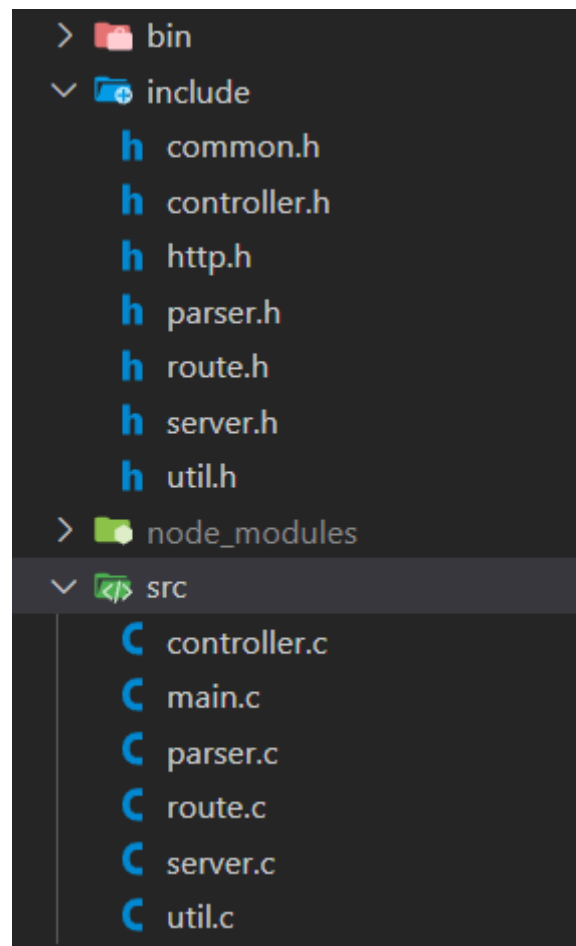


SisterJS

(disclaimer ini asli gangerti bener atau enggak, nikmati aja kode ampas saya)

NOTE: ini dibuat di ubuntu jadi keknya bakalan error kalo di windows ehe

Files



Kurang lebih begini terdapat 6 file .c dari source code yang mana

server.c: main server

util.c: isinya utilitas yang sering dipake

parser.c: parser buat

- Application/json
- Text/plain
- Application/x-www-form-urlencoded

route.c: untuk routing

controller.c: untuk handling request

- GET
- POST
- PUT
- DELETE

contoh dari controller tiap request

```
void GET_example(int client_socket, HttpRequest *req, HttpResponse* res) {  
    handle_request(client_socket, req, res);  
}
```

```
void POST_example(int client_socket, HttpRequest* req, HttpResponse* res) {  
    handle_request(client_socket, req, res);  
}
```

```
void PUT_example(int client_socket, HttpRequest *req, HttpResponse* res) {  
    handle_request(client_socket, req, res);  
}
```

```
void DELETE_example(int client_socket, HttpRequest *req, HttpResponse* res) {  
    handle_request(client_socket, req, res);  
}
```

Fungsi untuk handle request-nya

```

void handle_request(int client_socket, HttpRequest* req, HttpResponse* res){
    bool is_valid = true;

    memset(res->response, 0, sizeof(res->response));
    if (req->content_type[0] == '\0' && (strcmp(req->method, "POST") == 0 || strcmp(req->method, "PUT")
== 0)) {
        sprintf(res->response, "{\"status\": \"error\", \"message\": \"Content-Type header
missing\"}");
        send_response(client_socket, "400 Bad Request", "application/json", res->response);
        return;
    }

    if (strcmp(req->method, "GET") == 0) {
        if (strlen(req->params) != 0) {
            snprintf(res->response, sizeof(res->response), "GET with params: %.200s", req->params);
        } else {
            snprintf(res->response, sizeof(res->response), "GET Only");
        }
        send_response(client_socket, "200 OK", "text/plain", res->response);
        return;
    }

    if (strcmp(req->method, "POST") == 0 || strcmp(req->method, "PUT") == 0){
        parse_body(req->content_type, req->body, res->keys, res->values, &res->total_data);

        if (!is_valid) {
            strcat(res->status, "Unsupported content type");
            sprintf(res->response, "{\"status\": \"error\", \"message\": \"Unsupported content
type\"}");
            send_response(client_socket, "400 Bad Request", "application/json", res->response);
            return;
        }
        printf("method sebelum http: %s\n", req->method);

        generate_response_http(req, res);
        send_response(client_socket, "200 OK", "application/json", res->response);
        memset(res->response, 0, sizeof(res->response));
        return;
    }

    if (strcmp(req->method, "DELETE") == 0){
        if (res->total_data > 0) {
            strcat(res->response, "{\"status\": \"deleted\", \"data\": {\"");
            for (int i = 0; i < res->total_data; i++) {
                int pair_length = strlen(res->keys[i]) + strlen(res->values[i]) + 6;
                char* pair = malloc(pair_length * sizeof(char));
                sprintf(pair, "{\"%s\": \"%s\"}", res->keys[i], res->values[i]);
                strcat(res->response, pair);
                if (i < res->total_data - 1) {
                    strcat(res->response, ", ");
                }
                free(pair);
            }
            strcat(res->response, "}}");
        } else {
            strcat(res->response, "{\"status\": \"deleted\", \"data\": {}}");
        }
        send_response(client_socket, "200 OK", "application/json", res->response);
        return;
    }

    sprintf(res->response, "{\"status\": \"error\", \"message\": \"Unsupported HTTP method\"}");
    send_response(client_socket, "405 Method Not Allowed", "application/json", res->response);
}

