

# Introduzione

Il progetto denominato "Logview" vuole essere una semplice applicazione web per visualizzare i log con una comoda funzione di ricerca. Permette di scegliere il tipo di log e il contenuto delle entità da visualizzare.

Il progetto è stato reso pubblico sulla nota piattaforma GitHub con il seguente link: <https://github.com/XelK/logview/> , inoltre l'applicazione è stata deployata sull'hosting heroku ed è raggiungibile con questo url:

<https://llogviewer.herokuapp.com/>

## Analisi dei requisiti

### Destinatari

La tipologia di log utilizzati che è stata scelta è quella di log **Apache**, in quanto ben documentata e molto presente nel mercato aziendale. La documentazione del formato utilizzato è presente in questo url:

<https://httpd.apache.org/docs/2.4/logs.html> , mentre i file di log utilizzati per il test sono stati generati con l'applicazione opensource "flog" reperibile da questo url: <https://github.com/mingrammer/flog> .

Utente tipo per questo tipo di applicazione è un utente esperto che sa cosa cerca ed utilizza il device con uno schermo grande come computer. Inoltre è stata resa disponibile l'opzione per interrogare il log richiesto senza dover accedere alla pagina web tramite browser ma effettuando delle chiamate API.

### Modello

L'obiettivo è quello di fornire un supporto ad uno tecnico o ad un team di nizi e vuole consultare con semplicità porzioni di log . È un valore aggiunto poiché permette una rapida estrapolazione dei log di interesse senza dover collegarsi al server web.

## Aspetti tecnologici

Aspetti tecnologici

Per la realizzazione di Logviewer sono state sfruttate diverse tecnologie che andrò a riepilogare di seguito:

- NodeJS ovvero un runtime JavaScript costruito sul motore di JavaScript V8 di Chrome, sfruttato come backend;

- Express.js un framework web veloce, non categorico e minimalista per NodeJS;
  - HTML5, CSS3 e JS sono stati utilizzati per la parte di frontend, si è scelto di non caricare il frontend con dei framework in quanto si è cercato di mantenere l'interfaccia più leggera, semplice e minimale.
- L'accesso inoltre è stato protetto con username e password.

# Interfacce

L'applicazione è una classica single-page application con i seguenti parametri:

- scelta dell'intervallo di interesse
- scelta del tipo di log da visualizzare tra i log presenti (access/error/custom)
- opzionale parametri da ricercare nel log selezionato in precedenza.

