TP

Création d'une application de gestion de commandes de pizzas

https://www.jordansablon.fr/enseignement

https://github.com/jsablonEnseignement/tp_orders_sources

Sommaire

Présentation du sujet

L'objectif du TP est de réaliser une application permettant de gérer des commandes de pizzas.

Présentation du sujet

L'application offrira à l'utilisateur la possibilité de consulter les différentes commandes, de les modifier, d'en ajouter de nouvelles et d'en supprimer.

Il sera possible de sélectionner une commande à consulter via une liste déroulante qui contiendra l'ensemble des commandes existantes (affichage par numéro de commande).

Une fois une commande sélectionnée, on affichera l'ensemble de ses informations, à savoir :

- Le numéro de la commande
- Le nom et prénom du client
- La date de commande
- Le statut de la commande (« A traiter », « En cours de traitement », « Terminée »)
- La liste des différentes pizzas présentes dans la commande

Présentation du sujet

Lors de la consultation du détail d'une commande, l'utilisateur pourra ajuster la quantité de chacun des articles soit :

- En cliquant sur le bouton « - » qui permet de réduire d'une unité la quantité
- En cliquant sur le bouton « + » qui permet d'augmenter d'une unité la quantité
- En modifiant la valeur du champ de saisie qui contient la quantité

Chaque ligne peut être retirée (annulation de l'ensemble de la ligne) au clic sur le bouton « X ».

Le prix total de la ligne sera ajusté selon la quantité sélectionnée, ou lors de la suppression d'une ligne, tout comme le prix total de la commande. Ce dernier sera également ajusté lors de l'ajout d'un article.

L'ajout d'un article se fait via un clic sur le bouton « Ajouter un article ». On aura alors accès à l'ensemble des articles éligibles via une liste déroulante.

Dès qu'un article est sélectionné, il est automatiquement ajouté à la liste des articles de la commande.

`<Orders />`

Il s'agit du composant principal de votre application. Il sera utilisé dans le fichier `page.tsx` et contiendra l'ensemble du code de l'application React

`<Order />`

Il s'agit du composant qui permet de modéliser l'affichage d'une commande (c'est-à-dire l'affichage de l'ensemble des informations d'une commande). Ce composant sera appelé depuis le composant `<Orders />`

`<Item />`

Ce composant permet de modéliser l'affichage d'un article de la commande : le titre de l'article, sa description, la possibilité de modifier sa quantité pour la commande, son prix et le prix total en fonction de la quantité sélectionnée. On aura également un bouton permettant de supprimer l'article de la commande.

`<NewOrder />`

Il s'agit du composant qui permet de gérer l'affichage du formulaire de création d'une commande. On pourra y retrouver un champ de saisie pour le nom du client, un autre champ de saisie pour le prénom du client, et un bouton permettant d'ajouter un ou plusieurs articles à la commande.

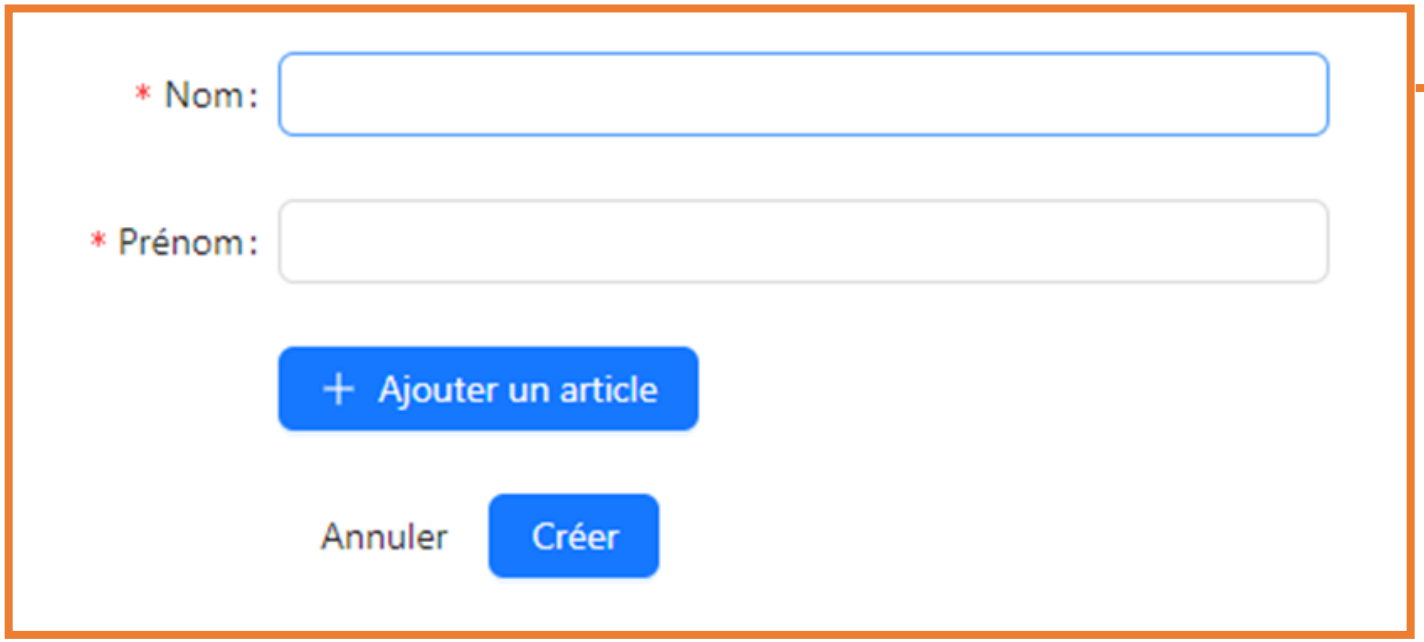
FRONTEND - Suggestions de composants à créer

<Orders />

<Order />

<Item />

FRONTEND - Suggestions de composants à créer



<NewOrder />

Orders

HTTP	URI	Description
GET	/orders	Permet de récupérer l'ensemble des commandes
GET	/orders/*{orders_id}*	Permet de récupérer une commande
POST	/orders	Permet de créer une commande
PUT	/orders/*{orders_id}*	Permet de mettre à jour une commande
DELETE	/orders/*{orders_id}*	Permet de supprimer une commande

