

Big-Url: A Fullstack URL Shortener

Jacopo Scudieri - 986645



Introduzione

Big Url è una web app che permette a chi la utilizza di abbreviare lunghi indirizzi web (URL) in equivalenti di pochi caratteri. Inoltre fornisce le statistiche di utilizzo dei link abbreviati (short URL) e ne permette la gestione all'interno del database.

L'abbreviazione degli URL elaborati dall'app si basa sulla creazione di un'istanza nel database cloud MongoDB, la quale segue la struttura descritta dallo schema predisposto e abbina all'URL originale uno short ID generato randomicamente e un numero di click che viene incrementato ad ogni utilizzo del relativo short URL.

Qualora l'utente cercasse di abbreviare nuovamente lo stesso URL originale, l'app restituirà lo short URL generato in precedenza, mostrando il numero di click aggiornato.

Contestualmente può essere eliminata la voce dal database, resettando short ID e utilizzi.

L'app è live, hostata da [Heroku](#), mentre su [GitHub](#) si può esaminare il codice integrale.

Breve analisi dei requisiti

Destinatari

Capacità e possibilità tecniche

L'applicazione è stata sviluppata in maniera estremamente user friendly, per permetterne l'utilizzo anche ad utenti che non presentano grande praticità nel campo informatico. Per questo motivo l'interfaccia presenta gli elementi minimi per l'utilizzo del servizio.

Una volta resa pubblica ([qui](#)), l'applicazione diviene accessibile da qualunque dispositivo connesso a Internet e dotato di Web Browser, indipendentemente dal fatto che sia desktop o mobile in quanto l'interfaccia presenta un design responsive.

Un'implementazione futura potrebbe essere l'aggiunta dei cosiddetti "Vanity Url", ovvero Url abbreviati che possono essere personalizzati per riflettere il nome di un brand o di un servizio, oltre all'utilizzo di UTM (Urchin Tracking Module) ossia dei tag che permettono, grazie a Google Analytics, di tracciare il traffico generato da un Url.

Linguaggio

La totalità dell'interfaccia dell'applicazione è realizzata in lingua inglese, la lingua più diffusa sul web oltre che nel mondo.

In futuro si potrebbe valutare l'aggiunta di più lingue per allargare il numero di utenti che possono utilizzare l'applicazione.

Modello di valore

L'utilizzo di un'app come Big Url porta vantaggi sia in termini di risparmio di spazio che di aspetto più gradevole: troppo spesso nell'uso quotidiano di Internet capita di dover

utilizzare, implementare o condividere Url dalla lunghezza chilometrica, spesso pieni di parametri.

L'aspetto sintetico degli indirizzi generati da Big Url incoraggia quindi la condivisione da parte dell'utente, infatti, gli Url eccessivamente lunghi sono quasi impossibili da memorizzare o da trascrivere e talvolta anche da copiare, dal momento che alcuni software o servizi web trancano automaticamente gli Url troppo lunghi.

E' inoltre da sottolineare la possibilità, seppur ancora espandibile, di tenere traccia dell'utilizzo degli Url generati dall'app, in modo che l'utente possa utilizzare le diverse statistiche disponibili a suo vantaggio (commerciale, social, etc.).

Un altro elemento che dà valore all'applicazione è la sua implementazione, effettuata seguendo il modello MVC. Essendo il modello (Model, più stabile nel tempo) svincolato dalla presentazione (View, più variabile nel tempo), si riesce ad ottenere una grande espandibilità in quanto è possibile variare l'architettura del back-end senza intaccare le funzionalità del front-end e viceversa, o comunque effettuando modifiche minime ad una delle due parti.

Flusso dei dati

I contenuti sono rappresentati dalle istanze create nel database per ogni url originale inserito dall'utente. Il loro livello di dettaglio è virtualmente illimitato ma allo stato attuale prevedono:

- Url originale
- Short ID
- Url abbreviato (BASE url + Short ID)
- Numero di click
- ID di riconoscimento

Essi vengono prodotti in automatico in formato JSON dalla API e archiviati nel database cloud [MongoDB](#) che garantisce rapidità di accesso, sicurezza e ampia scalabilità.

Aspetti tecnologici

Come detto, l'applicazione segue il modello Model-View-Controller.

Model

Il Model, ovvero la struttura dei dati, determina come il database è organizzato, definendo la sezione dell'applicazione che interagisce con il database stesso. Qui è dove vengono definite le proprietà di ogni istanza creata dall'utente all'inserimento dell'url originale.