### Simple log visualizer

ip	userid	date	method	path	proto	resp	dim
145.148.55.80	-	13/Aug/2021 10:39:35	HEAD	/solutions	HTTP/1.0	400	12925
82.214.6.245	-	13/Aug/2021 10:39:35	PUT	/b2c/networks/sexyl/empower	HTTP/1.0	405	21339
251.44.37.60	breitenberg4068	13/Aug/2021 10:39:35	PATCH	/relationships/riich/drive/e-markets	HTTP/2.0	301	18440
207.146.242.239	-	13/Aug/2021 10:39:35	DELETE	/intuitive/incubate/turn-key	HTTP/1.0	301	13788
34.188.98.143	-	13/Aug/2021 10:39:35	DELETE	/extend	HTTP/1.1	404	15404
112.207.215.105	hane3873	13/Aug/2021 10:39:35	POST	/experiences/platforms/content	HTTP/1.0	406	22401
37.73.185.117	feeney3004	13/Aug/2021 10:39:35	GET	/transition/leading-edge/global/vortals	HTTP/1.0	503	19248
25.252.18.76	gorczany4035	13/Aug/2021 10:39:35	PATCH	/envisioneer/whiteboard	HTTP/1.0	500	24463
144.182.10.238	smith7547	13/Aug/2021 10:39:35	POST	/open-source/engage	HTTP/1.1	503	16394
168.107.36.213	fahey2848	13/Aug/2021 10:39:35	GET	/synergize/whiteboard	HTTP/1.1	404	11690
71.205.93.220	-	13/Aug/2021 10:39:35	POST	/e-business/intuitive	HTTP/1.1	501	27401
235.110.199.159	romaguera1155	13/Aug/2021 10:39:35	PUT	/e-markets/models	HTTP/2.0	200	14096
63.196.185.69	-	13/Aug/2021 10:39:35	POST	/redefine/monetize/incubate/engage	HTTP/1.0	503	17352
243.63.184.242	-	13/Aug/2021 10:39:35	DELETE	/extend/iterate/vortals/web-readiness	HTTP/2.0	100	26836
210.150.97.82	lehner7344	13/Aug/2021 10:39:35	HEAD	/holistic/utilize/relationships	HTTP/1.0	500	6805
188.127.94.51	upton1232	13/Aug/2021 10:39:35	POST	/b2b	HTTP/1.0	205	21265
188.162.189.21	-	13/Aug/2021 10:39:35	HEAD	/real-time/envisioneer/revolutionary/reintermediate	HTTP/1.1	200	26836
128.87.68.228	ernser9635	13/Aug/2021 10:39:35	HEAD	/front-end	HTTP/1.1	406	9686
140.65.85.162	lemke8217	13/Aug/2021 10:39:35	POST	/24%2f	HTTP/1.0	204	7094
45.59.206.35	-	13/Aug/2021 10:39:35	DELETE	/innovate/customized	HTTP/2.0	403	18151
124.37.59.172	-	13/Aug/2021 10:39:35	PUT	/eyeballs/web+services/reintermediate	HTTP/2.0	201	11204
166.186.86.142	-	13/Aug/2021 10:39:35	POST	/turn-key/robust	HTTP/1.1	200	660
184.118.126.187	oreilly7933	13/Aug/2021 10:39:35	GET	/seamless/visualize	HTTP/1.0	100	27200
127.204.198.112	-	13/Aug/2021 10:39:35	PATCH	/vertical/interfaces/e-services	HTTP/1.0	100	11022
186.103.179.71	hoppe2914	13/Aug/2021 10:39:35	PATCH	/incentivize/rich/scale/channels	HTTP/1.1	416	4896
192.149.239.161	-	13/Aug/2021 10:39:35	HEAD	/compelling/24%2f/mission-critical	HTTP/1.0	400	10866
167.214.56.29	-	13/Aug/2021 10:39:35	DELETE	/architectures	HTTP/1.0	200	1551
73.133.106.18	-	13/Aug/2021 10:39:35	DELETE	/user-centric	HTTP/1.1	205	24979
74.244.156.201	brown4049	13/Aug/2021 10:39:35	GET	/partnerships/monetize	HTTP/1.1	416	3547
248.249.243.58	-	13/Aug/2021 10:39:35	DELETE	/reintermediate/integrate	HTTP/1.0	204	21930
177.99.56.29	-	13/Aug/2021 10:39:35	DELETE	/interactive/customized	HTTP/2.0	404	5333
140.254.14.225	-	13/Aug/2021 10:39:35	PUT	/channels/extend/matrix	HTTP/2.0	100	27317
91.164.141.220	-	13/Aug/2021 10:39:35	DELETE	/granular	HTTP/1.0	401	26526
222.26.143.70	-	13/Aug/2021 10:39:35	PUT	/b2b/holistic/paradigms	HTTP/1.0	404	14788
54.156.88.177	konwank2737	13/Aug/2021 10:39:35	HEAD	/web+source/holistic/extendible/integrated	HTTP/2.0	405	21714

**Select Logs:**

FROM: 08/01/2021, 01:00:00 AM ☐

TO: 08/31/2021, 01:00:00 AM ☐

PARAMS:

Access Log

Error Log

Custom Log

**Manual:**

**Interface:**

1. Select from date
2. Select to date
3. (Optional) Insert parameters to search in params field, use "param01=value,param02=value" syntax
4. Click on log type to visualize

**API call:**

- http://usr.psw@host.plapi?type=from/to?par
  - usr: username
  - psw: password
  - port
  - type: error/access/custom
  - from: start date format: 13-Aug-2021:10:39:45
  - to: end date, same format as from
  - par: (optional) list of parameters, format:
    - param1=value&param2=value

Simple web application, developed for the course of "Programmazione Web e Mobile" of [Sicurezza dei Sistemi e delle Reti Informatiche](#) at University of Milan. This application was created with use of HTML5, CSS3, Javascript for the frontend, NodeJs (Expressjs framework) for the backend. It also implement REST api call that return json file with searched logs items. Html/CSS was checked with [w3c validator](#)