Items

HTTP	URI	Description
GET	/items	Permet de récupérer l'ensemble des items
GET	/items/*{item_id}*	Permet de récupérer un item
POST	/items	Permet de créer un nouvel item
PUT	/items/*{item_id}*	Permet de mettre à jour un item
DELETE	/items/*{item_id}*	Permet de supprimer un item

Pour la base de données, vous aurez à créer deux collections :

orders

Cette collection regroupera l'ensemble des informations d'une commande


```
_id: ObjectId('6478a760f4feb3882d7ebdf3')
number: "2"
date: 2023-06-01T12:00:00.000+00:00
status: "done"
▾ client: Object
    name: "DOE"
    firstname: "John"
▾ items: Array
  ▾ 0: Object
      name: "Bacon Groovy"
      description: "Crème fraîche légère française, mozzarella, poulet rôti*, oignons, bac…"
      price: 10.9
      quantity: 1
      _id: ObjectId('6478a760f4feb3882d7ebdf4')
```

Voici les données à insérer dans la collection `orders` avec les types associés

Champs	Type	Description
_id	ObjectId	Identifiant d'un item
number	Number	Numéro de la commande
date	Date	Date de création de la commande
status	String	Statut de la commande (*A traiter, en cours, terminée*)
client	Object	Le client ayant passé la commande
Items	Array	La liste des articles de la commande

Un client se modélise par les données suivantes :

Champs	Type	Description
name	String	Nom du client
firstname	String	Prénom du client

Un item se modélise par les données suivantes :

Champs	Type	Description
_id	ObjectId	Identifiant de l'article
name	String	Nom du l'article
description	String	Description de l'article
price	Number	Prix de l'article
quantity	Number	Quantité de l'article dans la commande

Pour la base de données, vous aurez à créer deux collections :

`items`

Cette collection regroupera l'ensemble des informations d'une commande

```
_id: ObjectId('64799f0807d51c70ed799b2c')
name: "Cannibale"
description: "Sauce barbecue, mozzarella, poulet rôti*, merguez*, haché au bœuf
              assa…"
price: 12.9
```

Voici les données à insérer dans la collection `items` avec les types associés

Champs	Type	Description
_id	ObjectId	Identifiant d'un item
name	String	Nom de l'item
description	String	Description de l'item
price	Number	Prix de l'item

MONGOOSE – Quelques indications pour le schéma de données à créer

Définition d'un attribut « simple » dans un schéma :

```
example: {
    type: String,
},
```

Définition d'un attribut « complexe » (type Object) dans un schéma :

```
example: {
    attribut1: { type: String },
    attribut2: { type: String },
    attribut3: { type: String },
},
```

```
client: {
    id: { type: Types.ObjectId },
    name: { type: String },
    firstname: { type: String },
},
```

MONGOOSE – Quelques indications pour le schéma de données à créer

Il est possible d'extraire les données de l'objet client dans un schéma dédié :

```
const ClientSchema: Schema = new Schema(
  {
    id: { type: Types.ObjectId },
    name: { type: String },
    firstname: { type: String },
  },
  { versionKey: false }
);
```

On peut dès lors l'utiliser dans un autre schéma :

```
client: {
   id: { type: Types.ObjectId },
   name: { type: String },
   firstname: { type: String },
 },
```

```
client: ClientSchema,
```

MONGOOSE – Quelques indications pour le schéma de données à créer

Définition d'un tableau d'attributs « simples » dans un schéma :

```
tableauDeString: {
    type: Array<String>,
  },
```

