PROJETO FINAL
SERVIDOR WEB MULTITHREADED

OLÁ

JORISMAR BARBOSA MEIRA

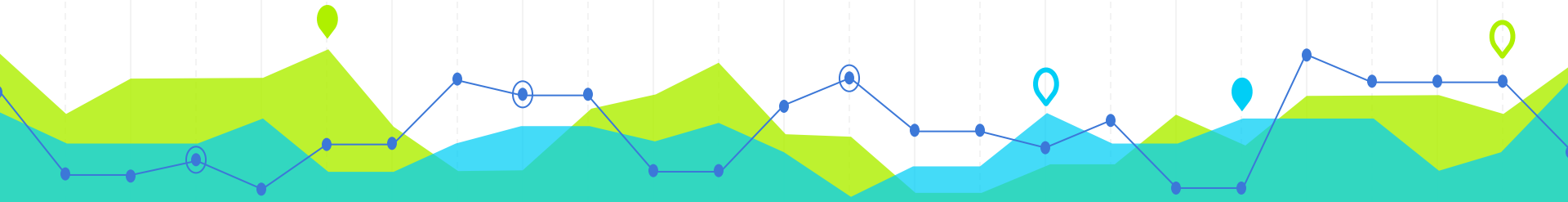
jorismar.barbosa@lavid.ufpb.br

REQUISITOS EXIGIDOS

- Tratar requisições GET
- Versão 1.0 do HTTP.
- Processar múltiplas requisições de serviços em paralelo.
- Identificar os tipos dos arquivos.
- Reportar erro de página não encontrada.

ESTRUTURA DA IMPLEMENTAÇÃO DO SERVIDOR

- Main
- Auxiliares
  - Classe Socket
  - Utils.
- Classe Webserver;
- Classe HTTP;



Implementação 1

Classes auxiliares e principal

MAIN

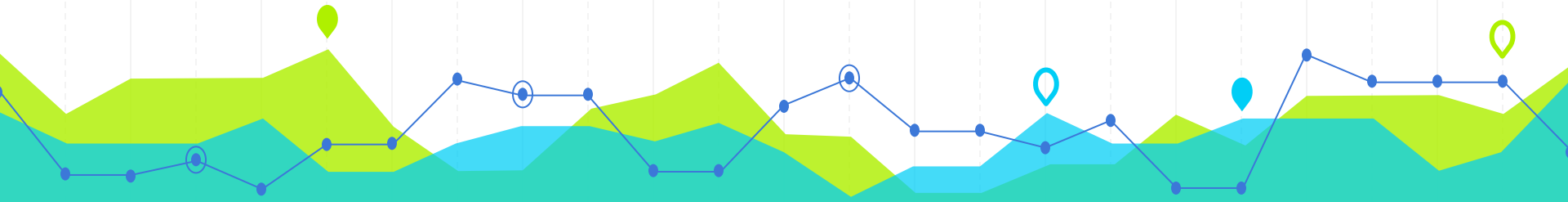
```
#include <iostream>     // cout
#include "webserver.h"

int main(int argc, char *argv[]) {
    try {
        std::string::size_type sz;
        int port = std::stoi(argv[1], &sz);

        Webserver * server = new Webserver(port, "site");

        server->start();
    } catch(...) {
        std::cout << "[ERROR] Invalid port entry." << std::endl;
    }

    return 0;
}
```

UTILS

```
#include <ctime>    // time_t
#include <string>   // strings
#include <iostream> // cout
#include <fstream>  // ifstream

#define CHECK_ERR(cond, msg, ret) {
                ...
}


typedef char t_byte;
typedef int  t_socket;


std::string getDate(std::string format);


int readFile(std::string path, t_byte ** buffer);
```