Oleksandr Kubashov @

Inoltre è possibile ottenere gli stessi risultati effettuando la chiamata get fornendo i campi di interesse. L'url di richiesta deve essere formata in questo modo:

<https://usr:psw@l0gviewer.herokuapp.com/api/type/from/to?params>

dove:

- usr/psw: inidcano username e password necessarie per autenticarsi
- l0gviewer.herokuapp.com: url dove è stata deployata l'applicazione
- api: url della pagina costruita con Express
- type: tipo di log da visualizzare (access/error/custom)
- from/to: intervallo di date in formato: **1-Aug-2021:11:40:52**
- params: lista di parametri opzionale, nel formato:

**parametro1=valore&parametro2=valore**

# Architettura

L'applicazione si compone di 3 grandi blocchi.

1. Pagina principale visibile visitando l'applicazione con browser. Questa ha i suddetti compiti:
  1. interagire con l'utente registrando i dati selezionati;
  2. effettuare una chiamata verso la pagina di api come se si effettuasse una chiamata get descritta in precedenza
  3. e dopo aver ricevuto il file json con il risultato trasformare questo in una tabella da visualizzare all'utente.Il codice di maggior interesse in questa parte è realizzato in javascript di seguito forniamo la funzione di interesse, che si occupa di trasformare i parametri selezionati dall'utente in formato accettato dalle api realizzati.

```
/**
 * Function used into html page.
 * Read parameters insereted into html page. Convert data parameters,
check the correctness
 * of parameters and execute the HttpRequest to the api page
 * @param button [selected button on the page]
 * @param logType [type of log selected, can be: access/error/custom]
 */
function selectLog(button, logType) {
    var i, tablinks;

    let from=document.getElementById("from").value;
    let to=document.getElementById("to").value;
    let params=document.getElementById("params").value;

    const month=
["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];

    // convert to date format requested by api
    const reg=/(\d+)-(\d+)-(\d+)T(\d+):(\d+):(\d+)/;
    let t=from.match(reg);
    from=t[3]+"-"+month[t[2]%10-1]+"-
"+t[1]+":"+t[4]+":"+t[5]+":"+t[6];
    t=to.match(reg);
```

```

to=t[3]+"-"+month[t[2]%10-1]+"-"+t[1]+":"+t[4]+":"+t[5]+":"+t[6];

tablinks = document.getElementsByClassName("tablinks");
for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active",
    "");
}
button.currentTarget.className += " active";

// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onreadystatechange=function(){
    if (xhttp.readyState === 4) {
        if (xhttp.status === 200) {
            createTable(this.responseText);
        } else {
            alert("Error code: "+xhttp.status+"\nPlease insert correct
parameters!");
        }
    }
}

// Send a request
if (params.length>0)

    var
request="api/"+logType+"/"+from+"/"+to+"/"+?" +params.replace(/\,/g,"&");
    else
        var request="api/"+logType+"/"+from+"/"+to;
    xhttp.open("GET",request, true);
    xhttp.send();

}

```

2. Pagina api, questa si posiziona in mezzo tra l'interfaccia web e il file che trasforma il file log in formato json. La suddetta pagina è accessibile anche via chiamata **GET**. Svolge inoltre il compito di verificare la correttezza dei dati passati ed eventualmente restituire la pagina d'errore:

```

router.get('/:type/:from/:to', function (req, res,next) {

  /**
   * Object that contain parameters passed via REST call.
   * @param type [type of log to read, can be access/error/custom]
   * @param from [start date]
   * @param to [end date]
   * @param params [optional list of parameters]
   */
  const request={
    // mandatory parameters:
    type: req.params.type,
    from:req.params.from,
    to:req.params.to,
    // optional params:
    params:req.query
  }

  let resp=getData(request);
  let js = createJson(resp,request.type);

  // Errors handling
  // from/to inserted not correctly
  if(dataRangeError(request.from,request.to))
    return next(createError(400)); // 400: bad request
  // wrong parameters format inserted
  if(Object.values(request.params).includes(''))
    return next(createError(400)); // 400: bad request

  resp=[];
  js.forEach(element => {
    if(dateBetween(request.from,request.to,element.date)){
      if(find(element,request.params))
        resp.push(element);
    }
  });
  res.json(resp);
});