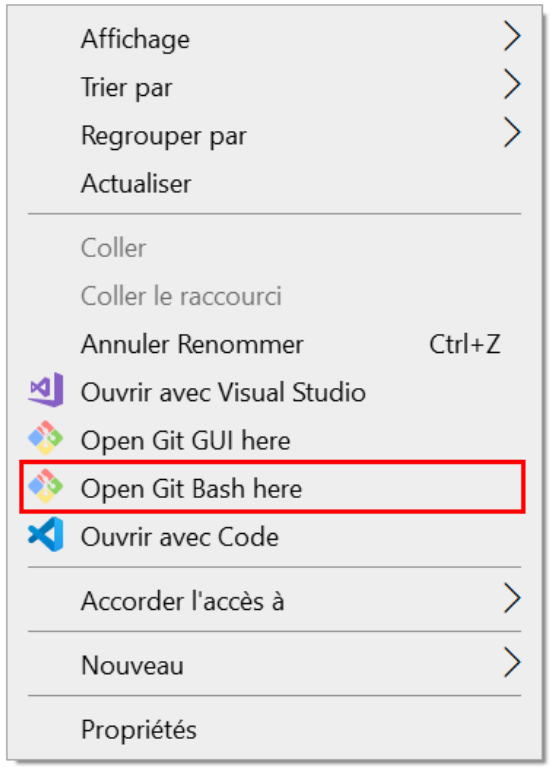
OU

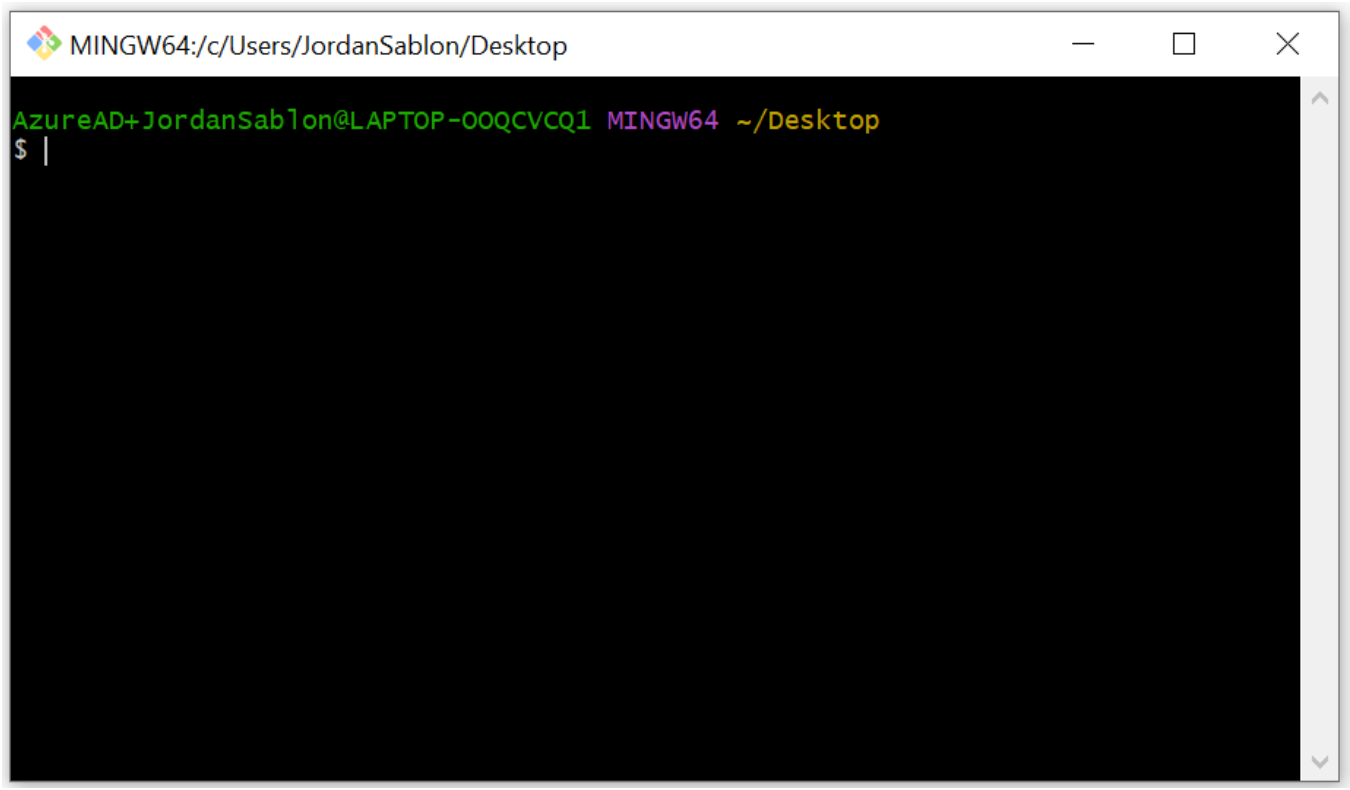
```
tableauDeString : {
    type : [String],
  },
```

Récupération du code source du projet & installation des dépendances

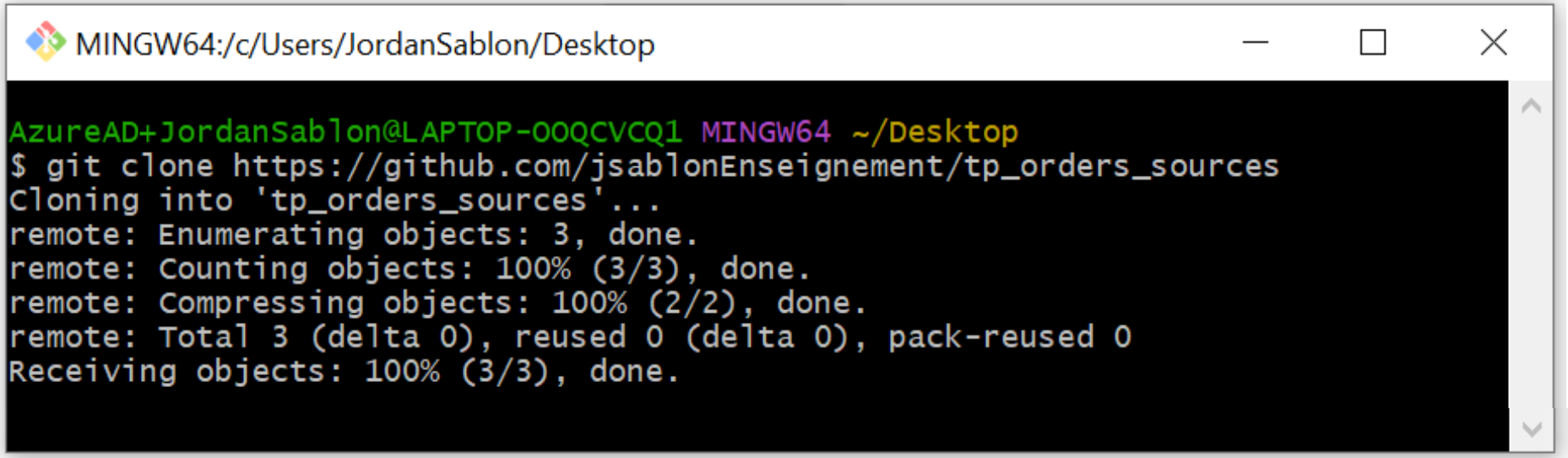
1. Se positionner à l'emplacement de votre choix (où se trouvera le futur dossier du projet)

2. Ouvrir un invité de commande à l'emplacement précédent :

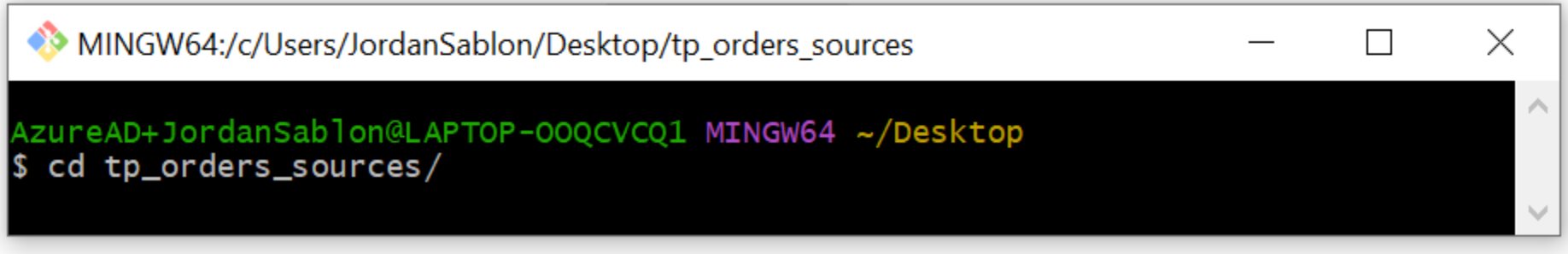




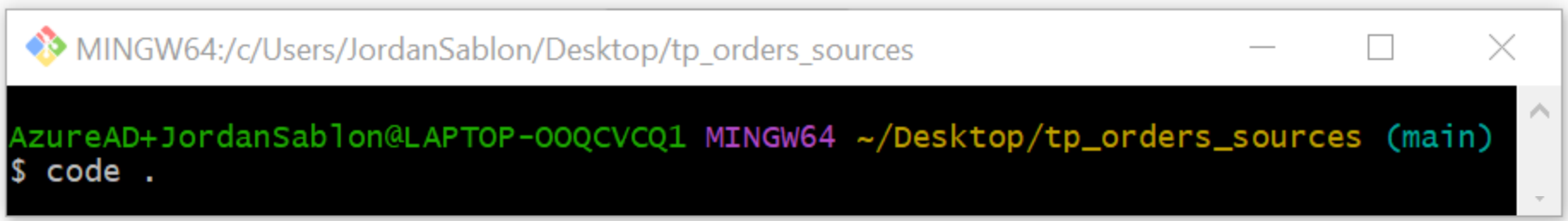
3. Cloner le répertoire Git : https://github.com/jsablonEnseignement/tp_orders_sources