SOCKET

```
#include <iostream>    // cout
#include <cstring>     // memset
#include <signal.h>    // SIGPIPE, SIG_IGN
#include <unistd.h>    // socket
#include <arpa/inet.h> // sockaddr
#include "util.h"

        ...

Socket();
Socket(int port);
int Bind();
int Listen(int backlog);
t_socket Accept();
int Connect(std::string ip, int port);
int Close();
static int readFrom(t_socket socket, t_byte * buffer, size_t size, int timeout);
static int sendTo(t_socket socket, t_byte * buffer, size_t size);
static int Close(t_socket socket);
```

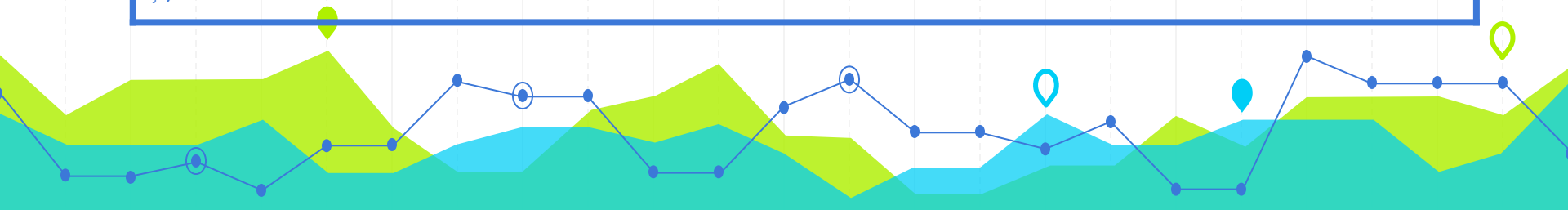


Implementação 2

Classe Webserver

WEBSERVER (HEADER)

```
#include <string>
#include <thread>
#include "socket.h"
#include "http.h"
#include "util.h"

class Webserver {
    private:
        Socket*     socket;         // TCP Socket for server
        std::string path;           // Path of the Website files
        int         port;           // Port used for the server
        bool        alive;          // Flag to keep the server alive.
    public:
        Webserver(int port, std::string path);
        virtual ~Webserver();
        void start();
        void stop();
        void startClient(t_socket cl);
        void setPort(int port);
        int getPort();
};
```

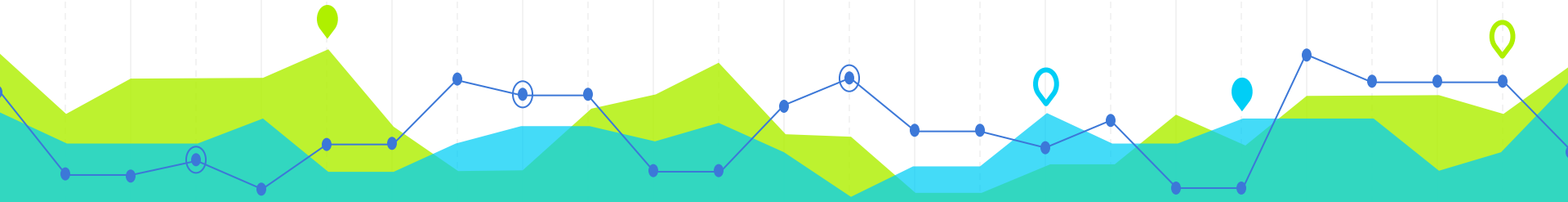


WEBSERVER::START()

```
void Webserver::start() {
    this->socket = new Socket(this->port);
    if(this->socket->Bind() >= 0) {                 // Bind the port
        if(this->socket->Listen(10) >= 0) {         // Listen the port
                t_socket client;                    // Socket to new client connection
                this->alive = true;
                std::cout << "[INFO] Server running on port: "<< this->port << std::endl;
                while(alive) {
                     client = this->socket->Accept(); // Wait for client connection
                     if(client >= 0) {
                          // Start the client thread
                          std::thread cl([=](){ Webserver::startClient(client); });
                          cl.detach();
                     }
                }
                std::cout << "[INFO] Session closed!" << std::endl;
          } else std::cout << "[ERROR] Webserver listen error" << std::endl;
    } else std::cout << "[ERROR] Webserver bind error" << std::endl;
    this->socket->CloseServer(); // Close server socket
}
```

