

Programmazione Web e Mobile

Rocco Amico - Matricola N. 12190A

Traccia

Metriche Youtube

Il progetto prende spunto dalla traccia “Metriche Twitter” proposta dal docente adattandola all'utilizzo della API di Youtube a seguito della restrizione al pubblico sulla API di Twitter.

Requisiti

HTML5/CSS

- Le pagine devono essere sviluppate in formato HTML5.
- Tutte le pagine devono essere validate.
- Il layout delle pagine deve essere sviluppato con CSS.
- L'applicazione dovrà servirsi di almeno una API HTML5.

AJAX

- Il progetto deve implementare una o più chiamate XMLHttpRequest.
- Le chiamate possono interrogare dati in JSON, XML, XHTML, TXT.

NodeJS

- Il progetto deve implementare una o più chiamate a un servizio NodeJS sviluppato dallo studente.
- Le chiamate devono interrogare o caricare dati in JSON o XML.

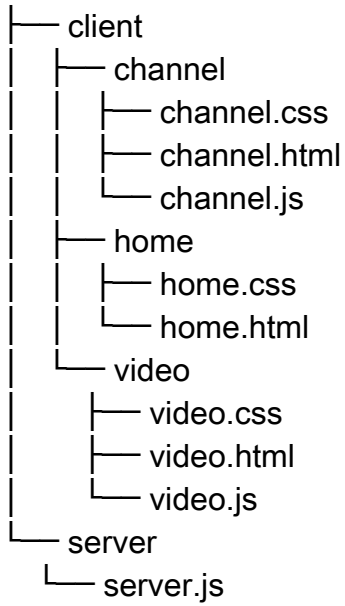
Progetto

MetricON nasce con l'idea di fornire statistiche chiave di un dato canale o di un dato video su Youtube utilizzando la API di Google “Youtube Data API v3”.

Il progetto vuole abbattere la barriera di difficoltà tecnologica che all'utente comune comporterebbe un utilizzo diretto delle API, favorendo l'usabilità da interfaccia web e riducendo ad un unico click quello che si otterrebbe con molteplici chiamate API.

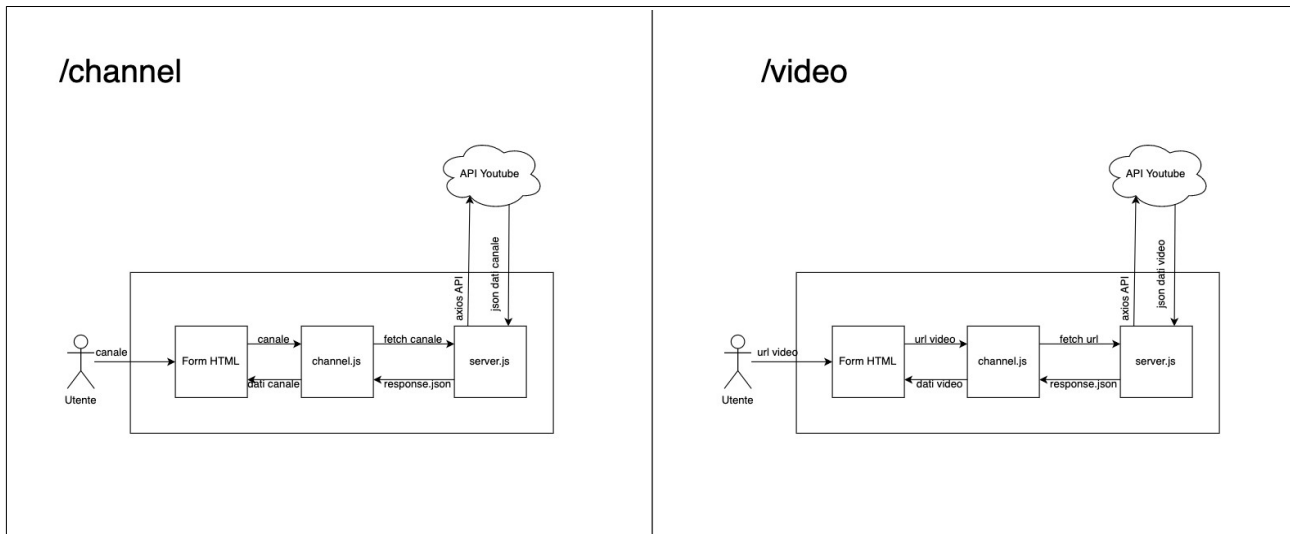
Il progetto è disponibile online al seguente link: <https://www.roccoamico.it/metricon>