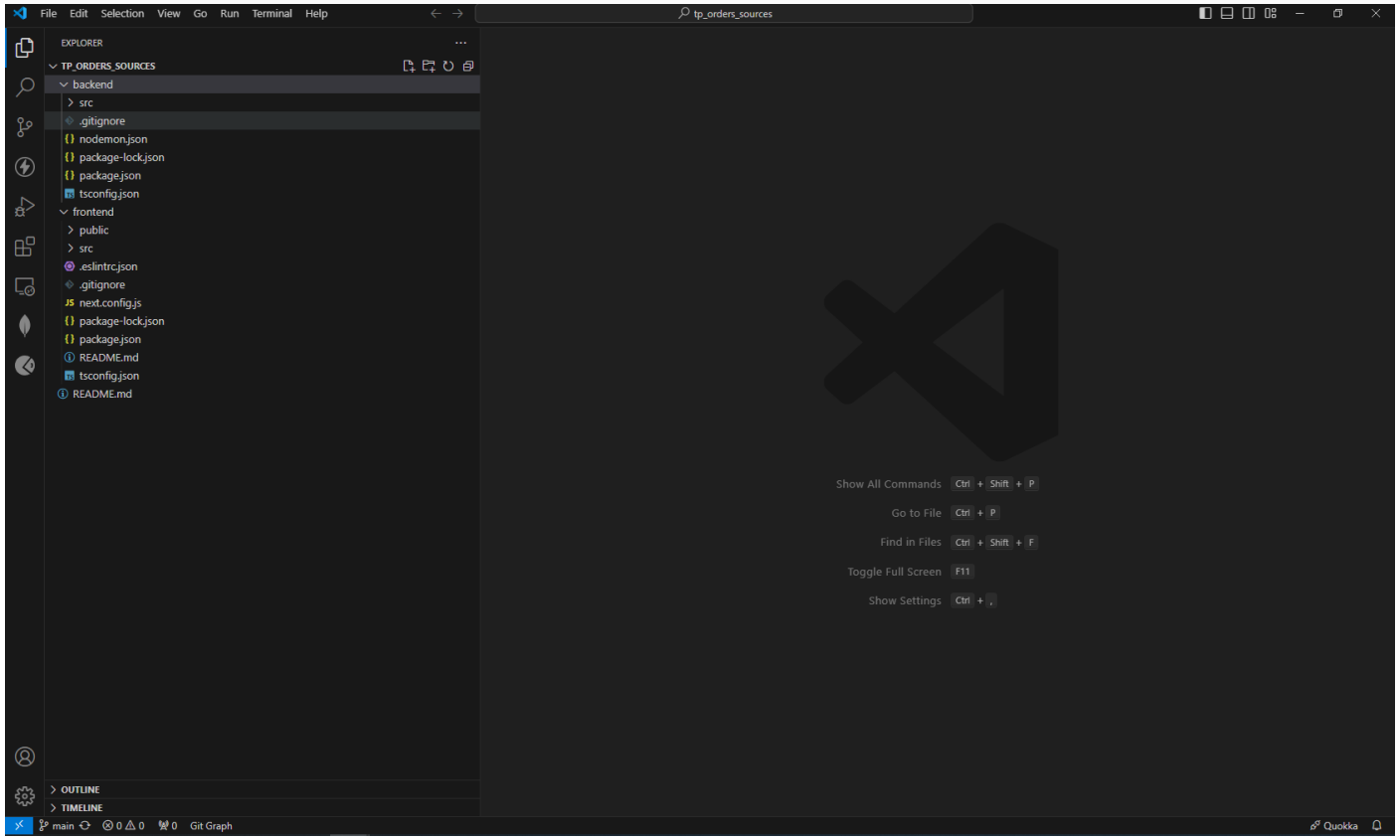


4. Se rendre dans le dossier précédemment cloné :



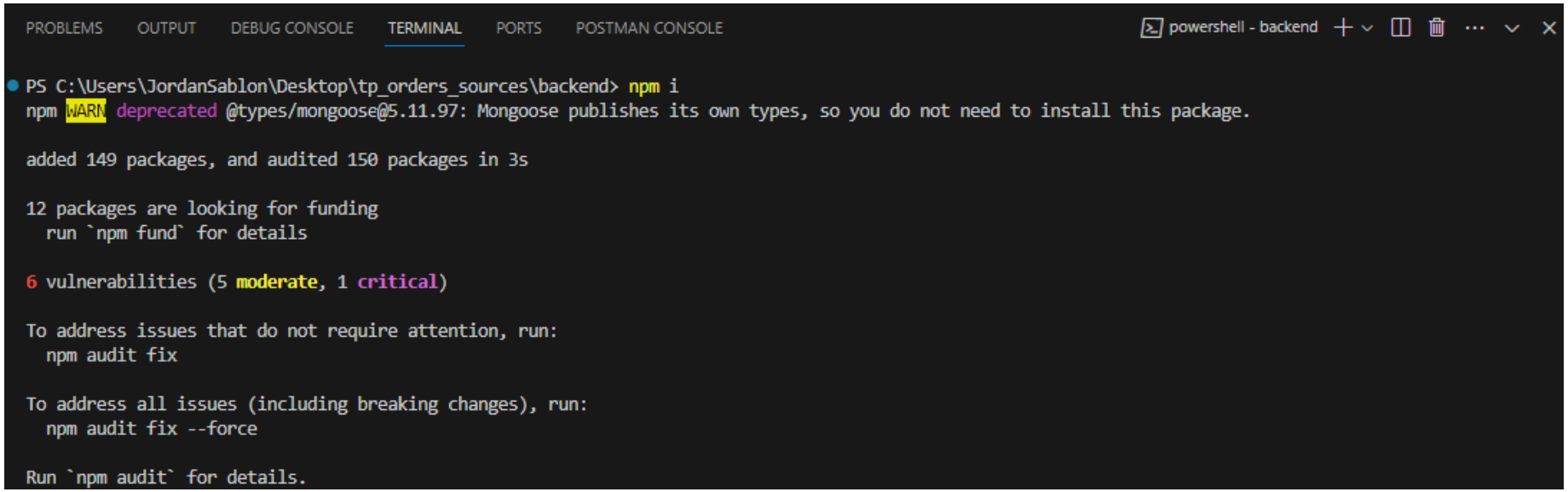
5. Ouvrir le projet dans VSCode :





