOTH,
  (state, edge) => {
    console.log("port", INPUT_PIN, "state", state, "edge", edge);
    socket.emit('light', state);
  });

  // Responde a mensajes del cliente para alternar el LED
  socket.on('light', function(data) {
    gpio.toggle_gpio(OUTPUT_PIN);
  });
});

// Limpieza de recursos GPIO cuando se interrumpe el proceso
process.on('SIGINT', function () {
  gpio.set_gpio(25, 0);
  gpio.deinit_gpio(25);
  gpio.deinit_gpio(INPUT_PIN);
  process.exit();
});
```


Crear instancia 'server' de la clase 'HTTPServer'
'server' maneja conexiones HTTP en el puerto 8080
Crear instancia 'fs' de la clase 'FileSystem'
'fs' se utiliza para leer archivos del sistema
Crear instancia 'io' de la clase 'WebSocketServer'
'io' utiliza 'server' para manejar conexiones WebSocket
Configurar GPIO
Se configuran pines GPIO para entrada y salida
Iniciar servidor HTTP en el puerto 8080
'server.listen(8080)'
true
Esperar eventos de red o GPIO
Evento HTTP recibido
true
Manejar solicitud HTTP
yes
Solicitud es para /index.html
no
Usar 'fs' para leer y servir index.html
Responder con 404 usando 'server'
Enviar index.html a través de 'server'
Evento WebSocket recibido
true
Establecer conexión WebSocket
Esperar mensajes del cliente
Mensaje es para cambiar estado del LED
yes
Alternar estado del LED usando GPIO
Cambio detectado en pin de entrada GPIO
true
Leer estado del botón con GPIO
Emitir estado a través de WebSocket
Manejar señal SIGINT
Limpieza y cierre seguro
Apagar LED y liberar recursos GPIO


Application
server : HTTPServer
fs : FileSystem
io : WebSocketServer
gpio : GPIO
setupHTTPServer() : void
setupWebSocketServer() : void
setupGPIO() : void
mainLoop() : void
HTTPServer
port : int
listen(port : int) : void
create(handler : Function) : HTTPServer
FileSystem
readFile(path : string, callback : Function) : void
WebSocketServer
server : HTTPServer
on(connection : Function) : void
emit(event : string, data : string) : void
GPIO
pin : int
mode : int
debounceUs : int
edge : int
init_gpio(pin : int, mode : int, value : int) : void
set_gpio(pin : int, value : int) : void
toggle_gpio(pin : int) : void
watch_gpio(pin : int, mode : int, debounceUs : int, edge : int, callback : Function) : void
deinit_gpio(pin : int) : void
initializes
serves
initializes
initializes
uses
controls
1