Struttura



Logica

Flusso

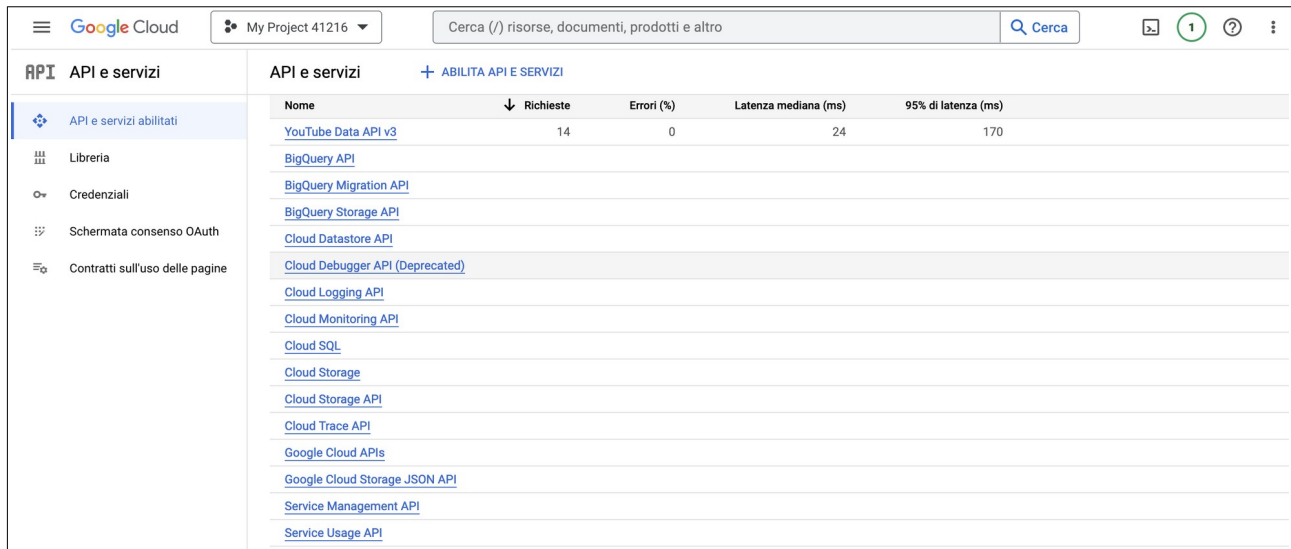


Chiave API

Registrandosi al programma sviluppatori di Google, è possibile chiedere l'accesso alle API, in particolare quella di nostro interesse è la Youtube Data API v3: viene fornita una chiave che dovrà essere utilizzato in tutte le chiamate.

Per comodità nel progetto la API Key è stata inserita come costante nel server.js (è stata generata con un account usa e getta creato ad hoc).

In un progetto deployato online la API Key è consigliabile definirla come una variabile d'ambiente all'interno del cloud provider/server.



API e servizi		+ ABILITA API E SERVIZI			
	Nome	↓ Richieste	Errori (%)	Latenza mediana (ms)	95% di latenza (ms)
API e servizi abilitati	YouTube Data API v3	14	0	24	170
Libreria	BigQuery API				
Credenziali	BigQuery Migration API				
Schermata consenso OAuth	BigQuery Storage API				
Contratti sull'uso delle pagine	Cloud Datastore API				
	Cloud Debugger API (Deprecated)				
	Cloud Logging API				
	Cloud Monitoring API				
	Cloud SQL				
	Cloud Storage				
	Cloud Storage API				
	Cloud Trace API				
	Google Cloud APIs				
	Google Cloud Storage JSON API				
	Service Management API				
	Service Usage API				

Dashboard del pannello Google Cloud dove richiedere la chiave di Youtube Data API v3

Home Page

Dalla home page possiamo selezionare l'opzione che ci interessa tra statistiche video e statistiche canale.



MetricON

Utilizziamo le API di YouTube per mostrarti alcuni dati interessanti riguardanti un dato canale o un dato video.