Récupération du code source du projet & installation des dépendances

Depuis le terminal de VSCode, se rendre dans le dossier **backend** et exécuter la commande suivante :

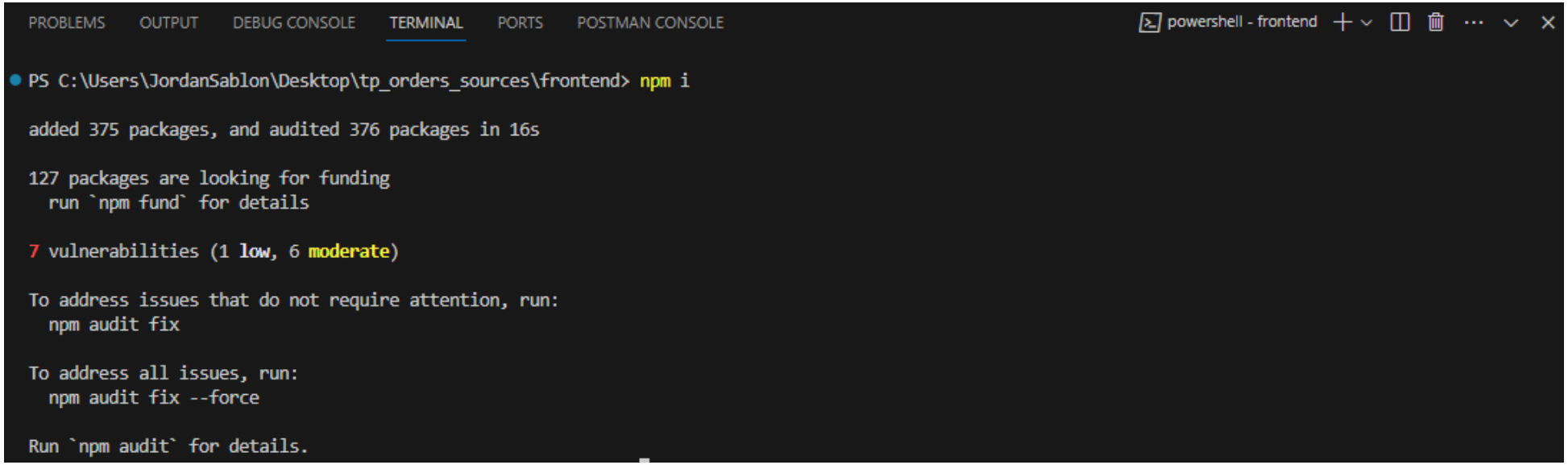


Les dépendances nécessaires au bon fonctionnement de l'application viennent d'être installées.

L'application peut alors être démarrée en exécutant la commande **`npm run start`**

Récupération du code source du projet & installation des dépendances

Depuis le terminal de VSCode, se rendre dans le dossier **frontend** et exécuter la commande suivante :



Les dépendances nécessaires au bon fonctionnement de l'application viennent d'être installées.

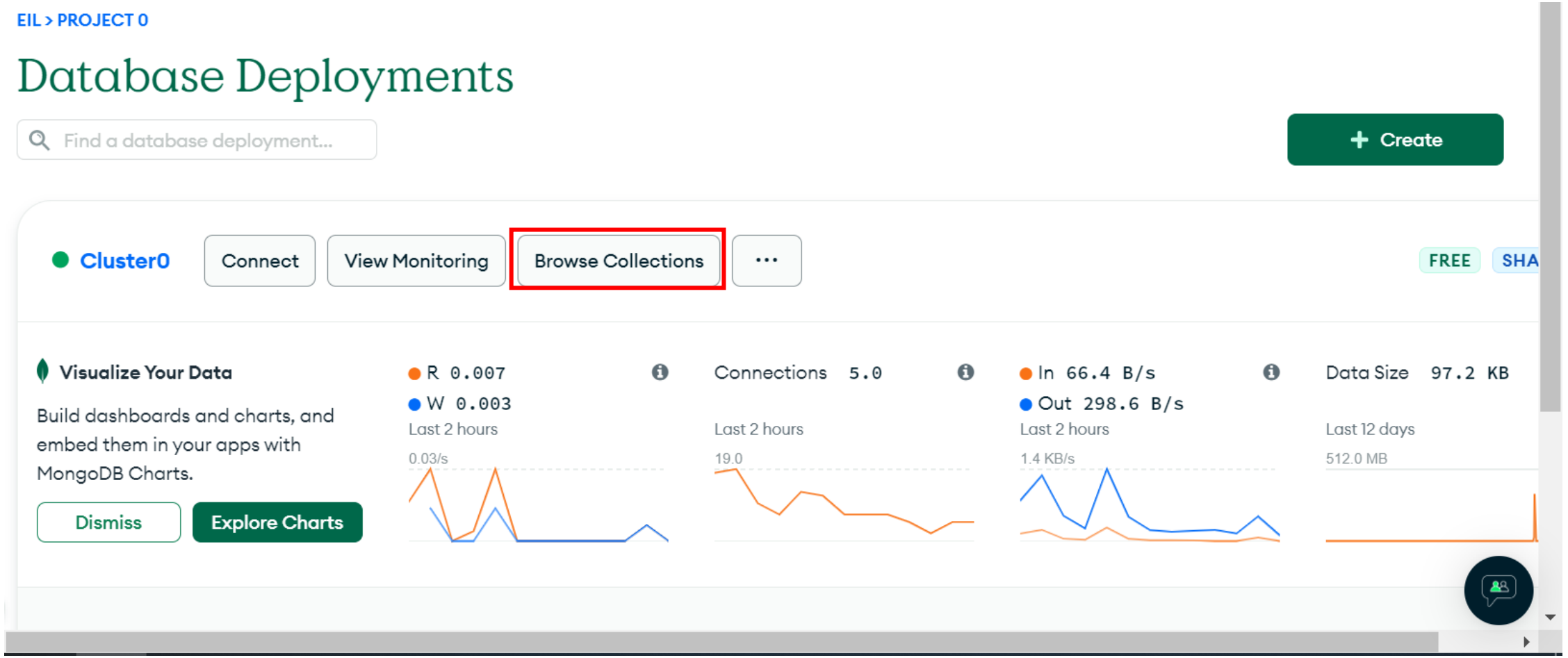
L'application peut alors être démarrée en exécutant la commande **`npm run start`**