```

Funzioni di controllo che sono stati realizzati nel codice:

```
/**
 * Function that search if param passed via REST is present in the js
object
 * @param item [item from file log]
 * @param params [array of params passed via REST call]
 */
function find(item,params){
    var ret=true;
    Object.keys(params).forEach(pKey => {
        if( (typeof params[pKey] != "undefined") &&
(!item[pKey].includes(params[pKey])) )
            ret=false;
    });
    return ret;
}

/**
 * Function that check the correctness of data range
 * @param start [start date]
 * @param end [end date]
 */
function dataRangeError(start,end){
    let s=start.replace(":", " ");
    let e=end.replace(":", " ");

    s=Date.parse(s);
    e=Date.parse(e);
    if(e<s) // end date is before start date
        return true;
    return false;
}

/**
 * Function that check if date from log is between start/end date
passed by user
 * @param start [start date]
 * @param end [end date]
 * @param data [date from file]
 */
```

```
function dateBetween(start,end,data){
    let s=start.replace(":", " ");
    let e=end.replace(":", " ");
    let d=data.day+"/"+data.month+"/"+data.year+"
"+data.hour+": "+data.minutes+": "+data.seconds;

    s=Date.parse(s);
    e=Date.parse(e);
    d=Date.parse(d);

    return( (s<=d) && (d<=e));

}
```

3. Infine è presente un terzo file javascript `logfiles.js` che si preoccupa di leggere il file log corretto e trasformare il contenuto in formato json. Questo è stato realizzato come una libreria esterna. Di seguito il codice per il log di tipo `access`:

```
/**
 * Function that trasform apache ACCESS log file into json
 * @param data [file content]
 * @returns json data with access log
 */
exports.jsonAccess = function (data){

    /* apache common log format: LogFormat "%h %l %u %t \"%r\" %>s %b"
common
    * see: https://httpd.apache.org/docs/2.4/logs.html
    * example:
    * 127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
    */

    var temp=data.split("\n");
    var resp=[];
    var i=0;
    const regAccess=/(\d+\.\d+\.\d+\.\d+) (\S+) (\S+) \
[(\d+)\]/(\w+)\]/(\d+)\]:(\d+)\]:(\d+)\](?:.*\)] "(\w+) (\S+)
(\S+)\\" (\d+) (\d+)/;
```

```

temp.forEach((line)=>{
  const w=line.match(regAccess);
  if(w !== null){
    resp.push({});
    resp[i].ip=w[1];
    resp[i].l=w[2];
    resp[i].userid=w[3];
    resp[i].date={};
    resp[i].date.year=w[6];
    resp[i].date.month=w[5];
    resp[i].date.day=w[4];
    resp[i].date.hour=w[7];
    resp[i].date.minutes=w[8];
    resp[i].date.seconds=w[9];
    resp[i].method=w[10];
    resp[i].path=w[11];
    resp[i].proto=w[12];
    resp[i].resp=w[13];
    resp[i].dim=w[14];
    i++;
  }
});
return resp;
}

```

## Codice

Utilizzo di **nodejs** e specialmente framework **express** ha facilitato la realizzazione dell'applicazione in questione. In particolare è stato possibile utilizzare **javascript** sia per il frontend che per il backend. Inoltre utilizzo del framework ha semplificato la realizzazione dell'autenticazione e in particolare la realizzazione del routing con la costruzione di una url dinamica per la chiamate API, che, altrimenti, sarebbe stata una delle parti più ardue da realizzare. Il codice html/css inoltre è stato testato con **w3cvalidator** <https://validator.w3.org/> superando correttamente il test.

Considero un grande vantaggio aver adottato il **modello MVC** separando la parte di frontend da quella di backend. E quindi realizzato e reso disponibile delle API in modo sia di garantire una fruizione di questa via chiamate GET, senza dover aprire necessariamente il browser o anche utilizzando ulteriori tool automatici o a



riga di comando per una futura elaborazione di dati.

Inoltre il fatto di separare la parte frontend da quella backend permette un eventuale futuro sviluppo da parte di 2 team di sviluppatori o anche una eventuale futura sostituzione di una delle due parti.