WEBSERVER::STARTCLIENT()

```
void Webserver::startClient(t_socket cl) {
     Http   * http   = new Http();
     t_byte * buffer = new t_byte[1024];
     t_byte * file   = NULL;

     Socket::readFrom(cl, buffer, 1024, 5);
     http->processRequest(buffer);

     int size = readFile(this->path + http->getReqstedFile(), &file);

     if(size > 0) {
          http->createResponseHeader(size, Http::Status::OK);
          http->createBinaryPacket(file, size);
     } else {
          http->createResponseHeader(0, Http::Status::NOT_FOUND);
          http->createBinaryPacket(NULL, 0);
     }

     Socket::sendTo(cl, http->getBinaryPacket(), http->getBinarySize());

     delete[] buffer;
     if(file != NULL) free(file);

    Socket::Close(cl);
}
```



Implementação 3

Classe HTTP

HTTP (HEADER)

```
#include <string>
#include <cstring>
#include <stdexcept>
#include "util.h"

class Http {
    private:
        int         begin_range;     // Requested content begin range.
        int         end_range;       // Requested content end range.
        size_t      buffer_size;     // Buffer size
        t_byte *    buffer;          // Buffer for binary packet generated.
        std::string reqst_file;      // Requested filename.
        std::string reqst_filetype;  // Requested file type.
        std::string server_name;     // Server name
        std::string header;          // Container generated header.

        std::string getfield(std::string src, std::string mark, char end);

        . . .
```

HTTP (HEADER)

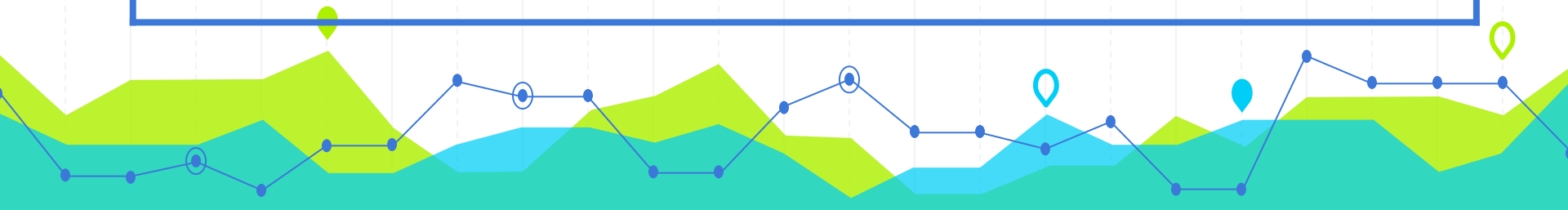
```
        . . .

    public:
        enum Status { OK = 200, PARTIAL_CONTENT = 206, NOT_FOUND = 404 };

        Http();
        virtual ~Http();

        void processRequest(t_byte* header);
        std::string createResponseHeader(size_t filelen, int status_code);
        t_byte* createBinaryPacket(t_byte * file_bin, size_t file_size);

        std::string getReqstedFile();
        std::string getHeader();
        size_t getBinarySize();
        t_byte * getBinaryPacket();

        void setServerName(std::string name);
};
```

HTTP::PROCESSREQUEST

```
void Http::processRequest(t_byte * header) {
    std::string msg(header);                          // Convert bytes to a string

    if(!(msg.find("GET") != std::string::npos))       // Check if is a GET Request
        return;

    this->reqst_file     = this->getfield(msg, "GET ", ' ');

    if(!this->reqst_file.compare("/")) {
        this->reqst_file = "/index.html";
        this->reqst_filetype = "text/html; charset=UTF-8";
    } else {
        std::string filetype = this->reqst_file.substr(
             this->reqst_file.rfind(
                  ".",
                  this->reqst_file.length() - 1
             ) + 1,
             this->reqst_file.length() - 1
        );

    . . .
```