Scopri di più:

[Visualizza statistiche video](#) [Visualizza statistiche canale](#)

La home page è presente alla root path, i due bottoni ci portano a /video e /channel

/channel

MetricON

Statistiche canale Youtube

Canale YouTube:

Visualizza statistiche canale

Una volta premuto il bottone, il canale inserito dall'utente viene comunicato lato client dal form nel channel.html al channel.js.

```
<form id="channelForm" class="mt-3">
  <div class="mb-3">
    <label for="channelInput" class="form-label">Canale YouTube:</label>
    <input type="text" class="form-control" id="channelInput" required>
  </div>
  <button type="submit" class="btn btn-primary">Visualizza statistiche canale</button>
</form>
```

channel.html

```
document.getElementById( 'channelForm' ).addEventListener( 'submit', listener: async ( event : SubmitEvent ) : Promise<void> => {
  event.preventDefault();

  const channel = document.getElementById( 'channelInput' ).value;

  const response : Response = await fetch( input: `/api/channel?channel=${channel}` );
  const responsePopular : Response = await fetch( input: `/api/popularVideos?channel=${channel}` );

  const data = await response.json();
  const dataPopular = await responsePopular.json();
```

channel.js

Il channel.js passa la richiesta al server.js che si occupa di effettuare due chiamate:

1. Recupero dati generali del canale che verranno inseriti nelle card
2. Recupero dati relativi ai video più visualizzati di un dato canale

Per effettuare le chiamate API, oltre ad Express, utilizziamo Axios, che assembla il nome del canale preso dal channel.js e la chiave API ottenendo indietro i dati in formato JSON.

```
// Gestione della richiesta GET a '/api/channel'
app.get('/api/channel', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
  try {

    const channel :... = req.query.channel;
    const apiUrl : string = `https://www.googleapis.com/youtube/v3/channels?part=snippet,statistics,status&forUsername=${channel}&key=${apiKey}`;
    const response : AxiosResponse<any> = await axios.get(apiUrl);

    const data = response.data;

    if (data.items.length === 0) {
      res.status( code: 404).json( body: { error: 'Channel not found' });
      return;
    }

    const statistics : ({...})[] = data.items[0].statistics;
    const snippet = data.items[0].snippet;
    const status = data.items[0].status;

    const channelData : {...} = {
      subscriberCount: statistics.subscriberCount,
      viewCount: statistics.viewCount,
      videoCount: statistics.videoCount,
      hiddenSubscriberCount: statistics.hiddenSubscriberCount,
      country: snippet.country,
      madeForKids: status.madeForKids
    };

    res.json(channelData);
  } catch (error) {
    res.status( code: 500).json( body: { error: 'Internal server error' });
  }
});
```

1. Recupero dati generali del canale che verranno inseriti nelle card

```
app.get('/api/popularVideos', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : Promise<void> => {
  try {
    const channel :... = req.query.channel;

    const apiUrlChannelID : string = `https://www.googleapis.com/youtube/v3/channels?key=${apiKey}&forUsername=${channel}&part=id`;
    const responseChannelID : AxiosResponse<any> = await axios.get(apiUrlChannelID);
    const dataChannelID = responseChannelID.data;
    const channelID = dataChannelID.items[0].id;

    const apiUrl : string = `https://www.googleapis.com/youtube/v3/search?part=snippet&channelId=${channelID}&key=${apiKey}&maxResults=5&order=viewcount`;
    const response : AxiosResponse<any> = await axios.get(apiUrl);
    const data = response.data;

    const videos : any[] = [];

    if (data.items.length === 0) {
      res.status( code: 404).json( body: { error: 'Channel not found' });
      return;
    }

    for (const item : any of data.items) {
      const videoID = item.id.videoId;
      const videoUrl : string = `https://www.googleapis.com/youtube/v3/videos?id=${videoID}&key=${apiKey}&part=snippet,contentDetails,statistics,status`;
      const videoResponse : AxiosResponse<any> = await axios.get(videoUrl);
      const videoData = videoResponse.data;
      const statistics : ({...})[] = videoData.items[0].statistics;
      const video : {...} = {
        title: item.snippet.title,
        description: item.snippet.description,
        thumbnail: item.snippet.thumbnails.default.url,
        viewCount: statistics.viewCount,
      };
      videos.push(video);
    }
    res.json( body: { videos });
  }
});
```

2. Recupero dati relativi ai video più visualizzati di un dato canale.

Viene effettuata una prima chiamata per estrarre l'ID del canale, una seconda per estrarre i 5 video più popolari (non è possibile utilizzando solo il nome del canale) ed infine una terza per popolare un array contenente i dati relativi ai 5 video.