View

La View, ossia la struttura della pagina con cui interagisce l'utente, è stata creata utilizzando una combinazione di HTML5 e CSS3 con l'aiuto del framework Bootstrap, che permette di progettare in modo facile e veloce l'interfaccia responsive di un'applicazione Web fornendo dei template preconfezionati per ogni elemento che può essere inserito all'interno di una pagina.

Per strutturare l'interfaccia è stato anche utilizzato jQuery. Principalmente per la gestione dell'invio delle richieste POST ai due endpoint previsti, oltre che per la modifica delle proprietà CSS degli elementi.

Controller

In Big Url il Controller è il framework Express.js, eseguito all'interno di un server Node.js, che permette di creare degli endpoint (API REST) per la creazione di una nuova istanza all'interno del database o per eliminarne una già esistente e riportare dinamicamente i dati alla pagina che determina l'interfaccia (index.html), generata in base alla richiesta che viene effettuata.

Tramite degli specifici URL viene effettuata quindi una richiesta al server, che si occuperà di effettuare le elaborazioni necessarie per fornire la risorsa richiesta:

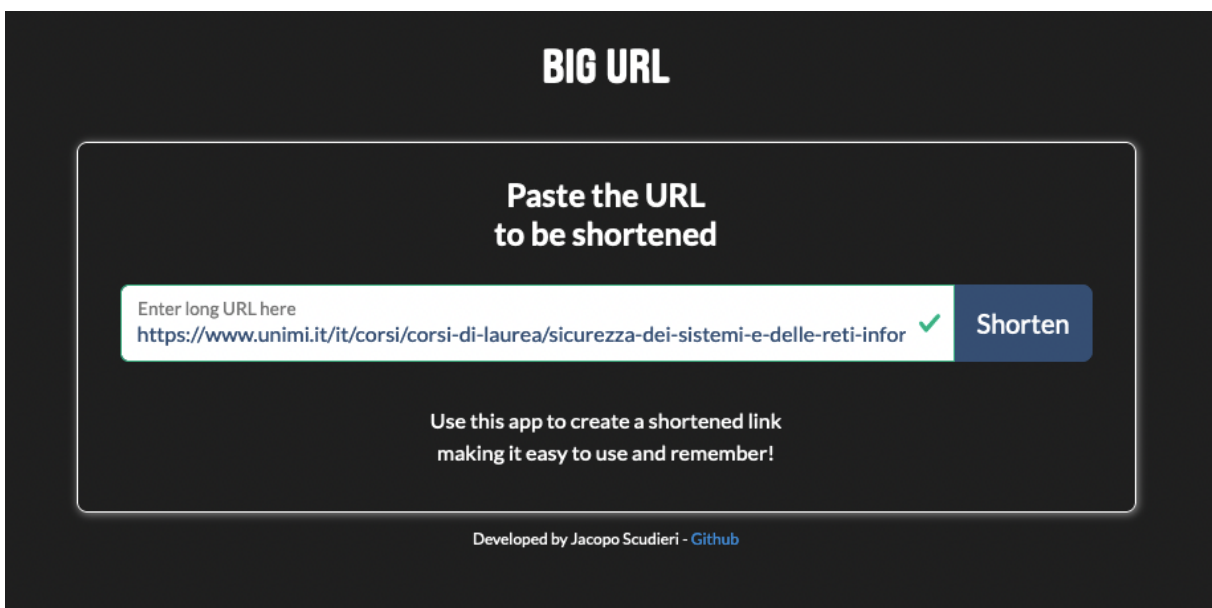
- verso `'/api/short'` la POST request per la creazione di una **nuova istanza**
- verso `'/shortID'` la GET request per il **reindirizzamento** all'url originale
- verso `'/del/short'` per l'**eliminazione** di un'istanza esistente dal database

Interfacce

L'applicazione è composta da una singola pagina principale più una di supporto nei casi di richiesta di short URL inesistenti.