Récupération du code pour débuter le TP

Dans le dossier backend, modifier les valeurs du fichier **.env** afin de renseigner les informations de connexion à votre base de données (reprendre les valeurs renseignées lors du TD2 sauf pour le DB_NAME qui sera à adapter en fonction du nom de votre nouvelle base de données) :

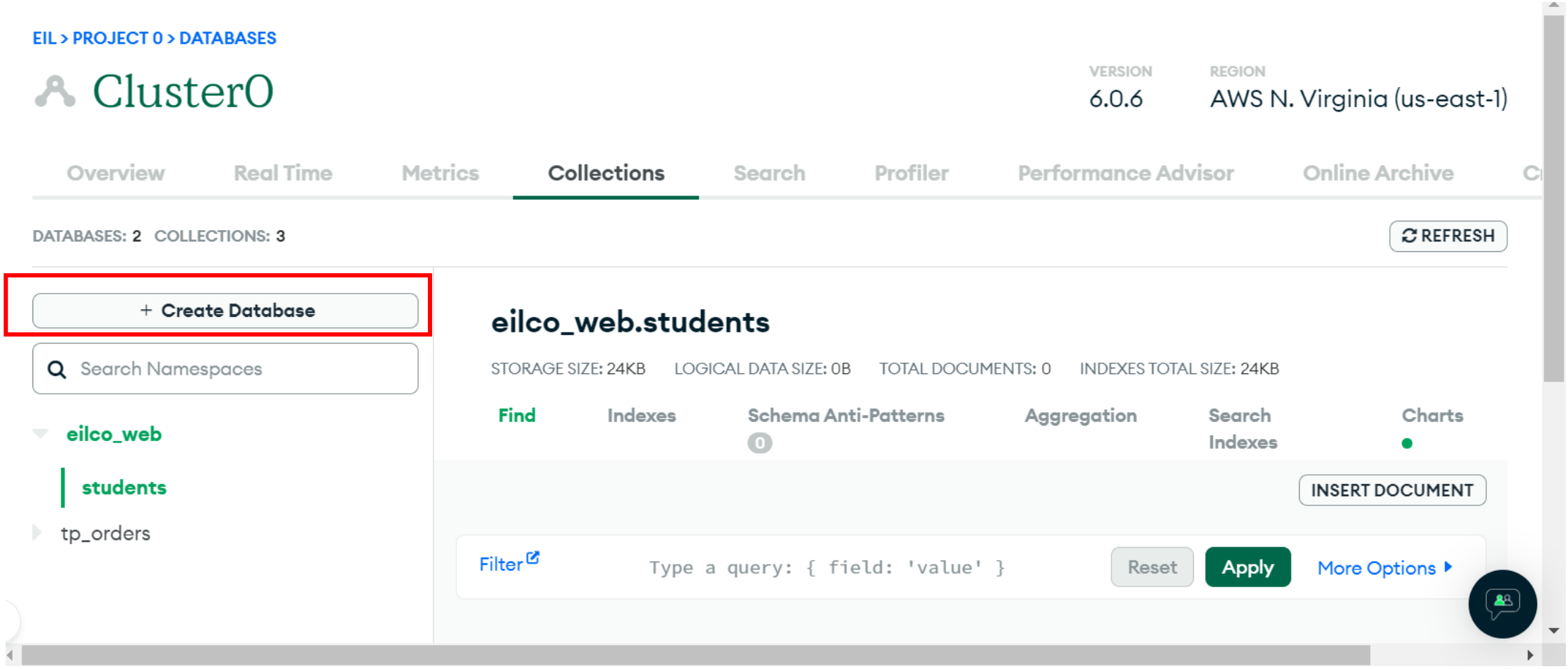
```
DB_USERNAME=your_username
DB_PASSWORD=your_password
DB_DOMAIN=your_domain.mongodb.net
DB_NAME=your_db_name
```