HTTP::PROCESSREQUEST

```
if(!filetype.compare("html") || !filetype.compare("htm"))
    this->reqst_filetype = "text/html; charset=UTF-8";
else if(!filetype.compare("jpg"))
    this->reqst_filetype = "image/jpg";
else if(!filetype.compare("png"))
    this->reqst_filetype = "image/png";
else if(!filetype.compare("gif"))
    this->reqst_filetype = "image/gif";
else if(!filetype.compare("ico"))

     . . .

else if(!filetype.compare("js"))
    this->reqst_filetype = "application/javascript; charset=UTF-8";
else if(!filetype.compare("css"))
    this->reqst_filetype = "text/css";
else
    this->reqst_filetype = "application/octet-stream";

. . .
```



HTTP::PROCESSREQUEST

```
        . . .

    }

    this->begin_range = -1;

    std::string aux = "";
    std::string::size_type sz;

    aux = this->getfield(msg, "Range: bytes=", '-');

    if(aux.length() > 0) {
        try {
            this->begin_range = std::stoi(aux, &sz);
        } catch(...) {
            this->begin_range = -1;
        }
    }
}
```

HTTP::CREATERESPONSEHEADER()

```
std::string Http::createResponseHeader(size_t filelen, int status_code) {
    std::string status          = "";
    std::string connection      = "";
    std::string filesize        = std::to_string(filelen);
    std::string content_length  = filesize;
    std::string content_range   = "";
    std::string custom_data     = "";
    std::string http_version    = "HTTP/1.0";
    std::string EOL             = "\r\n";
    std::string options         = "";

     . . .
```

HTTP::CREATERESPONSEHEADER()

```
    . . .

    if (status_code == Http::Status::OK && this->begin_range >= 0) {

        status_code    = Http::Status::PARTIAL_CONTENT;
        status         = "206 Partial Content";

        content_length = std::to_string(filelen - this->begin_range);
        content_range  = std::to_string(this->begin_range) + "-" + std::to_string
(filelen - 1) + "/" + filesize;

        connection     = "keep-alive";

    }

    . . .
```

HTTP::CREATERESPONSEHEADER()

```
. . .

if(this->begin_range < 0)
     this->begin_range = 0;

if (status_code == Http::Status::OK) {
     status      = "200 OK";
     connection = "keep-alive";
}

. . .
```

HTTP::CREATERESPONSEHEADER()

```
        . . .

else if(status_code == Http::Status::NOT_FOUND) {
        status      = "404 Not Found";
        connection = "close";

        custom_data = "\
                <html>\
                        <head><title>404 Not Found</title></head>\
                        <body>\

                                . . .

                        </body>\
                </html>\
        ";

        content_length = std::to_string(custom_data.length());
        this->reqst_filetype = "text/html; charset=UTF-8";
}

. . .
```

HTTP::CREATERESPONSEHEADER()

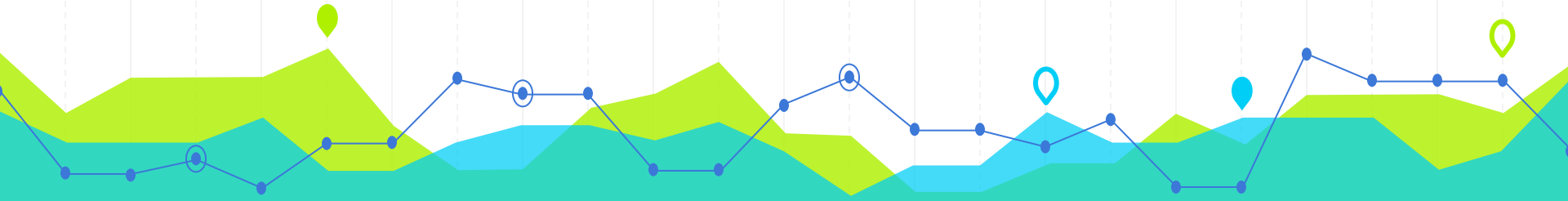
```
    . . .

    options += "Access-Control-Allow-Origin: *" + EOL;
    options += "Access-Control-Allow-Headers: X-Requested-With, Content-Type, X-
Codingpedia, X-HTTP-Method-Override" + EOL;
    options += "x-content-type-options: nosniff" + EOL;

    . . .
```

