

## Crée un application article

### 1. Pour tout projet avec node js

Il faut crée les **package** cd les npm install

- npm install mongoose --save : pour interagir avec la bdd mongodb
- npm install express ect...

**Après installation des package on peut commencer**

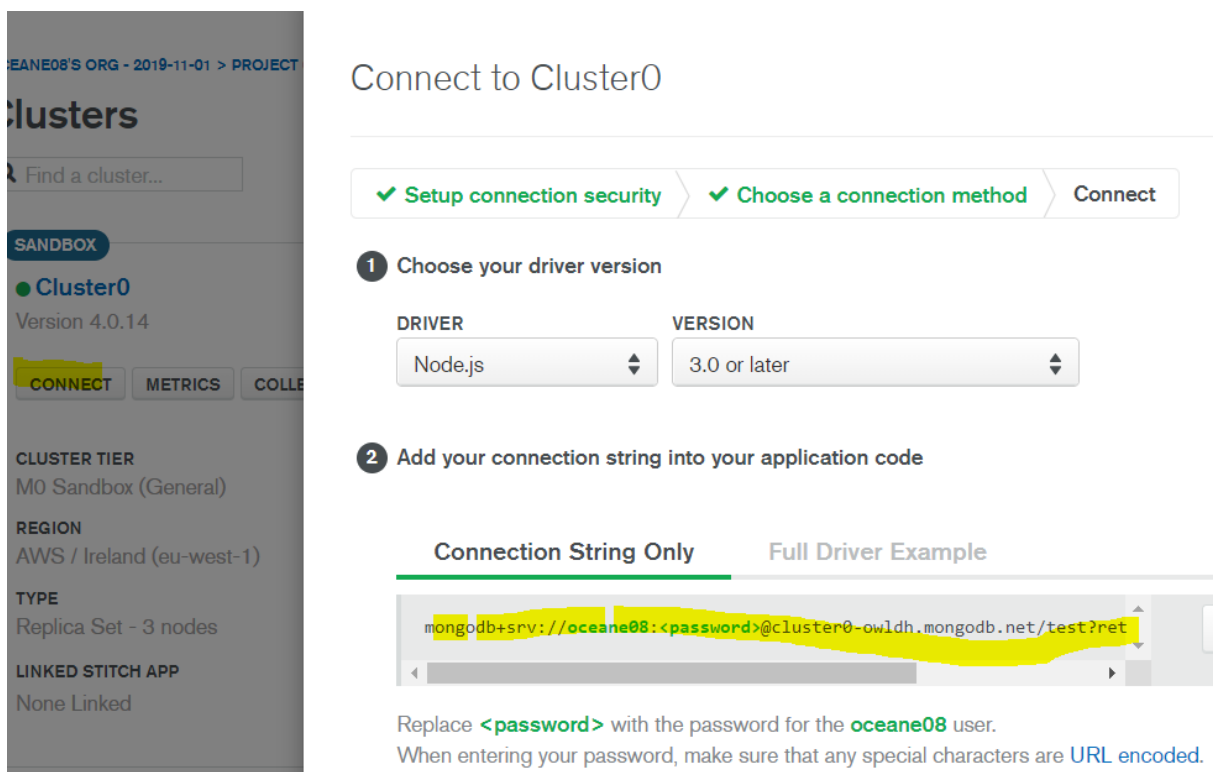
### 2. Crée sont app.js : la base pour lancer son serveur Node

a. D'abord il faut appeler les éléments installer des npm

```
const express = require('express');
const bodyParser = require('body-parser')
const mongoose = require('mongoose');
const fileUpload = require('express-fileupload');
const path = require('path');
```

### b. Puis crée la connexion entre app.js et la basse mongodb.

Pour cela il faut crée sa base de données : rdv sur mongodb atlas et recup le lien en changeant les champs



The screenshot shows the MongoDB Atlas 'Connect to Cluster0' page. On the left sidebar, under 'CLUSTERS', 'Cluster0' is selected with version 4.0.14. The 'CONNECT' button is highlighted. The main content area has a progress bar with 'Setup connection security' and 'Choose a connection method' completed. Step 1, 'Choose your driver version', shows 'Node.js' selected for the driver and '3.0 or later' for the version. Step 2, 'Add your connection string into your application code', shows two tabs: 'Connection String Only' and 'Full Driver Example'. The 'Connection String Only' tab is active, displaying the connection string: `mongodb+srv://oceane08:<password>@cluster0-owldh.mongodb.net/test?ret`. Below the string, instructions state: 'Replace <password> with the password for the oceane08 user. When entering your password, make sure that any special characters are URL encoded.'

