# Scénario 2 - Base de données Cassandra

L'objectif est de s'introduire à une base de données *NoSQL*, c'est-à-dire qu'elle n'offre pas de fonctionnalité tel intégrité de clés étrangères et jointure. **Cassandra** offre son propre language, *cql*, qui est très similaire à *SQL*.

Nous allons développer un système de journalisation. Ce genre de système est utilisé pour s'assurer que l'application est fonctionnelle et réagir rapidement lorsque le système de journalisation détecte des **pattern** inhabituelles.

## Environnement de développer

Suivre l'activité pédagogique en classe.

## Backend

### Package

On doit ajouter une dépendance à notre application *NodeJS*.

```
    "cassandra-driver": "^4.8.0"
```

### Modèle

Vous devez créer un modèle `logModels` qui sera responsable de sauvegarder les éléments de journalisation. Vous aurez uniquement besoins de faire deux fonctions : créer un `log` et lire tous les `logs`.

### Contrôleur

Votre contrôleur `logController` doit être en mesure de gérer l'insertion et la lecture des logs dans **cassandra**.

### Routes

Vous devez fournir deux routes, `\logs\` en GET (pour lire tous les logs) et `\logs\` en POST pour créer un `logs` à partir du *frontend*.

### Autres

En plus des routes fournies, nous voulons être en mesure d'ajouter des `logs` à partir du *backend* lorsqu'un `catch` est **triggered**. Ainsi, chaque contrôlleur doit ajouter de l'information pertinentes dans la base de données **cassandra** quand une erreurs est détecté. Enfin, on veut également qu'un `log` soit créer, de type **INFO** pour chaque route demandé.

## Frontend

### Visualiser les logs

Offrir un composante web pour afficher les `logs`.

## Erreur en frontend

Lorsque vous avez une erreur sur le *frontend* (par exemple, un *subscribe* qui ne fournit pas de *email* par exemple), vous devez envoyer un `log` via la route fournie.