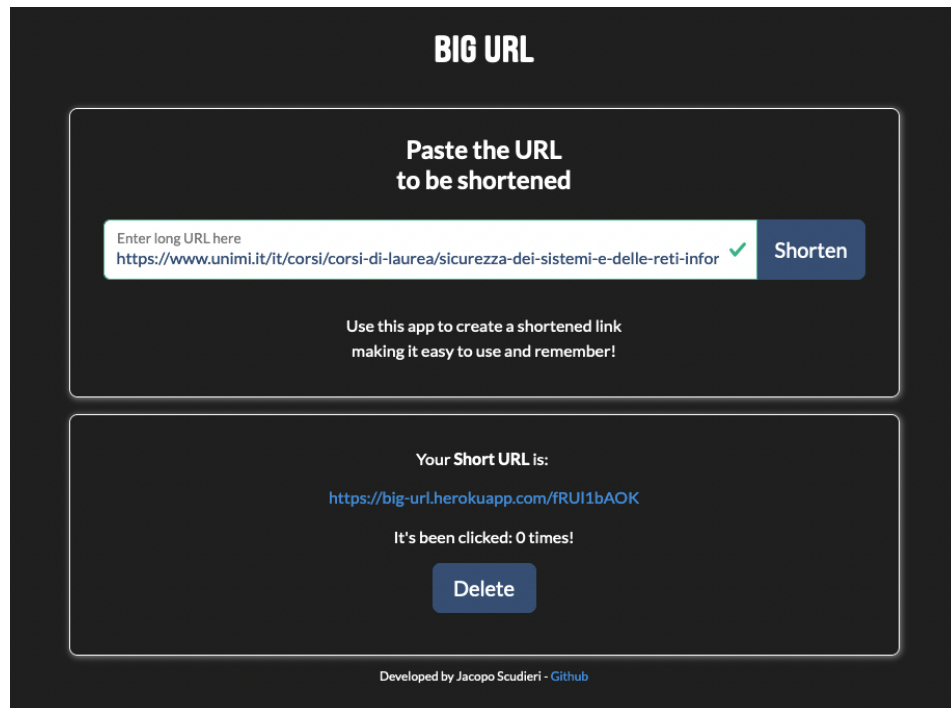
Pagina principale

In questa pagina sono presenti tutti gli elementi necessari per l'utilizzo di Big Url: il box di input dove inserire l'URL originale, il pulsante 'Shorten' per inviare la richiesta e una breve descrizione del servizio.

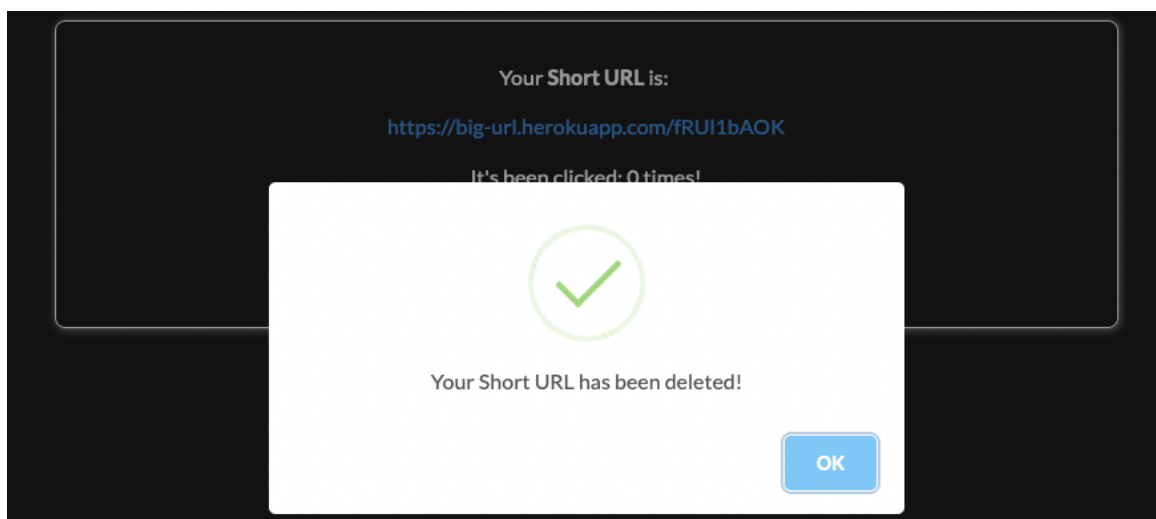


The screenshot shows the 'BIG URL' application interface. At the top, the title 'BIG URL' is displayed in white on a dark background. Below it, a central white box contains the text 'Paste the URL to be shortened'. Inside this box, there is a text input field with the placeholder 'Enter long URL here' and a green checkmark icon. The input field contains the URL 'https://www.unimi.it/it/corsi/corsi-di-laurea/sicurezza-dei-sistemi-e-delle-reti-infor'. To the right of the input field is a blue button labeled 'Shorten'. Below the input field, there is a message: 'Use this app to create a shortened link making it easy to use and remember!'. At the bottom of the white box, it says 'Developed by Jacopo Scudieri - Github'.