```

Framework yang kubuat ini cuman bisa di-port 8080 gabisa diganti lmao,
yaudah biarin aja yang penting jalan

“if it's work don't touch it”

Cuplikan kode client, tapi pake javascript ngehe.

```
const http = require('http');

function sendRequest(options, postData = null) {
  return new Promise((resolve, reject) => {
    const req = http.request(options, (res) => {
      let data = '';

      res.on('data', (chunk) => {
        data += chunk;
      });

      res.on('end', () => {
        console.log('Response received:\n${data}');
        resolve(data);
      });
    });

    req.on('error', (e) => {
      console.error('Problem with request: ${e.message}');
      reject(e);
    });

    if (postData) {
      req.write(postData);
    }

    req.end();
  });
}

async function main() {
  // Example GET request
  const getOptions = {
    hostname: 'localhost',
    port: 8080,
    path: '/nilai-akhir',
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
    },
  };

  console.log('Sending GET request...');
  await sendRequest(getOptions);

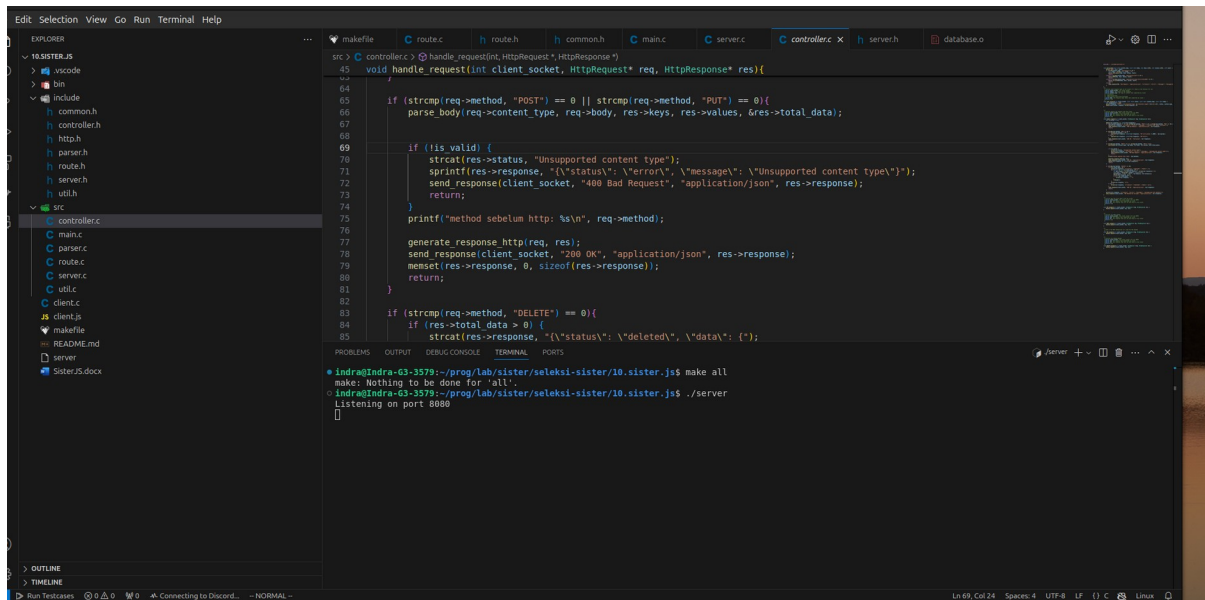
  // Example POST request
  const postData = JSON.stringify({ bjr: 'okeh' });
  const postOptions = {
    hostname: 'localhost',
    port: 8080,
    path: '/submit',
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Content-Length': Buffer.byteLength(postData),
    },
  };

  console.log('Sending POST request...');
  await sendRequest(postOptions, postData);
}

main().catch((err) => {
  console.error('Error: ${err.message}');
});
```

