

Mini guida - CMS Brescia Musei

Installazione:

Repository: [stefanopoma-97/cms_brescia_musei \(github.com\)](https://github.com/stefanopoma-97/cms_brescia_musei)

Per l'installazione seguire passo passo tutti i passaggi riportati di seguito (o riportati nel README della repository). Il progetto è stato testato in ambiente Windows e Mac.

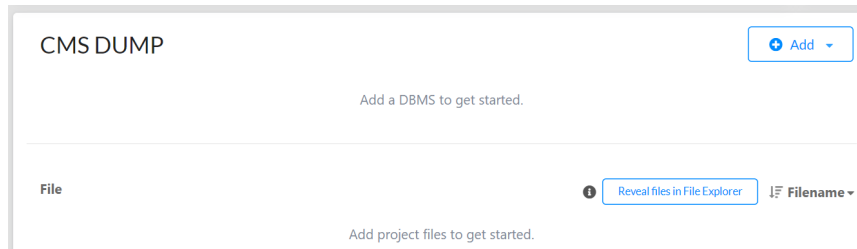
1. Scaricare Java SE 11:
<https://www.oracle.com/java/technologies/downloads/#java16>
2. Scaricare Apache NetBeans (o altro IDE):
<https://netbeans.apache.org/download/index.html>
3. Scaricare Ampss
4. Scaricare Composer: <https://getcomposer.org/> Questa versione sembra non andare con php 8.X Quindi è meglio scaricare php 7.X
<https://windows.php.net/download#php-7.0> inserire la cartella estratta accanto alla cartella PHP creata da Ampss o Xampp
5. Installare composer e selezionare il file php.exe nella cartella Ampss Il file php.exe deve essere quello associato alla versione 7.X
6. Con il composer installare Laravell: `composer global require "laravel/installer=~1.1"`
7. Eseguire i seguenti comandi nella directory del progetto `composer install`
`composer update`
8. cambia `.env.example` in `.env` (rimuovere la parola `".example"`)
9. Controllare che il contenuto di `.env` sia questo: `APP_NAME=Laravel`
`APP_ENV=local APP_KEY= APP_DEBUG=true APP_URL= http://localhost`

`LOG_CHANNEL=stack`
10. Eseguire i seguenti comandi: `php artisan key:generate` (il primo comando genererà la key nel file `env`)
11. nella directory del progetto lanciare il comando: `composer require laudis/neo4j-php-client`
12. lanciare il comando: `composer require tightenco/ziggy`
(<https://github.com/tighten/ziggy#installation>)
13. per far partire il sito lanciare il comando: `php artisan serve`

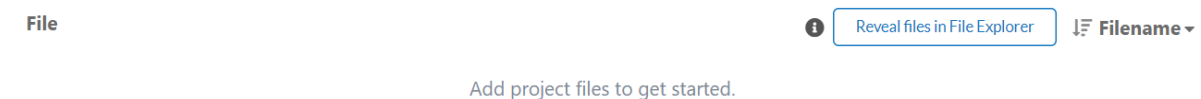
Ripristinare il database dal file .dump

Nella cartella /dump si trova il file: cms-brescia-neo4j-11-ott-2022-10-24-00.dump
Quest'ultimo può essere sfruttato per creare un'istanza di test del database senza doverlo popolare manualmente.

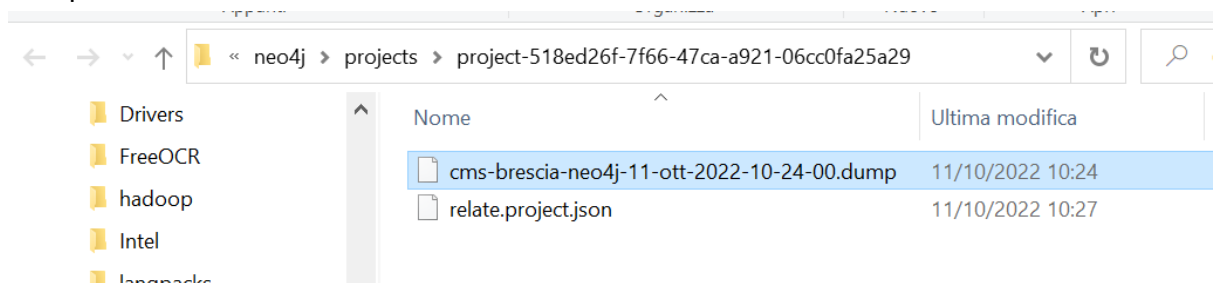
Per prima cosa bisogna creare un nuovo progetto su Neo4j (nell'esempio si sta sfruttando la versione desktop).