Al caricamento dell'applicazione viene mostrata unicamente questa sezione mentre, nel momento in cui viene premuto il pulsante, se è stato inserito un URL valido nel box di input (validato dalla specifica classe CSS per le form validation ".was-validated" di Bootstrap che si avvale di due pseudo classi ":invalid" e ":valid" applicate all'elemento di input) viene mostrata un'ulteriore sezione con i risultati dell'operazione, generata dinamicamente.



In questa nuova sezione sono presenti quindi i dati dell'istanza appena creata: short URL cliccabile e il numero di click eseguiti su quel determinato indirizzo. Inoltre viene visualizzato il pulsante 'Delete' che permette all'utente di eliminare dal database i dati inseriti con la relativa conferma.



Pagina 404

Questa pagina viene visualizzata se l'utente cerca di utilizzare uno short URL errato o non più presente nel database, dandogli la possibilità di tornare alla pagina principale tramite il pulsante 'Go Back'.



Architettura

L'applicazione è formata dalle seguenti risorse:

- Index.html, ovvero la pagina iniziale mostrata all'utente nel momento in cui accede all'applicazione e dalla quale può inserire l'URL originale da restringere.
- 404.html, la pagina mostrata in caso di richiesta ad uno short URL non esistente.
- urls.js, API REST che genera lo short URL in base ai dati inseriti dall'utente e all'ID generato casualmente.
- index.js, API REST che si occupa di effettuare il redirect all'URL originale quando l'utente richiede (GET Request) l'URL abbreviato.
- del.js, API REST per eliminare l'istanza legata allo short URL richiesta dall'utente.
- db.js, per il collegamento con il database MongoDB
- Url.js per la definizione dello Schema (modello) del database