I dati estrapolati sono di nuovo passati al channel.js che si occupa di presentarli.

```
if (response.ok) {
  // errobox nascosta
  document.getElementById( elementId: 'errorBox').classList.add('d-none');

  // Azzeramento dei valori delle statistiche
  document.getElementById( elementId: 'subscriberCount').textContent = '';
  document.getElementById( elementId: 'viewCount').textContent = '';
  document.getElementById( elementId: 'videoCount').textContent = '';
  document.getElementById( elementId: 'hiddenSubscriberCount').textContent = '';
  document.getElementById( elementId: 'country').textContent = '';
  document.getElementById( elementId: 'madeForKids').textContent = '';

  // Nuova assegnazione
  document.getElementById( elementId: 'subscriberCount').textContent = data.subscriberCount;
  document.getElementById( elementId: 'viewCount').textContent = data.viewCount;
  document.getElementById( elementId: 'videoCount').textContent = data.videoCount;
  document.getElementById( elementId: 'hiddenSubscriberCount').textContent = data.hiddenSubscriberCount;
  document.getElementById( elementId: 'country').textContent = data.country;
  document.getElementById( elementId: 'madeForKids').textContent = data.madeForKids;

  createBarChart(videos);

  statsChannel.classList.remove( tokens: 'd-none');
} else {
  statsChannel.classList.add('d-none');
  document.getElementById( elementId: 'errorBox').classList.remove( tokens: 'd-none');
}
```

channel.js

I dati generali vengono presentati come delle card

```
<div class="col-md-4">
  <div class="card">
    <div class="card-body">
      <h5 class="card-title">Numero di iscritti</h5>
      <p class="card-text" id="subscriberCount"></p>
    </div>
  </div>
</div>

<div class="col-md-4">
  <div class="card">
    <div class="card-body">
      <h5 class="card-title">Numero di visualizzazioni totali</h5>
      <p class="card-text" id="viewCount"></p>
    </div>
  </div>
</div>
```

estratto dell'html riguardante due card

Numero di iscritti 729000	Numero di visualizzazioni totali 192094615	Numero di video 1367
Counter iscritti nascosto false	Nazione IT	Made for Kids false