Sur Atlas, cliquer sur `Browse collections`

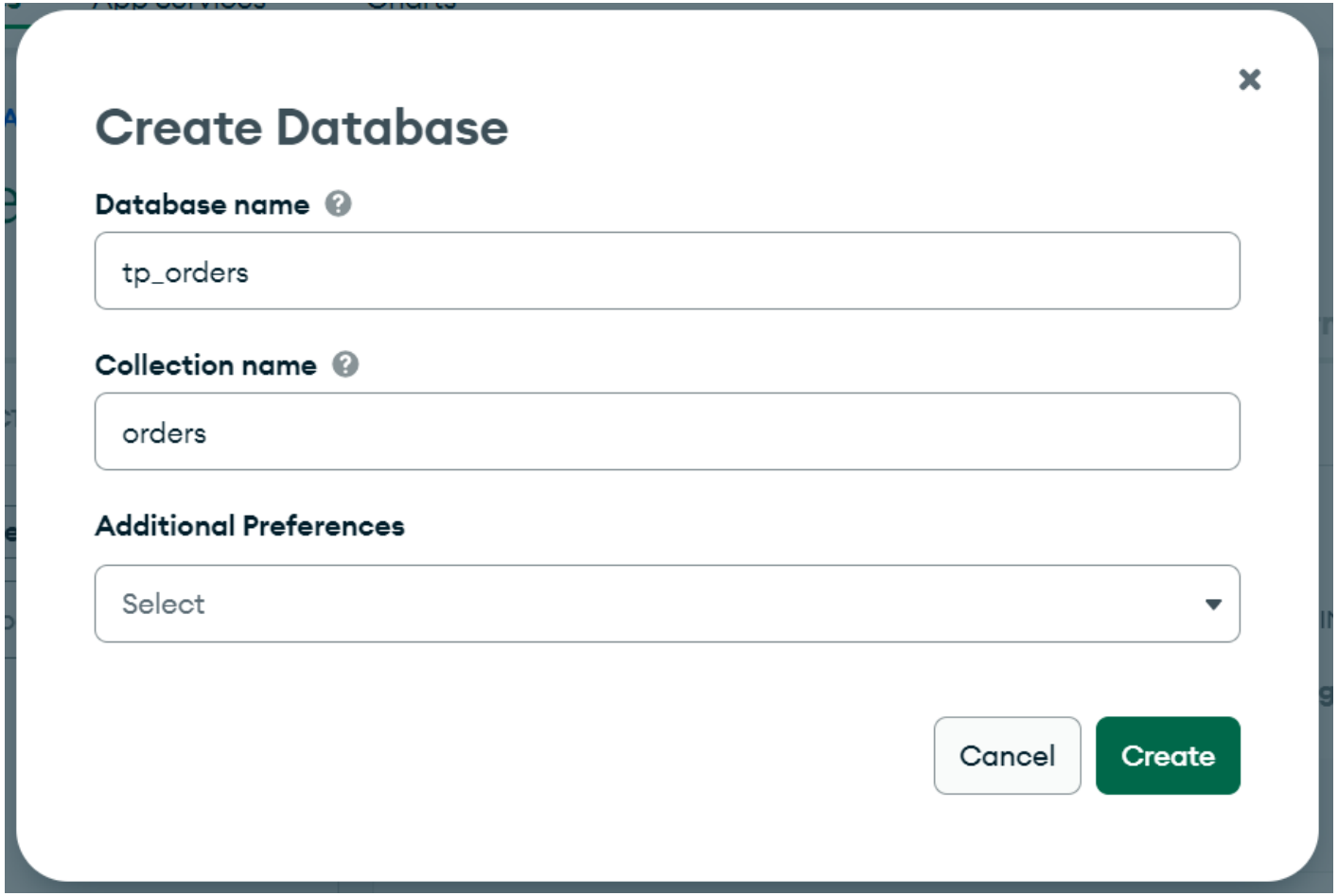


Création de la base de données et des collections

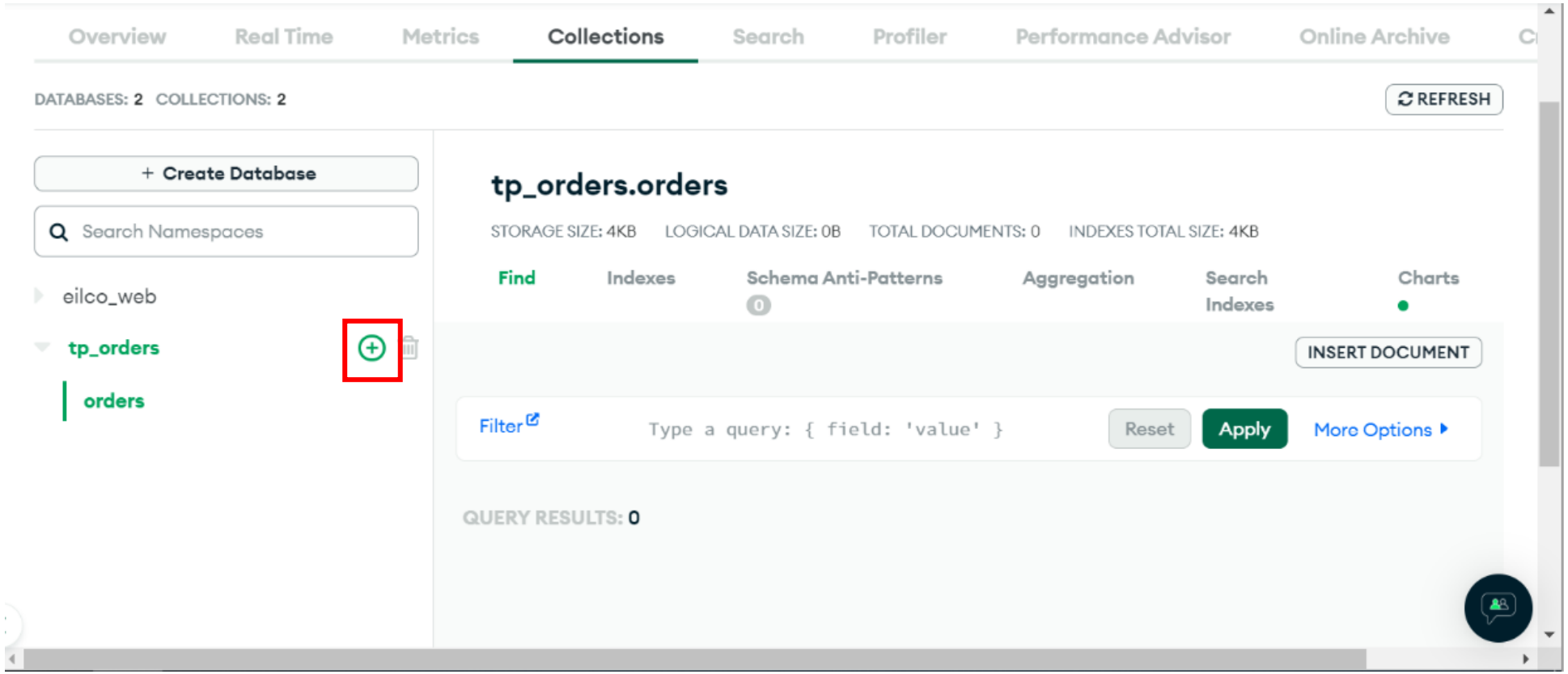
Cliquer sur `Create Database`



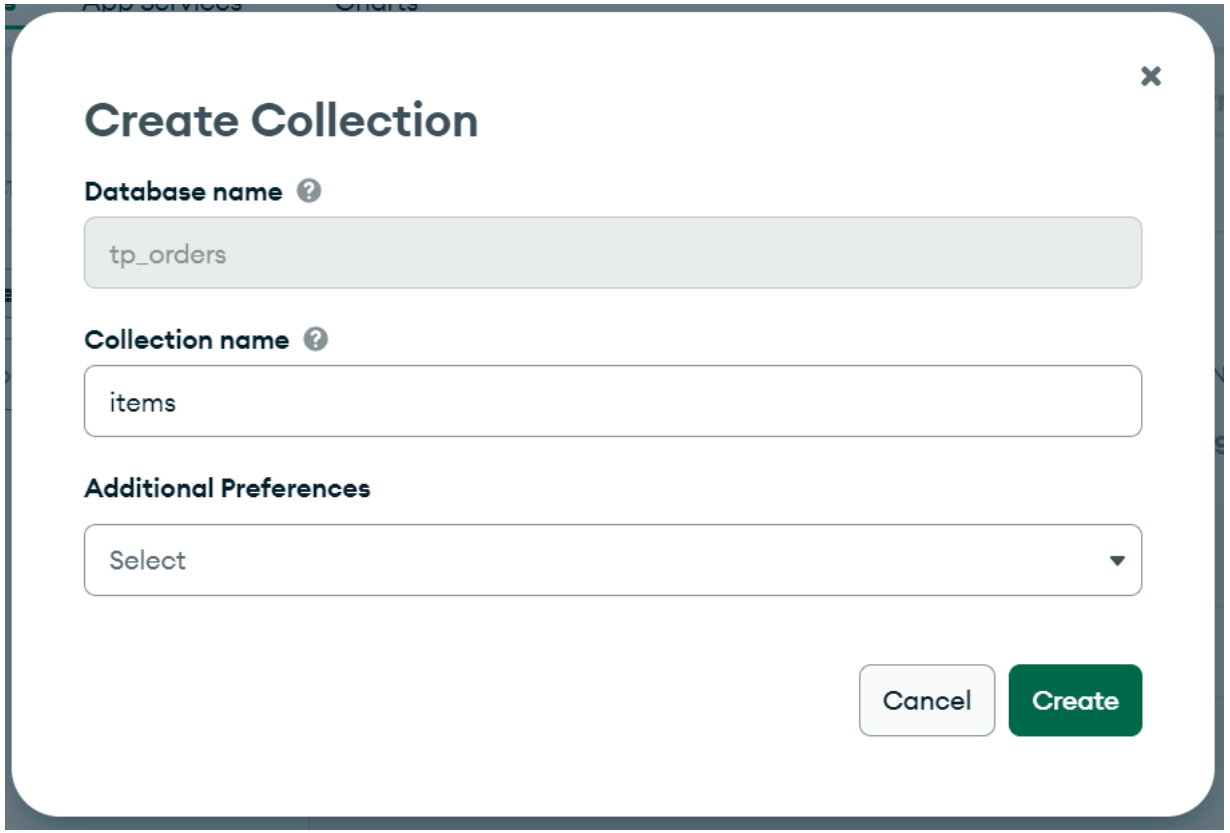
Renseigner le nom de la base de données `tp_orders` et la première collection `orders`



Cliquer sur le (+) à côté du nom de la base de données