HTTP::CREATERESPONSEHEADER()

```
    . . .

    this->header += http_version + " " + status + EOL;

    if(status_code != Http::Status::NOT_FOUND)
         this->header += "Accept-Ranges: bytes" + EOL;

    if(status_code == Http::Status::PARTIAL_CONTENT)
         this->header += "Content-Range: bytes " + content_range + EOL;

    this->header += "Cache-Control: public, max-age=0" + EOL;
    this->header += "Content-Type: " + this->reqst_filetype + EOL;
    this->header += "Content-Length: " + content_length + EOL;
    this->header += "Connection: " + connection + EOL;
    this->header += options;
    this->header += "Date: " + getDate("%a, %d %b %Y %T %Z") + EOL;
    this->header += "X-Powered-By: " + this->server_name + EOL;
    this->header += EOL;
    this->header += custom_data;

    return this->header;
}
```

HTTP::CREATEBINARYPACKET()

```
t_byte * Http::createBinaryPacket(t_byte * file_bin, size_t file_size) {
     if(this->header.length() < 1) return NULL;

    size_t data_size = 0;

    if(file_bin == NULL) {
        file_size = 0;
    } else data_size = file_size - this->begin_range;

    this->buffer_size = this->header.length() + data_size;

    this->buffer = new t_byte[this->buffer_size];

    memcpy(this->buffer, this->header.c_str(), this->header.length());

    if(data_size  > 0)
        memcpy(this->buffer + this->header.length(), file_bin + this->begin_range,
data_size);

    return this->buffer;
}
```



Resultados

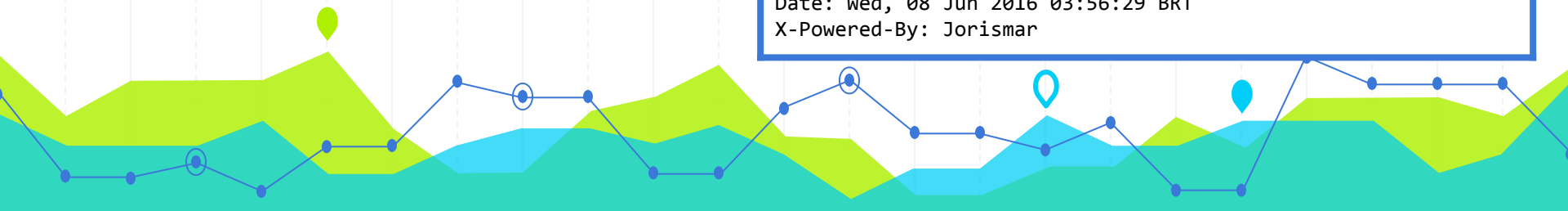
Requisição/Resposta

4

ARQUIVOS SUPORTADOS PELO SERVIDOR

```
GET /ufpb_logo.png HTTP/1.1
Host: 192.168.77.132:8080
Connection: keep-alive
Cache-Control: max-age=0
Save-Data: on
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79
Safari/537.36
Accept: image/webp,image/*,*/*;q=0.8
DNT: 1
Referer: http://192.168.77.132:8080/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