Diagramma dell'ordine di utilizzo dell'applicazione

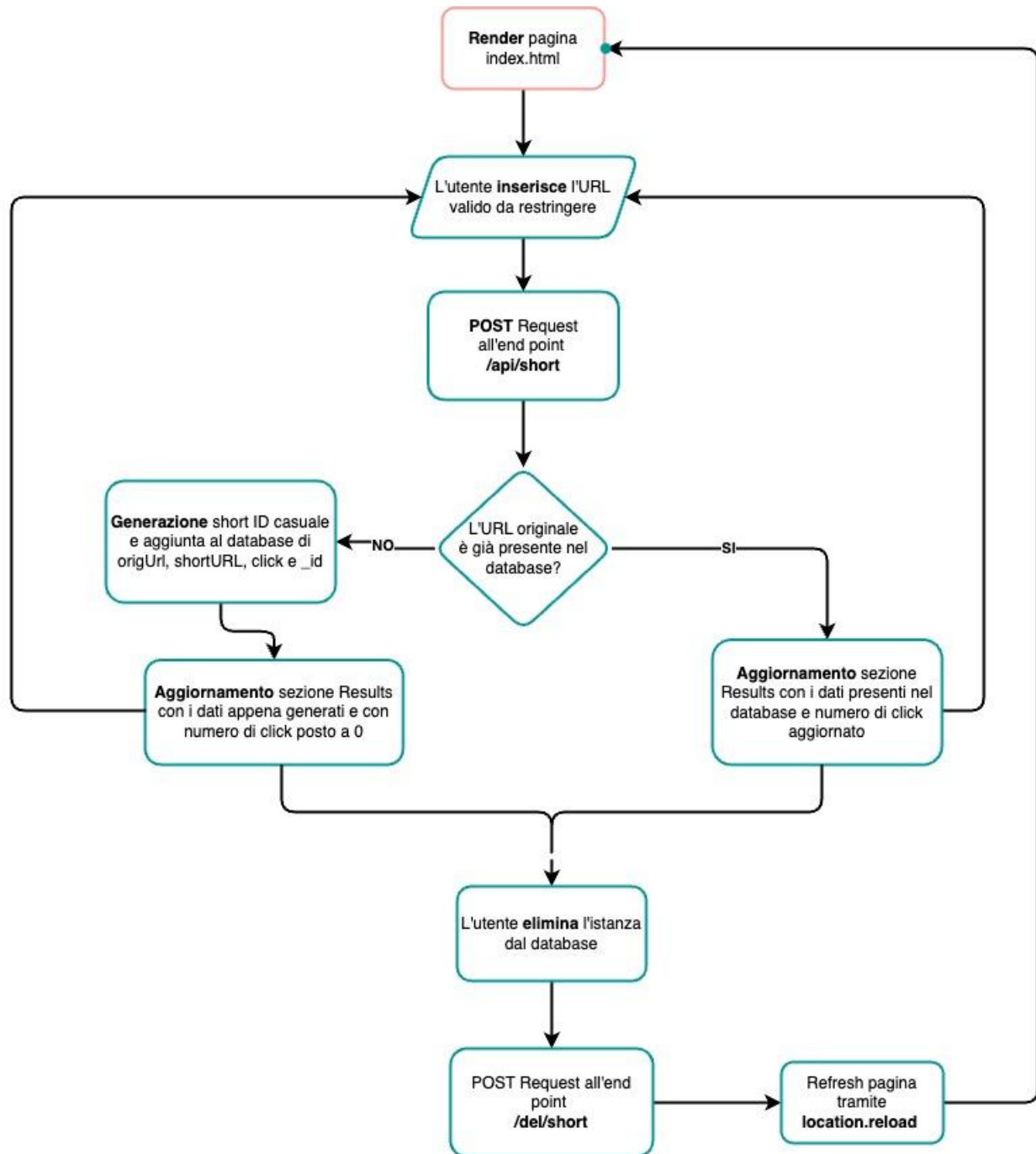


Diagramma descrittivo della risorsa Urls (generazione short URL)

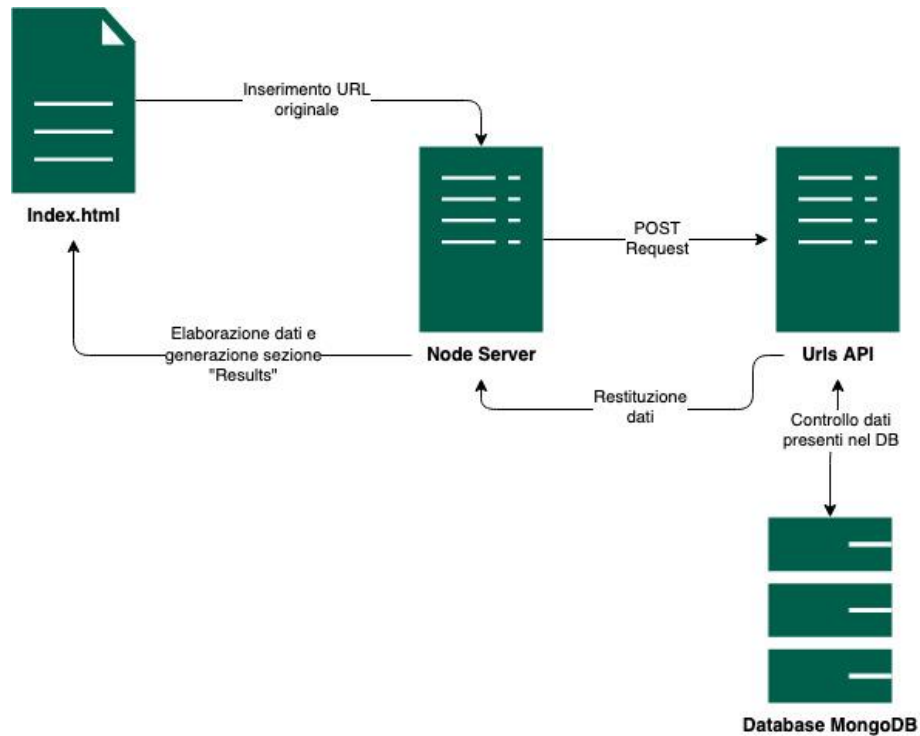


Diagramma descrittivo della risorsa Index (redirect all'URL originale)