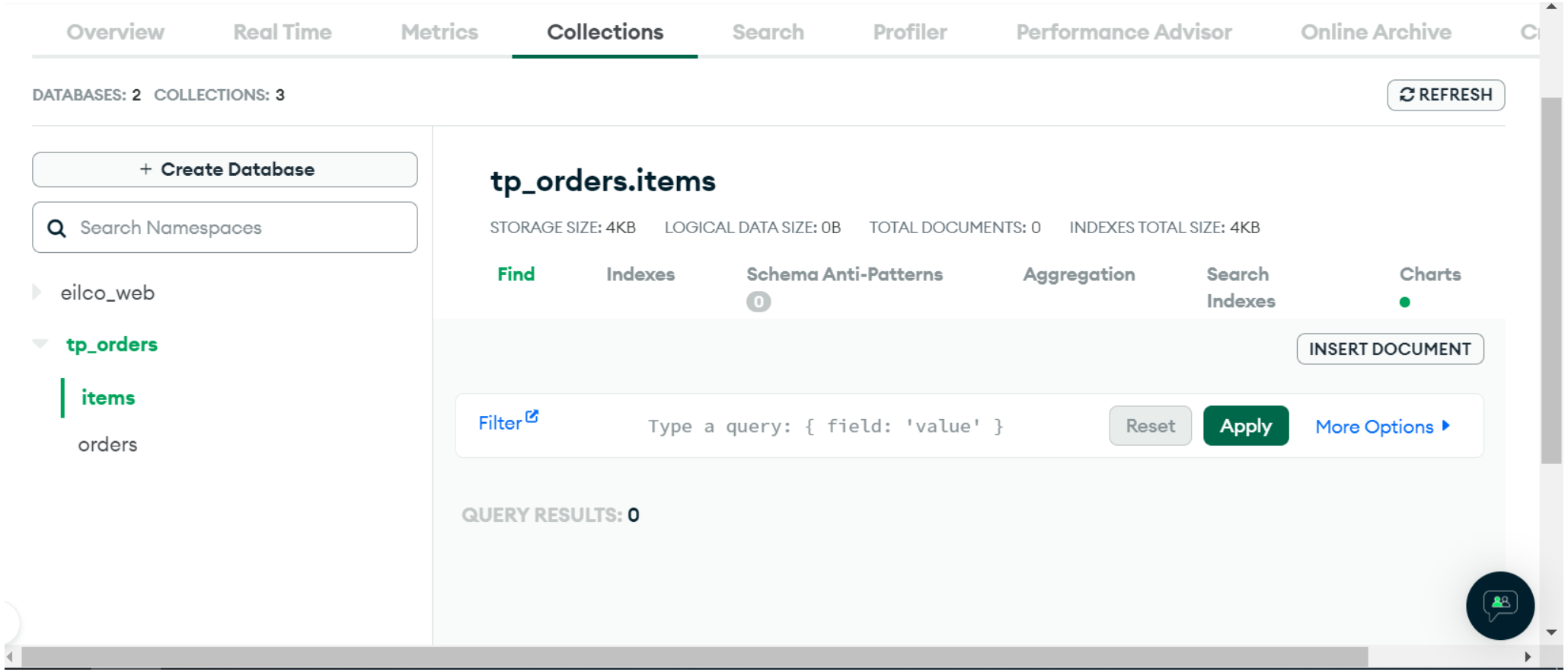


Renseigner le nom de la seconde collection `items`



Création de la base de données et des collections

Les deux collections sont maintenant créées



Insertion des données dans les collections de la base de données

Télécharger les jeux de données pour la base de données :

tp_orders.items.json et tp_orders.orders.json

Les importer dans la base de données