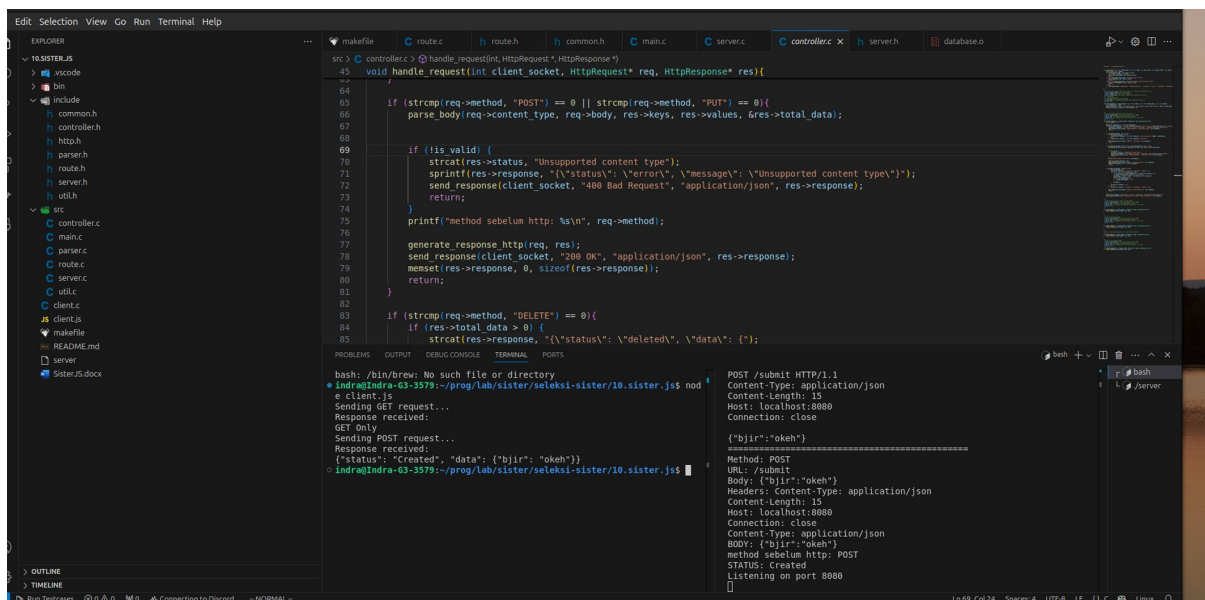
Source code yang saya buat ini tidak menggunakan database. Hanya dibuat untuk untuk terhubung saja, sumpah saya skill issue banget. Udah skill issue, tapi ngide ngerjain pake C, aneh emang orang gila satu ini (ya, itu saya).

Ini screenshot saat program jalan, langsung semuanya PUT, DELETE, GET, POST.



The screenshot shows the Visual Studio Code editor with a C project open. The Explorer panel on the left shows the file structure, including folders like 'src' and 'include'. The main editor window displays the source code for 'controller.c', which implements a simple HTTP server. The code includes headers for 'http.h', 'common.h', and 'server.h'. It defines a 'handle_request' function that processes incoming requests. The function checks the request method (GET, POST, PUT, DELETE) and handles them accordingly. For POST, PUT, and DELETE, it checks if the request body is valid and sends a response. For GET, it sends a response with the data from the request body. The main function calls 'handle_request' and listens on port 8080.

```
src> controller.c> handle_request(int client_socket, HttpRequest* req, HttpResponse* res){
45 void handle_request(int client_socket, HttpRequest* req, HttpResponse* res){
46
47     if (strcmp(req->method, "POST") == 0 || strcmp(req->method, "PUT") == 0){
48         parse_body(req->content_type, req->body, res->keys, res->values, &res->total_data);
49     }
50
51     if (!is_valid){
52         strcat(res->status, "Unsupported content type");
53         sprintf(res->response, "{\n\"status\": \"error\", \"message\": \"Unsupported content type\"}");
54         send_response(client_socket, "400 Bad Request", "application/json", res->response);
55         return;
56     }
57
58     printf("method sebelum http: %s\n", req->method);
59
60     generate_response_http(req, res);
61     send_response(client_socket, "200 OK", "application/json", res->response);
62     memset(res->response, 0, sizeof(res->response));
63     return;
64 }
65
66 if (strcmp(req->method, "DELETE") == 0){
67     if (res->total_data > 0){
68         strcat(res->response, "{\n\"status\": \"deleted\", \"data\": \"\"}");
69     }
70 }
71
72 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
73
74 Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$ make all
75 make: Nothing to be done for 'all'.
76 Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$ ./server
77 Listening on port 8080
78
```