# Strumenti

L'applicazione è stata realizzata utilizzando **vscode** per la scrittura del codice in quanto si integra perfettamente con javascript/html/css. Inoltre è stato utilizzato **git** con **github** per il versioning del codice. Durante lo sviluppo inoltre sono stati utilizzati **nodemon** tool che aiuta lo sviluppo con node.js in quanto esegue in automatico il riavvio dell'applicazione quando si effettua la modifica del codice. Per il test delle API invece è stata utilizzata l'applicazione **insomnia** per la semplicità di configurazione, di seguito qualche screenshot di questa:

- Error log:

The screenshot shows the Insomnia API client interface. The left sidebar lists several API endpoints, with 'API-GET-error-PRODUCTION' selected. The main panel displays the details of a GET request to 'https://18gviewer.herokuapp.com/api/type/from/to'. The response is a JSON object with the following structure:

```
1 {
2   "ip": "104.211.141.61",
3   "date": {
4     "day": "13",
5     "weekDay": "Fri",
6     "month": "Aug",
7     "year": "2021",
8     "hour": "18",
9     "minutes": "39",
10    "seconds": "45"
11  },
12  "msg": "We need to connect the online IS driver!",
13  "severity": "emerg",
14  "module": "isss"
15 }
16 {
17   "ip": "181.52.131.12",
18   "date": {
19     "day": "13",
20     "weekDay": "Fri",
21     "month": "Aug",
22     "year": "2021",
23     "hour": "18",
24     "minutes": "39",
25     "seconds": "45"
26  },
27  "msg": "If we compress the hard drive, we can get to the PMS transmitter through the mobile J50N system!",
28  "severity": "info",
29  "module": "vel"
30 }
31 {
32   ...
33 }
```

- Access log:

The screenshot shows the Insomnia API client interface. The left sidebar lists several API endpoints, with 'API-GET-access-PROD' selected. The main panel displays the details of a GET request to 'https://18gviewer.herokuapp.com/api/type/from/to'. The response is a JSON object with the following structure:

```
1 {
2   "ip": "121.141.93.103",
3   "l": "...",
4   "userId": "muller2134",
5   "date": {
6     "year": "2021",
7     "month": "Aug",
8     "day": "13",
9     "hour": "18",
10    "minutes": "39",
11    "seconds": "35"
12  },
13  "method": "PUT",
14  "path": "/morph/markets",
15  "proto": "HTTP/1.0",
16  "resp": "403",
17  "dir": "28890"
18 }
19 {
20   ...
21 }
```

- Custom log:

The screenshot shows the Insomnia API client interface. The left sidebar lists several API endpoints, with 'API-GET-custom-PROD' selected. The main panel displays the details of a GET request to 'https://18gviewer.herokuapp.com/api/type/custom/from/to'. The response is a JSON object with the following structure:

```
1 {
2   "ip": "189.33.62.123",
3   "l": "...",
4   "userId": "turcotte4977",
5   "date": {
6     "year": "2021",
7     "month": "Aug",
8     "day": "14",
9     "hour": "18",
10    "minutes": "39",
11    "seconds": "35"
12  },
13  "method": "DELETE",
14  "path": "/applications/reinvent",
15  "proto": "HTTP/1.0",
16  "statusCode": "200",
17  "dir": "msssp",
18  "referal": "https://www.humanpower.io/innovative",
19  "userAgent": "Mozilla/5.0 (Windows NT 5.1; AppleWebKit/5331 (KHTML, like Gecko) Chrome/57.0.879.0 Mobile Safari/5331"
20 }
21 {
22   ...
23 }
```

Nel codice inoltre sono presenti numerosi **espressioni regolari**, per semplicità di realizzazione è stato utilizzato lo strumento disponibile a questo url:

<https://regex101.com/>

Per quanto riguarda il deploy è stato scelto **heroku** in quanto presenta un piano gratuito e si integra perfettamente con github e nodejs.

## Conclusione

In conclusione posso dire che il progetto è stato interessante, mi ha permesso di interfacciarmi con il nuovo linguaggio NodeJs e framework Express. Ho trovato di notevole interesse il funzionamento di NodeJs e la semplicità del linguaggio permettendomi di concentrarmi sulla parte backend, dove ho potuto divertirmi con l'elaborazione dei dati di log, e realizzazione di un semplice **API REST** anche se implementando solo il comando **GET**. In aggiunta ho potuto esercitarmi con delle espressioni regolari, strumento indispensabile per qualsiasi sistemista.