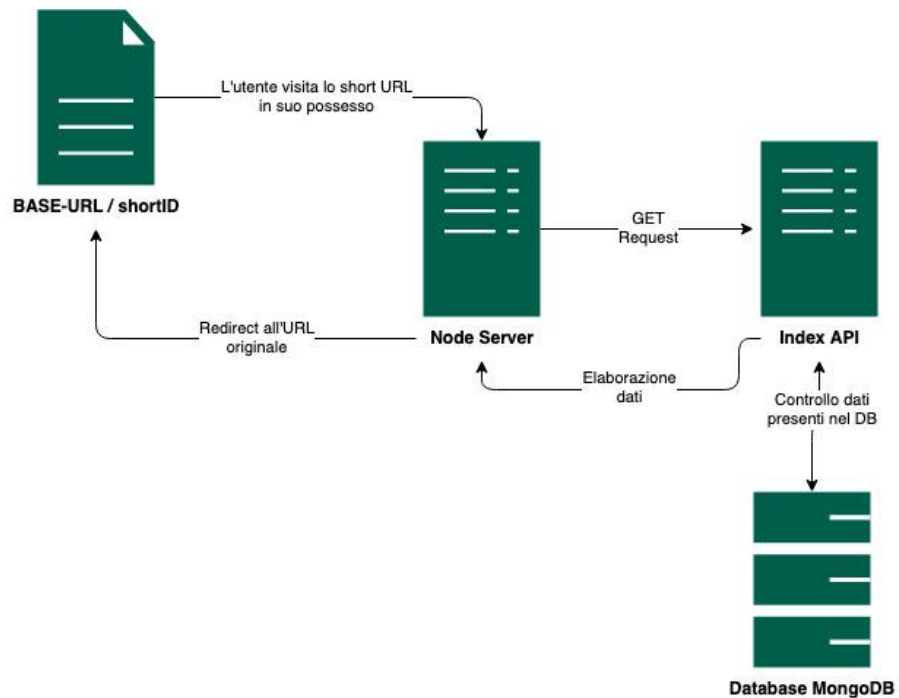
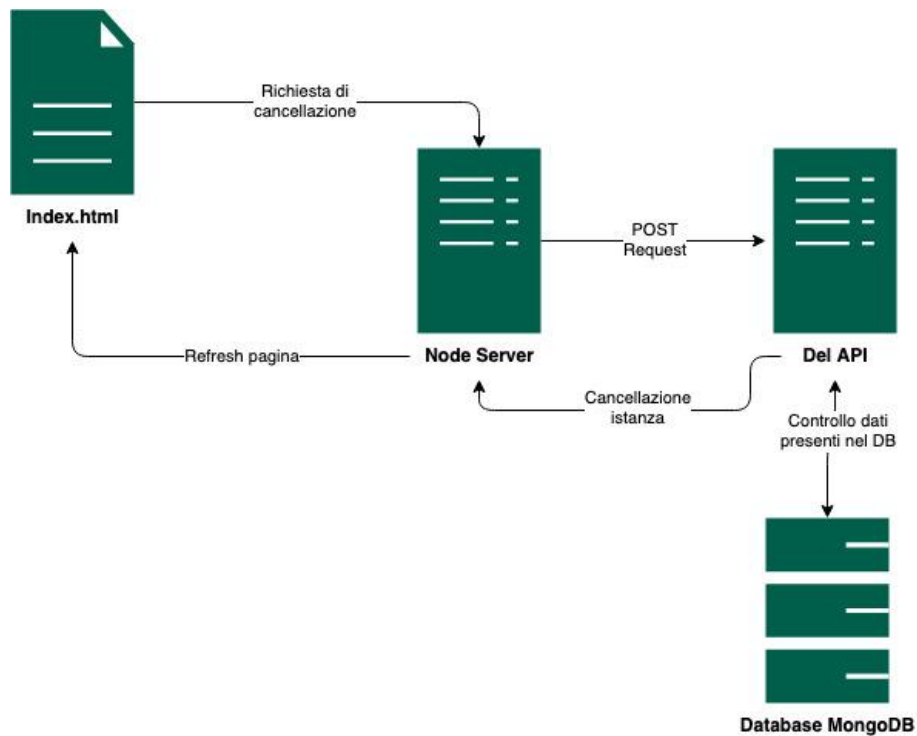


Diagramma descrittivo della risorsa Del (cancellazione short URL)



Esempio istanza database

```
_id: ObjectId('6348670faf1889262eb5a971')
urlId: "1RuU-ByFf"
origUrl: "https://www.unimi.it/it/corsi/corsi-di-laurea/sicurezza-dei-sistemi-e-..."
shortUrl: "https://big-url.herokuapp.com/1RuU-ByFf"
clicks: 0
__v: 0
```