The screenshot shows the Visual Studio Code editor with the same C project open. The Explorer panel on the left shows the file structure. The main editor window displays the source code for 'controller.c'. The terminal window at the bottom shows the output of the program, including the message 'Listening on port 8080'. A network traffic capture window is also visible, showing a POST request to 'http://localhost:8080' with a body of '{\"bjir\": \"okeh\"}' and a response of '{\"status\": \"Created\", \"data\": {\"bjir\": \"okeh\"}}'.

```
bash: /bin/brew: No such file or directory
Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$ nod
e client.js
Sending GET request...
Response received:
GET Only
Sending POST request...
Response received:
{"status": "Created", "data": {"bjir": "okeh"}}
Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$

POST /submit HTTP/1.1
Content-Type: application/json
Content-Length: 15
Host: localhost:8080
Connection: close

{"bjir": "okeh"}

Method: POST
URL: /submit
Body: {"bjir": "okeh"}
Headers: Content-Type: application/json
Content-Length: 15
Host: localhost:8080
Connection: close
Content-Type: application/json
BODY: {"bjir": "okeh"}
method sebelum http: POST
STATUS: Created
Listening on port 8080
```

This screenshot shows the VS Code editor with the `client.js` file open. The file contains an `async function main()` that sends a PUT request, followed by a DELETE request. The terminal shows the output of these requests, including status codes and response data. The Explorer panel on the left shows the project structure, including `src` and `client.js`.

```
117 sister.js
118
119 include
120   h common.h
121   h controller.h
122   h http.h
123   h parser.h
124   h route.h
125   h server.h
126   h util.h
127
128 src
129   C controller.c
130   C main.c
131   C parser.c
132   C route.c
133   C server.c
134   C util.c
135   JS client.js
136
137 makefile
138 README.md
139 server
140 SisterJS.docx
```

```
1  async function main() {
2
3  };
4
5  console.log('Sending PUT request...');
6  await sendRequest(putOptions, putData);
7
8  // Example DELETE request
9  const deleteOptions = {
10    hostname: 'localhost',
11    port: 8080,
12    path: '/delete',
13    method: 'DELETE',
14    headers: {
15      'Content-Type': 'application/json',
16    },
17  };
18  console.log('Sending DELETE request...');
19  await sendRequest(deleteOptions);
20
21  main().catch((err) => {
22    console.error('Error: ${err.message}');
23  });
24 }
```

```
1  Sending GET request...
2  Response received:
3  GET Only
4  Sending POST request...
5  Response received:
6  {"status": "Created", "data": {"bjir": "okeh"}}
7  e client.js
8  Sending GET request...
9  Response received:
10 GET Only
11 Sending POST request...
12 Response received:
13 {"status": "Created", "data": {"bjir": "okeh"}}
14 Sending PUT request...
15 Response received:
16 {"status": "Submitted", "data": {"bjir": "updated"}}
17 Sending DELETE request...
18 Response received:
19 {"status": "deleted", "data": {}}
20 e client.js
21 Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$
```

```
1  Request received:
2  DELETE /delete HTTP/1.1
3  Content-Type: application/json
4  Host: localhost:8080
5  Connection: close
6
7  action: close
8
9  {"bjir": "updated"}
10 =====
11 Method: DELETE
12 URL: /delete
13 Body: action: close
14
15 {"bjir": "updated"}
16 Headers: Content-Type: application/json
17 Host: localhost:8080
18 Connection: close
19 Content-Type: application/json
20 Listening on port 8080
```

This screenshot shows the VS Code editor with the `main.c` file open. The file contains a `main` function that sets up a simple HTTP server using `libmicrohttpd`. The terminal shows the output of the server, including status codes and response data. The Explorer panel on the left shows the project structure, including `src` and `main.c`.

```
1  src > C main.c > ...
2  #include "../include/server.h"
3
4  void random_handler(int client_socket, const char* body, const char* content_type) {
5    char response[MAX] = "Random Handler";
6    send_response(client_socket, "200 OK", "text/plain", response);
7  }
8
9  int main(int argc, char const* argv[]) {
10
11    add_route("GET", "/LMAO", (void*) random_handler);
12    start_server();
13    return 0;
14  }
15
16
```

```
1  e client.js
2  Sending GET request...
3  Response received:
4  GET Only
5  Sending POST request...
6  Response received:
7  {"status": "Created", "data": {"bjir": "okeh"}}
8  e client.js
9  Sending GET request...
10 Response received:
11 {"status": "Submitted", "data": {"bjir": "updated"}}
12 Sending DELETE request...
13 Response received:
14 {"status": "deleted", "data": {}}
15 e client.js
16 Indra@Indra-G3-3579:~/prog/lab/sister/seleksi-sister/10.sister.js$
```

```
1  {"bjir": "updated"}
2  =====
3  Method: DELETE
4  URL: /delete
5  Body: action: close
6
7  {"bjir": "updated"}
8  Headers: Content-Type: application/json
9  Host: localhost:8080
10 Connection: close
11 Content-Type: application/json
12 Listening on port 8080
```

Sekian terima kasih, dan mohon maaf kalau ini jelek karena saya skill issue