Nella pagina che viene presentata bisogna cliccare su “Reveal files in File Explorer”

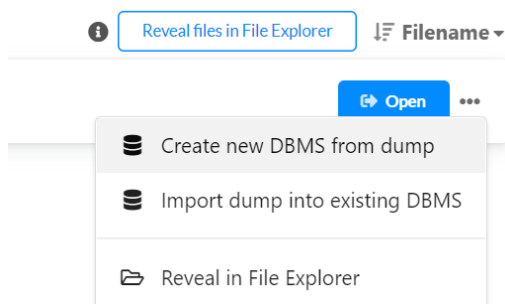


In questo modo si avrà accesso alla cartella del progetto. Nella cartella bisogna copiare il file .dump.



Una volta copiato il file verrà visualizzato anche in Neo4j.

Bisogna ora cliccare sui 3 pallini accanto al file e poi su “Create new DBMS from dump”




Seguirà una procedura guidata in cui è richiesto specificare il nome del dbms (qualsiasi) e la password: neo4j_cms_brescia

Una volta fatto bisogna premere su “Start”

Project 1

+ Add ▾

 cms_brescia_musei 4.4.5

Start


Open ▾


...

Il database viene aperto e automaticamente viene creata l'istanza chiamata “neo4j”. Nel database sono contenute un po' di informazioni utili per testare il funzionamento del sito.

Project 1

+ Add ▾

 cms_brescia 4.4.5 ● ACTIVE

 system

 neo4j (default)

+ Create database

Refresh

Funzionamento del sito

Il sito si apre sulla pagina home.

la scritta “caricamento informazioni...” indica che si è aperta una connessione con il database per recuperare alcuni dati che verranno inseriti nella SESSION.

Tutto viene realizzato con una funzione ajax che non blocca la fruizione della pagina.

Home

Crea un nuovo percorso

Caricamento informazioni ...

Una volta terminato il processo la scritta cambia. Si consiglia di attendere sulla home page qualche secondo affinché tutte le informazioni vengano salvate.

Crea un nuovo percorso

Informazioni caricate

Cliccando sul pulsante si accede alla pagina di creazione di un percorso:

Crea un percorso

[Home](#)

Criterio di raggruppamento:

Qualsiasi



Filtra



Hai selezionato qualsiasi. Verranno mostrate tutte le opere

Nella form si può specificare un criterio di raggruppamento e un parametro (se richiesto).
Cliccando poi su filtra la lista delle opere verrà aggiornata, così come le informazioni evidenziate.

| Titolo | Tipologia | |
|---------------------------|-----------|----------------------------|
| Titolo 5 | Scultura | + Aggiungi |
| Titolo 6 | Scultura | + Aggiungi |
| Titolo 7 | Scultura | + Aggiungi |
| Titolo 8 | Scultura | + Aggiungi |
| Titolo 9 | Scultura | + Aggiungi |
| Titolo 10 | Scultura | + Aggiungi |

Cliccando sul nome di un'opera si accede ad una pagina con le informazioni di riepilogo della stessa:

Opera

[Home](#) / [Opera](#)

| Titolo 5 |
|--------------------------|
| Tipologia: Scultura |
| Autore: Nome Autore 5 |
| Anno: 1940 |
| Seolo: 20 |
| Luogo: Firenze |
| Numero visite: 1 |
| Tempo visite: 11 secondi |

Cliccando su “Aggiungi” l’opera viene spostata in un secondo elenco. L’elenco viene mantenuto anche se si effettuano altri filtri.

| Titolo | Tipologia | |
|-----------|-----------|-----------------------------|
| Titolo 6 | Scultura | <button>+ Aggiungi</button> |
| Titolo 8 | Scultura | <button>+ Aggiungi</button> |
| Titolo 9 | Scultura | <button>+ Aggiungi</button> |
| Titolo 10 | Scultura | <button>+ Aggiungi</button> |

Aggiunte:

Crea

Titolo

Titolo 5 ×

Titolo 7 ×

Cliccando su “Crea” (bottone cliccabile solo se sono selezionate delle opere) si accede alla pagina di conferma percorso.