```
HTTP/1.0 200 OK
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Content-Type: image/png
Content-Length: 696941
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With, Content-Type,
X-Codingpedia, X-HTTP-Method-Override
x-content-type-options: nosniff
Date: Wed, 08 Jun 2016 03:56:29 BRT
X-Powered-By: Jorismar
```



ARQUIVOS NÃO SUPORTADOS PELO SERVIDOR

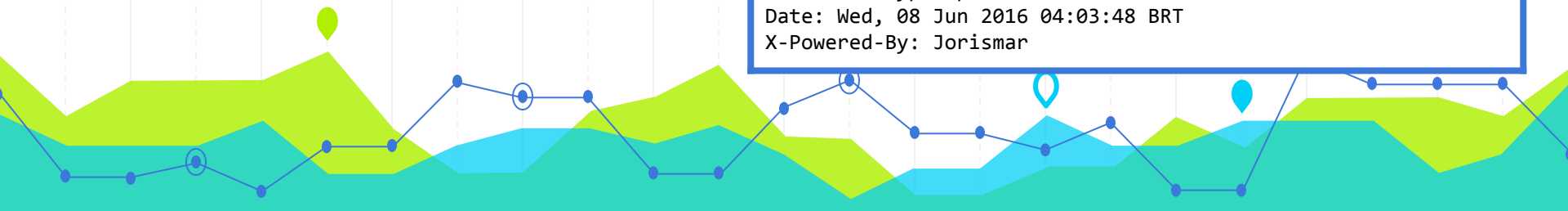
```
GET /devreport.pdf HTTP/1.1
Host: 192.168.77.132:8080
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Save-Data: on
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
DNT: 1
Referer: http://192.168.77.132:8080/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

```
HTTP/1.0 200 OK
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Content-Type: application/octet-stream
Content-Length: 323627
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With, Content-Type,
X-Codingpedia, X-HTTP-Method-Override
x-content-type-options: nosniff
Date: Wed, 08 Jun 2016 04:04:39 BRT
X-Powered-By: Jorismar
```

ARQUIVOS COM ENVIO PARCIAL (PARTIAL CONTENT)

```
GET /video.mp4 HTTP/1.1
Host: 192.168.77.132:8080
Connection: keep-alive
Accept-Encoding: identity;q=1, *;q=0
Save-Data: on
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79
Safari/537.36
Accept: */*
DNT: 1
Referer: http://192.168.77.132:8080/
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
Range: bytes=0-
```

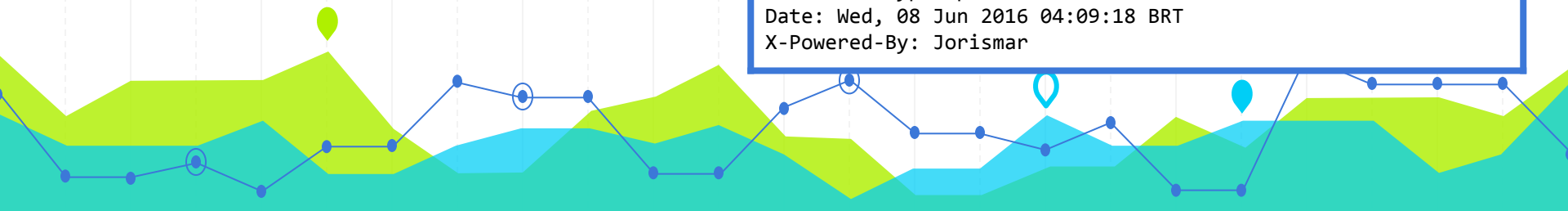
```
HTTP/1.0 206 Partial Content
Accept-Ranges: bytes
Content-Range: bytes 0-45349551/45349552
Cache-Control: public, max-age=0
Content-Type: video/mp4
Content-Length: 45349552
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With, Content-Type,
X-Codingpedia, X-HTTP-Method-Override
x-content-type-options: nosniff
Date: Wed, 08 Jun 2016 04:03:48 BRT
X-Powered-By: Jorismar
```



ARQUIVOS COM ENVIO PARCIAL (PARTIAL CONTENT)

```
GET /video.mp4 HTTP/1.1
Host: 192.168.77.132:8080
Connection: keep-alive
Accept-Encoding: identity;q=1, *;q=0
Save-Data: on
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79
Safari/537.36
Accept: */*
DNT: 1
Referer: http://192.168.77.132:8080/
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
Range: bytes=25607620-
```