Codice

jQuery

Per l'invio del modulo contenente l'URL originale viene utilizzato un semplice script jQuery che permette l'invio contestuale della POST Request all'API e il riempimento dei DIV relativi con i dati ottenuti.

```
views > assets > script > JS script.js > ...
1  // Attach a submit handler to the form
2  $( "#myForm" ).submit(function( event ) {
3      // Stop form from submitting normally
4      event.preventDefault();
5
6      let url = '/api/short';
7      let origUrl = $("#origUrl").val();
8
9      // Send the data
10  var posting = $.post( url, { origUrl }, function( data ) {
11      $( "#short" ).empty().append( data.shortUrl );
12      $( "#short" ).attr("href", () => {
13          return data.shortUrl;
14      })
15      $( "#click" ).empty().append( "It's been clicked: " + data.clicks + " times!" );
16      $( "#id" ).empty().append(data._id);
17      $( "#results" ).removeClass("visually-hidden");
18  } );
19  });
```

Server Node.js e framework Express.js

Il motore principale dell'applicazione è il Controller, implementato utilizzando il framework Express.js ed eseguito all'interno di un server Node.js. Si occupa di gestire le richieste provenienti dagli utenti, effettuare le richieste alle API, elaborare i dati e generare le pagine da restituire poi agli utenti.

```
// Body Parser to extract the entire body portion of a request and expose it on req.body
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

//Setting access to /views folder
app.use(express.static(__dirname + '/views'));

//Importing routes
app.use('/', require('./routes/index'));
app.use('/api', require('./routes/urls'));
app.use('/del', require('./routes/del'));

//Rendering 'index.html' when a GET request is received
app.get('/', (req, res) => {
  res.render('index');
});

// Server Setup
const PORT = process.env.PORT || 3333;
app.listen(PORT, () => {
  console.log(`Server is running at PORT ${PORT}`);
});
```

Creazione di uno short URL (urls.js)

Nel momento in cui viene effettuata la POST Request all'endpoint '/api/short' viene estratto dal body della View (index.html) l'url originale inserito dall'utente. Se questo è già presente nel database vengono riportati i dati relativi. Altrimenti viene generato un ID casuale che verrà unito all'url base utilizzato dall'app in quell'istante e, insieme all'url originale, verranno inseriti in una nuova istanza del database MongoDB collegato.

```

8  router.post('/short', async (req, res) => {
9      const { origUrl } = req.body;
10     const base = process.env.BASE;
11
12     const urlId = shortid.generate();
13
14     try {
15         let url = await Url.findOne({ origUrl });
16         if (url) {
17             res.json(url);
18         } else {
19             const shortUrl = `${base}/${urlId}`;
20
21             url = new Url({
22                 origUrl,
23                 shortUrl,
24                 urlId,
25             });
26
27             await url.save();
28
29             res.json(url);
30         }

```

Redirect all'URL originale (index.js)

Nel caso invece di una GET Request all'endpoint '/shortId' viene immediatamente controllato se lo shortID richiesto è presente nel database e, in caso positivo, viene effettuato il redirect all'url originale corrispondente nell'istanza del database MongoDB. In caso negativo invece viene inviata la pagina 404.html per la notifica dell'errore.

```

5  //Redirecting to the original URL
6  router.get('/:urlId', async (req, res) => {
7      try {
8          const url = await Url.findOne({ urlId: req.params.urlId });
9          if (url) {
10             url.clicks++;
11             url.save();
12             return res.redirect(url.origUrl);
13         } else return res.status(404).sendFile('404.html', { root: './views' });
14     } catch (err) {
15         console.log(err);
16         return res.status(404).sendFile('404.html', { root: './views' });
17     }
18 });

```

Eliminazione di un'istanza (del.js)

Quando l'utente effettua una POST Request tramite il pulsante 'Delete' nella sezione 'Results' viene interrogata l'API predisposta alla cancellazione di un'istanza presente nel database. Viene quindi innanzitutto controllato se esiste un'istanza con '_ID' richiesto e in caso positivo verrà eliminata.

