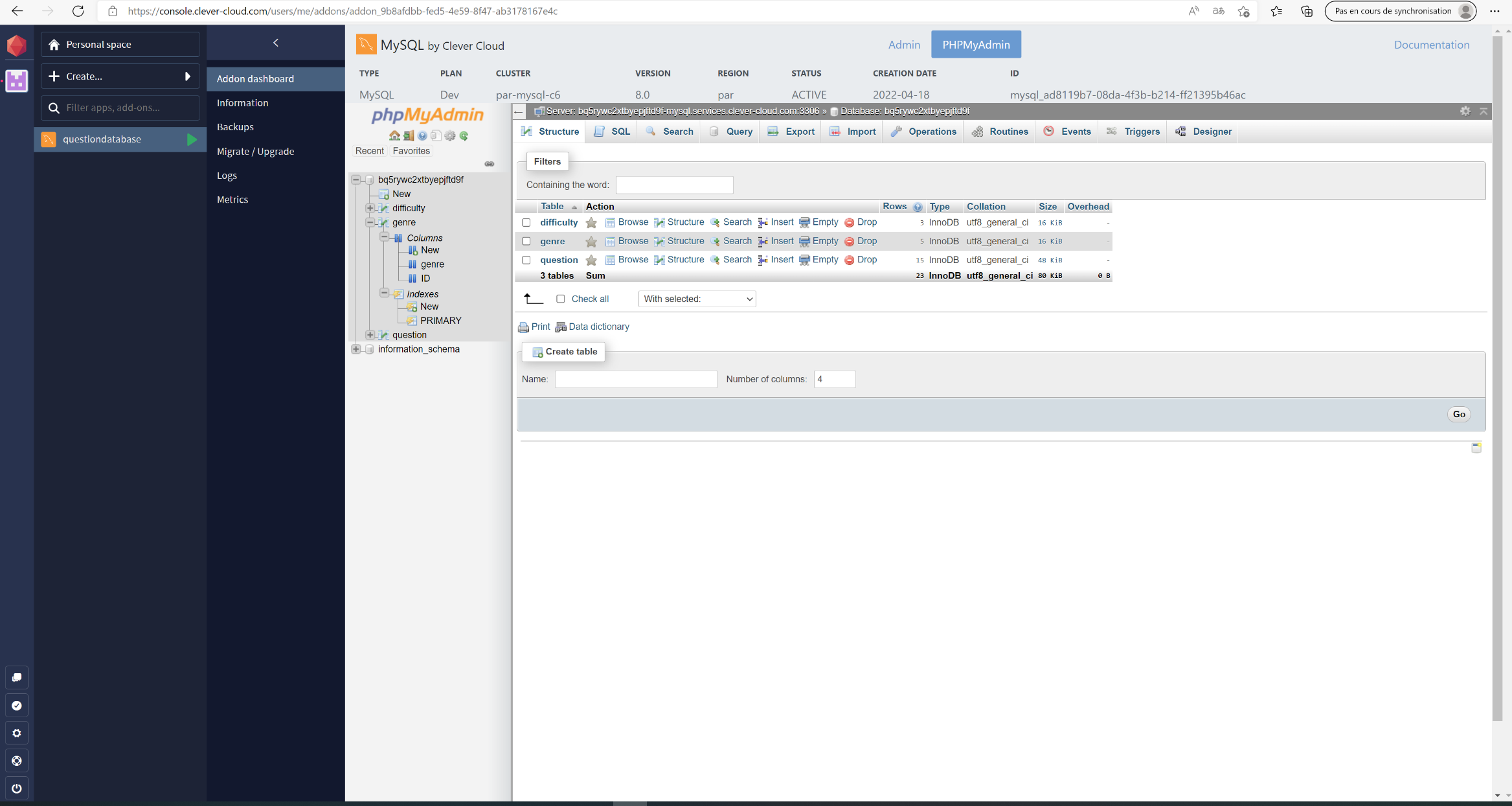
* **Base de données :**

1. Hébergement de la base de données en ligne (Clever Cloud : [Home - Clever Cloud (clever-cloud.com)](https://www.clever-cloud.com/) ). MySQL géré avec un client PHPmyAdmin.
2. Remplissage de la BDD (+50 questions)