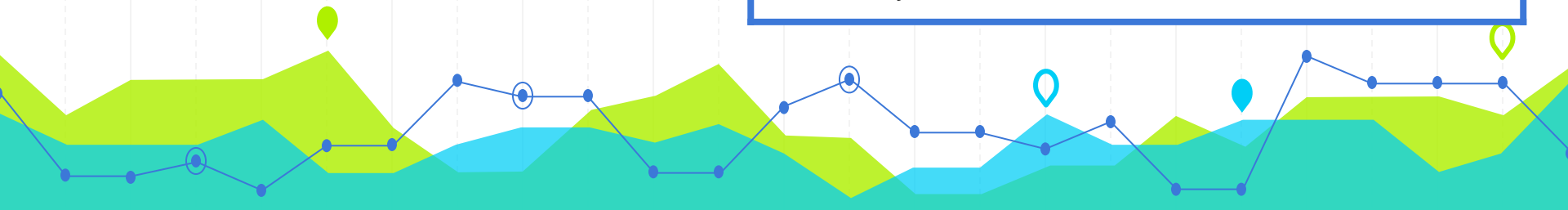
```
HTTP/1.0 206 Partial Content
Accept-Ranges: bytes
Content-Range: bytes 25607620-45349551/45349552
Cache-Control: public, max-age=0
Content-Type: video/mp4
Content-Length: 19741932
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With, Content-Type,
X-Codingpedia, X-HTTP-Method-Override
x-content-type-options: nosniff
Date: Wed, 08 Jun 2016 04:09:18 BRT
X-Powered-By: Jorismar
```

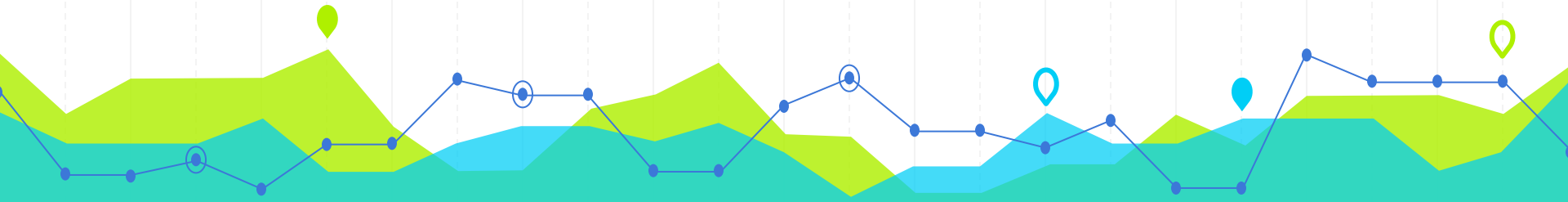


PÁGINA NÃO ENCONTRADA

```
GET /not_found.html HTTP/1.1
Host: 192.168.77.132:8080
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Save-Data: on
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
DNT: 1
Referer: http://192.168.77.132:8080/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

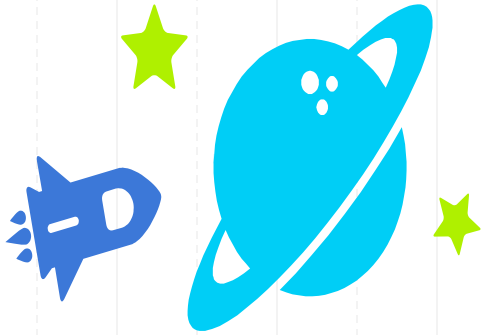
```
HTTP/1.0 404 Not Found
Cache-Control: public, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Length: 492
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Requested-With, Content-Type,
X-Codingpedia, X-HTTP-Method-Override
x-content-type-options: nosniff
Date: Wed, 08 Jun 2016 04:28:05 BRT
X-Powered-By: Jorismar
```





Demonstração 4

Webserver + Webpage



LET'S TRY

http://127.0.0.1:8090

OBRIGADO!

DÚVIDAS?

jorismar.barbosa@lavid.ufpb.br