Nom bdd

mongodb+srv://ocean08:<password>@cluster0owldh.mongodb.net/test?retryWrites=true&w=majority

```
mongoose.connect("mongodb+srv://ocean08: <password> @cluster0-owldh.mongodb.net/nomdbb?retryWrites=true&w=majority", {useNewUrlParser: true},() => console.log("BDD CONNECTER") );
```

**c. Puis dire au serveur qu'on utilise certaine ressource avec les app.use et app.set pour indiquer qu'on utilise express**

```
app.set('views', __dirname + '/views');
app.set('views engine','ejs');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}))
```

**d. Crée ces middlewares**

Importer d'abord le fichier route liée à app.js

Remarque :ne pas oublier de crée le fichier correspondant

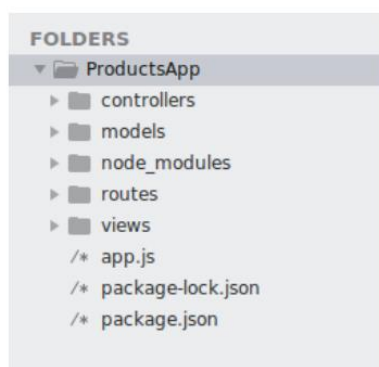
```
const user = require('./routes/user');

app.use('/user', user)
```

Avec le app.use on indique après la /... il doit utiliser cette route dans url

Ex : localhost :3000/user

```
app.listen(3000, () => {
});
```



App structure

Structure du CRUD

- App.js liée vers routes
- Routes envoie vers les Controller
- Controller exécute action et redirige le résultat vers le font end les views

### 3. Crée son modèle :

Il permet de créer et de lire les documents à partir d'un schéma

```
//Importation du module mongoose
const mongoose = require("mongoose");

//Création du schema

const Article = mongoose.Schema({
  titre: { type: String, required: true }, //required :champs requis
  contenus: { type: String, required: true },
  parution: { type: Date, default: Date.now },
  photo: String,

});

module.exports = mongoose.model("articles", Article);
//Nom de la collection > articles
```

➔ Donnée un nom au schéma : ex Article

Le schéma va créer le document avec les champs suivants

➔ Exportation du schéma en modèle :

Elle va créer le document dans la collection suivantes : « articles »

Structure de MongoDB :



### 4. Routeur

```
//Importation de express
var router = require('express').Router();

//Importation du controllers
var ajoutController = require('../controllers/indexController.js')

//Action
router.get('/', ajoutController.ajout);
router.post('/add', ajoutController.save);

//Importation de la route
module.exports = router;
```

D'abord on appelle le module express

Puis on importe la fichier lien au routeur dans exemple le fichier qui correspond est dans le dossier controllers -> nom du fichier indexController.js dans mon repo

Action qui va emprunter la route pour effectuer son action

**Get** : est utilisé pour demander des données à une ressource spécifiée qui utilise route racine « / »

**Post** : est utilisé pour envoyer des données à un serveur pour créer / mettre à jour une ressource qui utilise la route « /add »

**Remarque : ne pas confondre les routes du routeur aux routes utiliser dans app.js**

### 5.Crée son controller :

- Structure de base d'un controllers :
- Importation mongoose
- Importation du modèle
- Connexion à la bdd
- Action du controller
- Exporter le controller
- 

```
//Connexion bdd mongoclient
var mongoose = require('mongoose');
const Article=require("../models/shemas");

var controller = {};

mongoose.connect("mongodb+srv://oceane08:password@cluster0-
owldh.mongodb.net/articles?retryWrites=true&w=majority",
{useNewUrlParser: true},() =>

console.log("BDD CONNECTER")
);
```

```
controller.index = (req, res) => {
  //Chemin vers l'affichage du formulaire
  ACTION -----
};
```

```
//Important pour export
module.exports = controller;
```

## 6.Views : template EJS gestion de l'affichage des données du front

### Syntaxe ejs <%...%>

```
//Route de app.js pour barre de nav
<a href="/">Notre page article</a>
<a href="/ajout-article">Formulaire </a>
<a href="/user">Connexion </a>
```

Crée une boucle pour afficher les données de la bdd dans un tableau par exemple

```
<% if (articles) { %>

//nom_variable.name champs du formulaire
<h4><%= articles.titre %></h4>

<% } %>
```

### Formulaire envoi données

//action : routes app.js (cf.2.d) + action du routeur (cf.4)

```
<form method="POST" role="form" action="/ajout_article/add" enctype="multipart/form-data" >
```

Modif donnée :

```
//Routeur
router.get('/edit/:id', indexController.edit);
```

```
//Controller
controller.edit = (req, res) => {

  Article.findById(req.params.id, function (err, article) {

    res.render('form_edit.ejs', {article:article}); ->article nom donnée

  });
}
```

//action : routeur index.js + nom-variable+id correspond à l'article

```
//Ejs views
<form method="POST" role="form" action="/edit/<%= article.id %>">
```