I dati relativi ai video più visualizzati vengono mostrati con un grafico a barre.

```
function createBarChart(videos) : void { 1usage
// Estrai i dati necessari per il grafico (titoli dei video e visualizzazioni)
const labels = videos.map(video => video.title);
const views = videos.map(video => video.viewCount);

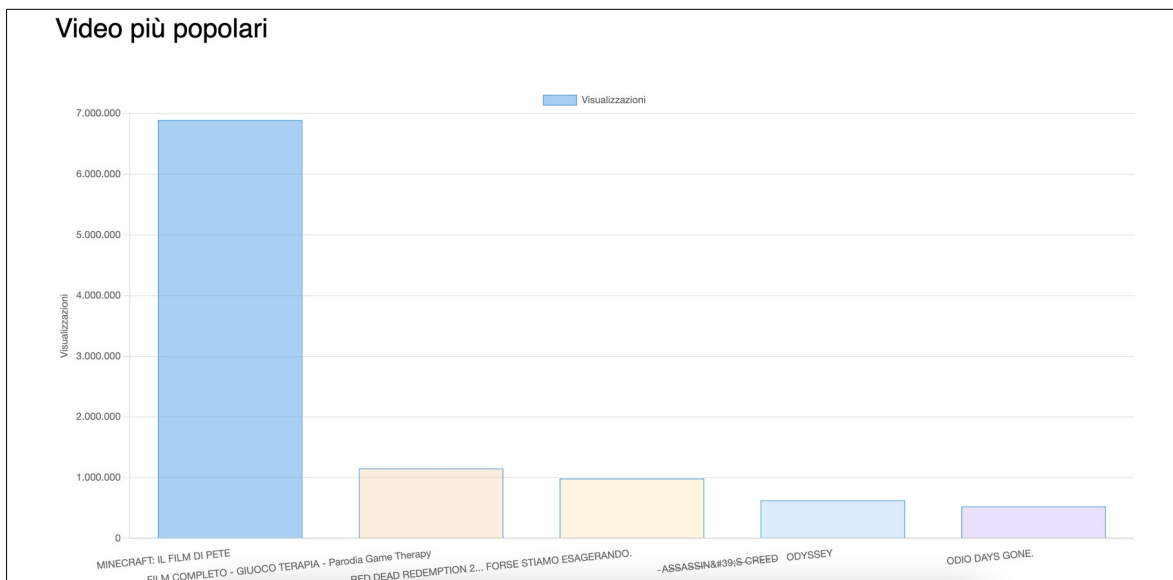
const existingChart : any = Chart.getChart("popularVideosChart");
if (existingChart) {
  existingChart.destroy();
}

const chartConfig : = {
  type: 'bar',
  data: {
    labels: labels,
    datasets: [{
      label: 'Visualizzazioni',
      data: views,
      backgroundColor: ['rgba(54, 162, 235, 0.5)', 'rgba(255, 159, 64, 0.2)', 'rgba(255, 205, 86, 0.2)', 'rgba(54, 162, 235, 0.2)',
      borderColor: 'rgba(54, 162, 235, 1)',
      borderWidth: 1,
      datalabels: {
        anchor: 'end',
        align: 'end'
      }
    }]
  },
  options: {
    plugins: {
      datalabels: {
        color: '#333', // colore del testo delle etichette
        font: {
          weight: 'bold'
        },
        formatter: (value, context) => {
          return value; // visualizza il valore delle visualizzazioni come etichetta
        }
      }
    }
  },
},
```

channel.js

```
<div class="chart-container">
  <h2 class="mt-4">Video più popolari</h2>
  <canvas id="popularVideosChart"></canvas>
</div>
```

channel.html



/video

MetricON

Statistiche video Youtube

URL Video:

Visualizza statistiche video

Una volta premuto il bottone, l'url inserito dall'utente viene comunicato lato client dal form nel video.html al video.js.

```
<form id="urlForm" class="mt-3">
  <div class="mb-3">
    <label for="urlInput" class="form-label">URL Video:</label>
    <input type="text" class="form-control" id="urlInput" required>
  </div>
  <button type="submit" class="btn btn-primary">Visualizza statistiche video</button>
</form>
```

video.html

```
document.getElementById( elementId: 'urlForm').addEventListener( type: 'submit', listener: async (event : SubmitEvent )
  event.preventDefault();

  const url = document.getElementById( elementId: 'urlInput').value;

  const response : Response = await fetch( input: `/api/video?url=${url}`);
  const data = await response.json();

  const statsVideo : HTMLElement = document.getElementById( elementId: 'statsvideo');
```

video.js

Il video.js passa la richiesta al server.js che si occupa estrarre dall'URL il suo ID univoco

```
app.get('/api/video', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> )
  try {

    const url = req.query.url;
    const videoId = extractVideoId(url);

    // Funzione per estrarre l'ID del video YouTube dalla URL
    function extractVideoId(url) :... { 1 usage
      // Controlla se l'URL contiene il parametro 'v'
      const urlParams : URLSearchParams = new URLSearchParams(new URL(url).search);
      if (urlParams.has( name: 'v')) {
        return urlParams.get('v');
      }

      // Controlla se l'URL contiene l'ID nel formato /watch?v=xxxxx
      const match = url.match(/\/watch\?v=(\w+)/);
      if (match) {
        return match[1];
      }

      // Controlla se l'URL contiene l'ID nel formato /embed/xxxxx
      const embedMatch = url.match(/\/embed\/(\w+)/);
      if (embedMatch) {
        return embedMatch[1];
      }

      // Restituisci null se l'ID del video non può essere trovato
      return null;
    }
  }
```