La pagina verrà poi refreshata automaticamente grazie ad un semplice script jQuery.

```
5 //Deleting the Short URL
6 router.post('/short', async (req, res) => {
7   const { _id } = req.body;
8   try {
9     let url = await Url.findOne({ _id });
10    if (url) {
11      url.deleteOne();
12    } else return res.status(404).sendFile('404.html', { root: './views' });
13   } catch (err) {
14     console.log(err);
15     res.status(500).json('Server Error');
16   }
17 });
```

Pagina Index.html

Contiene il codice HTML5 che definisce le sezioni 'Form' e 'Results' e i DIV al loro interno. Oltre a richiamare gli script necessari ad intercettare gli eventi emessi dai pulsanti.

```
<body>
  <main>
    <header>
      <h1 class="title">Big Url</h1>
    </header>
    <!-- Start: form -->
    <section>
      <div class="divForm d-flex flex-column flex-grow-1 flex-shrink-1 justify-content-center align-content-center">
        <h4 class="divTitle"><strong>Paste the URL<br>to be shortened</strong></h4>
        <form id="myForm" class="was-validated myForm d-flex flex-row flex-grow-1 flex-shrink-1 justify-content-center align-items-center" method="post">
          <div class="form-floating flex-grow-1 flex-shrink-1">
            <input required id="origUrl" class="myInput form-control text-primary" type="url" placeholder="Enter long URL here">
            <label for="origUrl">Enter long URL here</label>
          </div>
          <div class="flex-grow-0 flex-shrink-0">
            <button class="sendButton btn btn-lg btn-primary" type="submit">Shorten</button>
          </div>
        </form>
        <p class="desc text-center">Use this app to create a shortened link<br>making it easy to use and remember!</p>
      </div>
    </section>
    <!-- End: form -->
```

```

<!-- Start: Results -->
<section>
  <div id="results" class="results visually-hidden d-flex flex-column align-items-center">
    <div>
      <h6 id="del" class="del">Your <strong>Short</strong> <strong>URL</strong> is:&nbsp;</h6>
    </div>
    <div>
      <a id="short" class="text-info" href="#" target="_blank"></a>
    </div>
    <div class="d-flex flex-row align-items-center align-content-center">
      <h6 id="click" class="click"></h6>
    </div>
    <h6 id="id" style="display: none;"></h6>
    <button id="delete" class="deleteBtn btn btn-lg btn-primary" type="button">Delete</button>
  </div>
</section>
<!-- End: Results -->
<footer>
  <p class="myFooter">Developed by Jacopo Scudieri &nbsp;<a class="text-info" href="https://github.com/Jot4roKujo" target="_blank">Github</a></p>
</footer>
</main>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js"></script>
<script src="/assets/script/script.js"></script>
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
</body>

```

Database MongoDB

Collegamento al database

Il database cloud fornito da MongoDB viene collegato con l'applicazione tramite il file 'db.js' che sfrutta l'URI presente in '.env' per stabilire e autenticare la connessione.

```

require('dotenv').config({ path: './config/.env' });

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log('Database Connected');
  } catch (err) {
    console.error(err.message);
    process.exit(1);
  }
};

module.exports = connectDB;

```

Creazione dello Schema utilizzato per creare le istanze

Nel file 'Url.js' viene invece creato lo Schema utilizzato da MongoDB grazie al package Mongoose opportunamente importato.

```
models > JS Url.js > ...
1  const mongoose = require('mongoose');
2
3  const UrlSchema = new mongoose.Schema({
4    urlId: {
5      type: String,
6      required: true,
7    },
8    origUrl: {
9      type: String,
10     required: true,
11   },
12   shortUrl: {
13     type: String,
14     required: true,
15   },
16   clicks: {
17     type: Number,
18     required: true,
19     default: 0,
20   },
21 });
22
23 module.exports = mongoose.model('Url', UrlSchema);
```

Conclusioni

La scelta di sviluppare Big Url nasce principalmente dalla curiosità personale di voler approfondire e capire nel dettaglio il funzionamento di un servizio utilizzato innumerevoli volte nel corso degli anni.

Questa curiosità si è poi trasformata nella volontà di creare delle API che sfruttassero i concetti teorici imparati a lezione oltre allo sviluppo di un'interfaccia responsive in grado di permettere la migliore user experience ai destinatari di questa semplice ma potente applicazione.

Il risultato è sicuramente un'applicazione allo stato embrionale e, come già detto, sarebbero innumerevoli le features meritevoli di essere implementate. In ogni caso è usufruibile nella sua totalità da un ipotetico utente che voglia rendere i suoi URL più gradevoli.

Risorse e sitografia

[Big Url](#) su Heroku

[GitHub](#) con il codice integrale

[Bootstrap 5.0](#)

[jQuery](#)

[Node.js](#) ed [Express.js](#)

[MongoDB](#)

[Mozilla Developer Network](#)

[W3Schools.com](#)