* **Backend :**

1. Mini affichage de la base de données avec Express.js et EJS :
2. Express.js, qu’est-ce ? Express.js est un framework node.js. Pourquoi ? Car il est populaire, très bien documenté et flexible (notamment avec les nombreuses librairies disponibles sur npm).
3. EJS, qu’est-ce ? Embedded JavaScript templating est une librairie de npm qui permet de générer simplement du HTML à partir du code javascript. Pourquoi ? Car il est pratique lorsqu’on a beaucoup de de données, typiquement ici c’est le cas avec notre base de données.

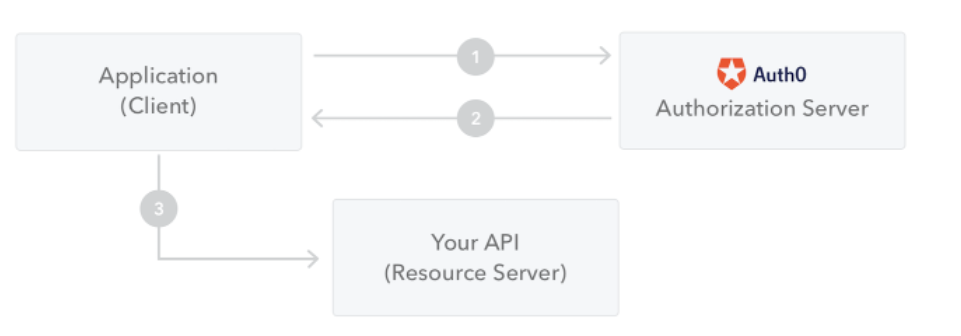
On pense donc partir sur Express.js pour faire tout notre backend car il nous permet de récupérer facilement les données de notre BDD et est aussi compatible avec nos options de sécurité.

Une image contenant table

Description générée automatiquement

* **Sécurité :**

1. API Jwts (auth0, frontegg) : C’est la solution la plus simple pour notre sécurité. L’API auth0 par exemple est simplement une API que l’on va appeler dans notre backend, nous n’avons pas besoin de stocker les login/mdp. L’API va nous générer un token JSON qui va pouvoir donner l’accès ou non aux données de l’app après vérification.



1. Chiffrement : Ici on va encrypter les données de l’utilisateurs dans une base de données et donner l’accès ou non a l’app après vérification des tuples.
2. Faire notre propre API : Ici on va stocker les login/mdp puis après générer de nous même un token JSON pour l’authentification. Le point fort est que nous sommes beaucoup plus libres dans notre interface en front.

Nous pensons partir sur le 3) même si cela n’est pas le plus simple pour pouvoir gérer plus librement notre application et contrôler nous-même la confidentialité de notre application.

**Attention**, il nous faut donc faire attention aux injections SQL. Pour cela nous pensons contrer ceci avec du place holder dans la fonction qui gère les requêtes SQL.

**LIENS :**

Express.js & EJS :

[*https://expressjs.com/fr/guide/database-integration.html#mysql*](https://expressjs.com/fr/guide/database-integration.html#mysql)

[*https://www.youtube.com/watch?v=FHB48qu7hok*](https://www.youtube.com/watch?v=FHB48qu7hok)

Sécurité :

[*https://talks.freelancerepublik.com/node-js-express-securiser-serveur-back-end/*](https://talks.freelancerepublik.com/node-js-express-securiser-serveur-back-end/)

[*https://jwt.io/introduction*](https://jwt.io/introduction)

[*https://www.sohamkamani.com/nodejs/jwt-authentication/*](https://www.sohamkamani.com/nodejs/jwt-authentication/)

[*https://www.veracode.com/blog/secure-development/how-prevent-sql-injection-nodejs*](https://www.veracode.com/blog/secure-development/how-prevent-sql-injection-nodejs)

Antoine LEURIDAN

Chercheur 21/04/22