Solo ora è possibile richiedere i dati relativi al video.

```
const apiUrl : string = `https://www.googleapis.com/youtube/v3/videos?id=${videoId}&key=${apiKey}&part=snippet,contentDetails,statistics,status`;
const response : AxiosResponse<any> = await axios.get(apiUrl);

const data = response.data;

if (data.items.length === 0) {
  res.status( code: 404).json( body: { error: 'Video not found' });
  return;
}

const statistics : ({...})[] = data.items[0].statistics;

const videoData : {...} = {
  viewCount: statistics.viewCount,
  likeCount: statistics.likeCount,
  commentCount: statistics.commentCount
};
```

Come prima, i dati vengono passati lato client al video.js

```
if (response.ok) {
  document.getElementById( elementId: 'errorBox').classList.add('d-none');

  document.getElementById( elementId: 'videoViewCount').textContent = "";
  document.getElementById( elementId: 'likeCount').textContent = "";
  document.getElementById( elementId: 'commentCount').textContent = "";

  document.getElementById( elementId: 'videoViewCount').textContent = data.viewCount;
  document.getElementById( elementId: 'likeCount').textContent = data.likeCount;
  document.getElementById( elementId: 'commentCount').textContent = data.commentCount;
}
```

Che oltre alle card crea un grafico a torta rappresentante la proporzione tra commenti e mi piace.

```
const chartData : {datasets: [{backgroundColor: string[], b..., labels: ...}] = {
  labels: ['Likes', 'Comments'],
  datasets: [
    {
      data: [data.likeCount, data.commentCount],
      backgroundColor: ['rgba(54, 162, 235, 0.2)', 'rgba(255, 99, 132, 0.2)'],
      borderColor: ['rgba(54, 162, 235, 1)', 'rgba(255, 99, 132, 1)'],
      borderWidth: 1
    }
  ]
};

const canvas : HTMLCanvasElement = document.getElementById( 'videoChart');
const ctx = canvas.getContext('2d');

// Se il grafico esiste già, distruggilo prima di crearne uno nuovo
if (videoChart) {
  videoChart.destroy();
}

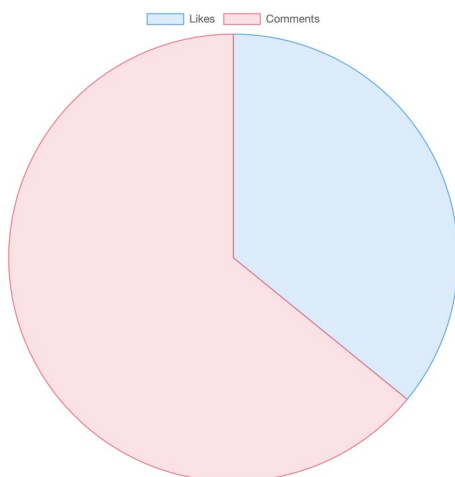
videoChart = new Chart(ctx, {
  type: 'pie',
  data: chartData,
  options: {
    title: {
      display: true,
      text: 'Statistiche video'
    }
  }
});
```

Numero di visualizzazioni
1647336

Numero di like
152016

Numero di commenti
271254

Proporzione tra like e commenti



Stile

Il sito utilizza congiuntamente Bootstrap e i fogli di stile per definire il layout grafico e la presentazione degli elementi.

Testing

Il parametro di input di /video è stato testato con

`https://www.youtube.com/watch?v=6i1uPfPzXlc`

`https://www.youtube.com/watch?v=WDmgVlwjFto`

`https://www.youtube.com/watch?v=bef8QLNHubw`

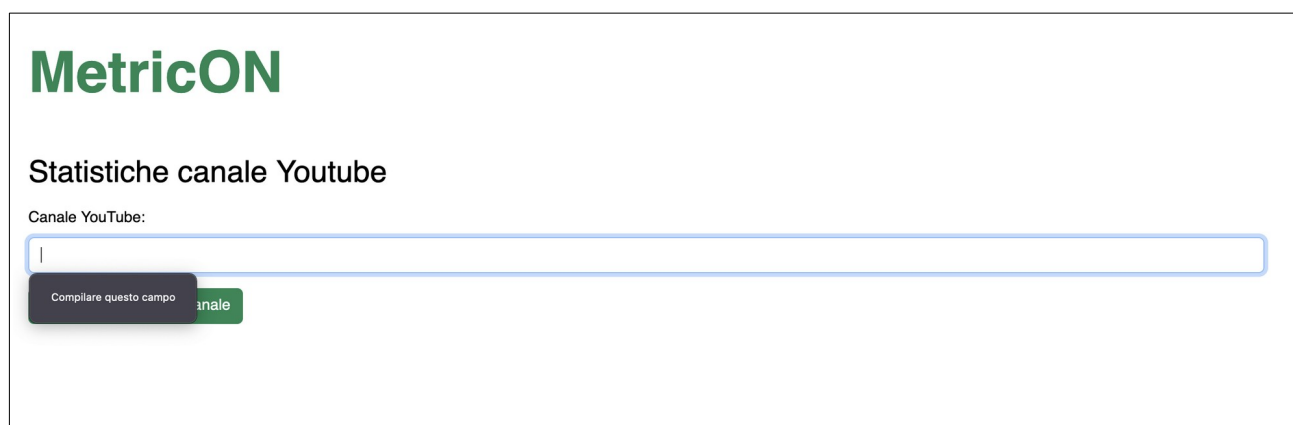
Il parametro di input di /channel è stato testato con