Conferma percorso

[Home](#)

Titolo del percorso:

Descrizione percorso:

ConfermaAnnulla

| Titolo | Tipologia | Autore | Anno | Secolo | Luogo |
|----------|-----------|---------------|------|--------|---------|
| Titolo 5 | Scultura | Nome Autore 5 | 1940 | 20 | Firenze |
| Titolo 7 | Scultura | Nome Autore 7 | 1955 | 20 | Roma |

Una volta specificato titolo e descrizione (obbligatori) si può confermare la creazione.

Percorso creato

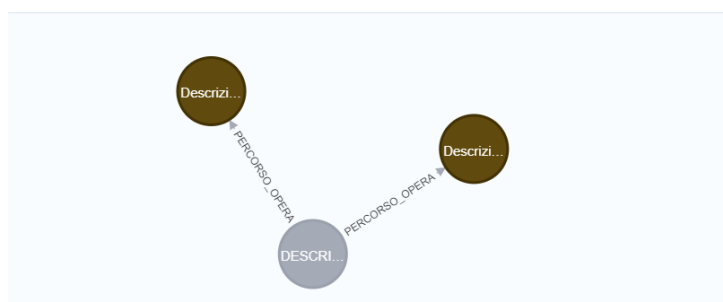
[Ritorna alla Home](#)

Su Neo4j utilizzando questa query:

“MATCH p=()-[r:PERCORSO_OPERA]->() RETURN p LIMIT 25”

si possono vedere tutte le relazioni “PERCORSO_OPERA” e tutti i nodi collegati.

```
MATCH p=()-[r:PERCORSO_OPERA]->() RETURN p LIMIT 25
```



Comunicazione con Neo4j

La gestione della comunicazione con il database si trova nel file DataLayer.php

Al suo interno viene creata la comunicazione con il database sfruttando il progetto "neo4j_php_client" [neo4j-php/neo4j-php-client: Php client and driver for neo4j database \(github.com\)](https://github.com/neo4j-php/neo4j-php-client)

```
$builder = ClientBuilder::create();  
// A client manages the drivers as configured by the builder.  
$this->client = $builder  
    ->withDriver('bolt', 'bolt://neo4j:neo4j_cms_brescia@localhost') // creates a bolt driver  
    ->withDefaultDriver('bolt')  
    ->build();
```

Per lanciare una query va creata una stringa contenente la query Cypher:

```
$results = ($this->client->run('MATCH (o:Opera)  
    WITH o.id as id, o.titolo as titolo, o.autore as autore, o.tipologia as tipologia, o.anno as an  
    RETURN id, titolo, tipologia, autore, anno, secolo, luogo, visite, tempo, per_categoria, per_eta  
    ORDER BY id ASC'));  
  
return $results;
```

Le query possono utilizzare al loro interno anche delle variabili php (interi, stringhe, array ecc.). Nell'esempio sottostante viene inserito nella query un array di interi.

```
$results = ($this->client->run('MATCH (o:Opera)  
    WITH o.id as id, o.titolo as titolo, o.autore as autore, o.tipologia as tipologia, o.anno  
    WHERE id IN $int_array  
    RETURN id, titolo, tipologia, autore, anno, secolo, luogo, visite, tempo, per_categoria, pe  
    ORDER BY id ASC', ['int_array' => $int_array]));
```

Le query non necessariamente devono restituire qualcosa, si può quindi utilizzare tutte le query di eliminazione e modifica previste nella sintassi Cypher:

```
//creo relazioni  
($this->client->run('MATCH (p:Percorso)  
    MATCH (o:Opera)  
    WHERE o.id in $int_array AND p.id = $id  
    MERGE (p)-[:PERCORSO_OPERA]->(o)', ['int_array' => $int_array, 'id'=>$id]));
```

Le query restituiscono delle strutture dati chiamate CypherMap (*\Laudis\Neo4j\Types\CypherMap). I commenti ad ogni funzione realizzata rendono comunque abbastanza chiaro quale sia l'output.

Al fine di osservare la struttura di una CypherMap è possibile stamparla una volta ritornata ad un controller, grazie alla funzione dump:

```
$opere_selezionate_array_id = $dl->getIdSelezionate($opere_selezionate_array);  
// dump($opere_selezionate_array_id);
```

Attualmente tutti i dump sono commentati.