`queiduesulserver`

`pewdiepie`

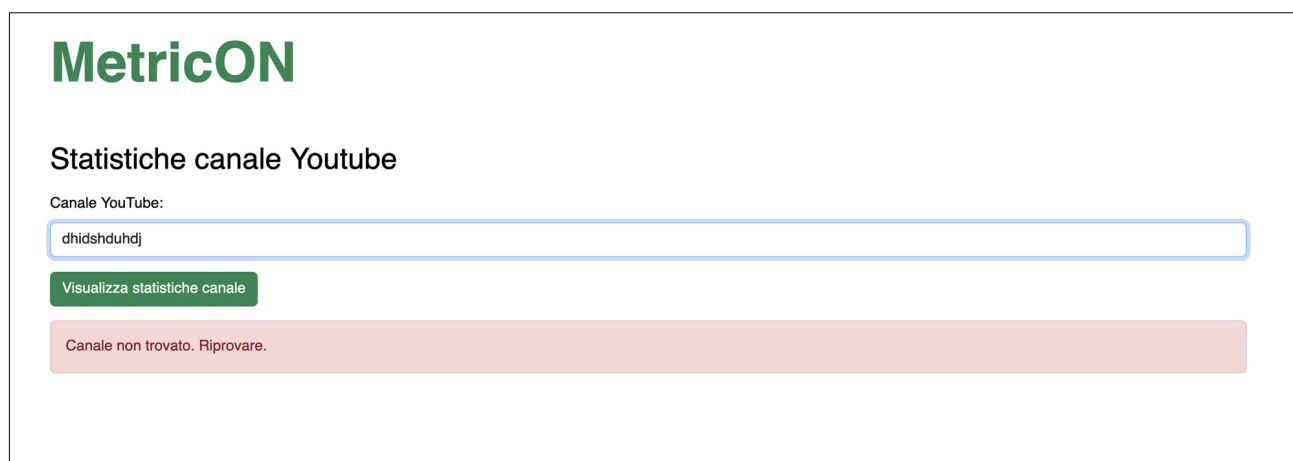
`Davie504`

In entrambi gli input non è possibile fare una richiesta con input vuoto.



The screenshot shows the MetricON website interface. At the top is the logo 'MetricON' in green. Below it is the title 'Statistiche canale Youtube'. Under the title is the label 'Canale YouTube:'. Below the label is a long, empty text input field. To the left of the input field, there is a dark grey tooltip that says 'Compilare questo campo'. To the right of the input field, there is a green button with the text 'Canale'.

Entrambi restituiscono un errore in caso di input non valido (in caso di canale o url video non esistente).



The screenshot shows the MetricON website interface. At the top is the logo 'MetricON' in green. Below it is the title 'Statistiche canale Youtube'. Under the title is the label 'Canale YouTube:'. Below the label is a text input field containing the text 'dhidshduhdj'. Below the input field is a green button with the text 'Visualizza statistiche canale'. Below the button is a red error message that says 'Canale non trovato. Riprovare.'

Versioning e hosting

Il progetto sfrutta git per il versionamento, è stato deployato su DigitalOcean e collegato alla zona DNS di OVH per il dominio roccoamico.it.

Modello di valore

Un possibile modello di valore è quello di inserire ads a pagamento all'interno del sito.

Istruzioni per l'esecuzione in locale

npm install

npm run start