

# TP

## Création d'une application de gestion de commandes de pizzas



<https://www.jordansablon.fr/enseignement>



[https://github.com/jsablonEnseignement/tp\\_orders\\_sources](https://github.com/jsablonEnseignement/tp_orders_sources)

- Présentation du sujet
- FRONTEND - Suggestions de composants à créer
- BACKEND - Création des CRUD pour les commandes et articles
- BASE DE DONNEES – Présentation des collections
- MONGOOSE – Quelques indications pour le schéma de données à créer
- Récupération du code source du projet & installation des dépendances
- Récupération des collections pour Thunder Client
- Création de la base de données
- Insertion des données dans les collections de la base de données

# Présentation du sujet

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

[+ Créer une nouvelle commande](#)

Liste des commandes :

**Commande n°1 - John DOE** [Supprimer la commande](#)

Réalisée le 01/06/2023 14:00:00

**Statut : done**

**Bacon Groovy**    10.9€ **10.9€**

*Crème fraîche légère française, mozzarella, poulet rôti\*, oignons, bacon, sauce barbecue.*

**Prix total 10.9€**

[+ Ajouter un article](#)

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

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

The image shows a screenshot of a web application interface for managing orders. The interface is annotated with colored boxes and arrows pointing to suggested component names:

- A red box highlights the top navigation area, including a blue button labeled "+ Créer une nouvelle commande" and a dropdown menu labeled "Liste des commandes : Commande n°1". A red arrow points from this box to the text `<Orders />`.
- A green box highlights the main content area, which includes the order details: "Commande n°1 - John DOE", a red button "Supprimer la commande", the date "Réalisée le 01/06/2023 14:00:00", and the status "Statut : done". A green arrow points from this box to the text `<Order />`.
- A yellow box highlights a single order item: "Bacon Groovy" with quantity controls (minus, 1, plus), price "10.9€", and a description "Crème fraîche légère française, mozzarella, poulet rôti\*, oignons, bacon, sauce barbecue". A yellow arrow points from this box to the text `<Item />`.



# FRONTEND - Suggestions de composants à créer

\* Nom:

\* Prénom:

+ Ajouter un article

Annuler

→ `<NewOrder />`

## Orders

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

## Items

HTTP	URI	Description
GET	<code>/items</code>	Permet de récupérer l'ensemble des items
GET	<code>/items/{item_id}</code>	Permet de récupérer un item
POST	<code>/items</code>	Permet de créer un nouvel item
PUT	<code>/items/{item_id}</code>	Permet de mettre à jour un item
DELETE	<code>/items/{item_id}</code>	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
<code>_id</code>	ObjectId	Identifiant d'un item
<code>number</code>	Number	Numéro de la commande
<code>date</code>	Date	Date de création de la commande
<code>status</code>	String	Statut de la commande ( <i>A traiter, en cours, terminée</i> )
<code>client</code>	Object	Le client ayant passé la commande
<code>Items</code>	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	Double	Prix de l'article
quantity	Int32	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
<code>_id</code>	ObjectId	Identifiant d'un item
<code>name</code>	String	Nom de l'item
<code>description</code>	String	Description de l'item
<code>price</code>	Double	Prix de l'item



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,
```

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

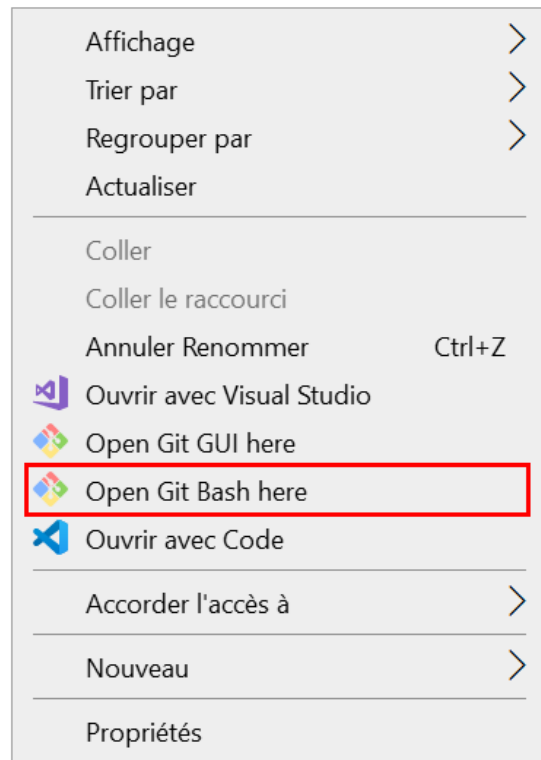
```
tableauDeString: {  
  type: Array<String>,  
}
```

OU

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

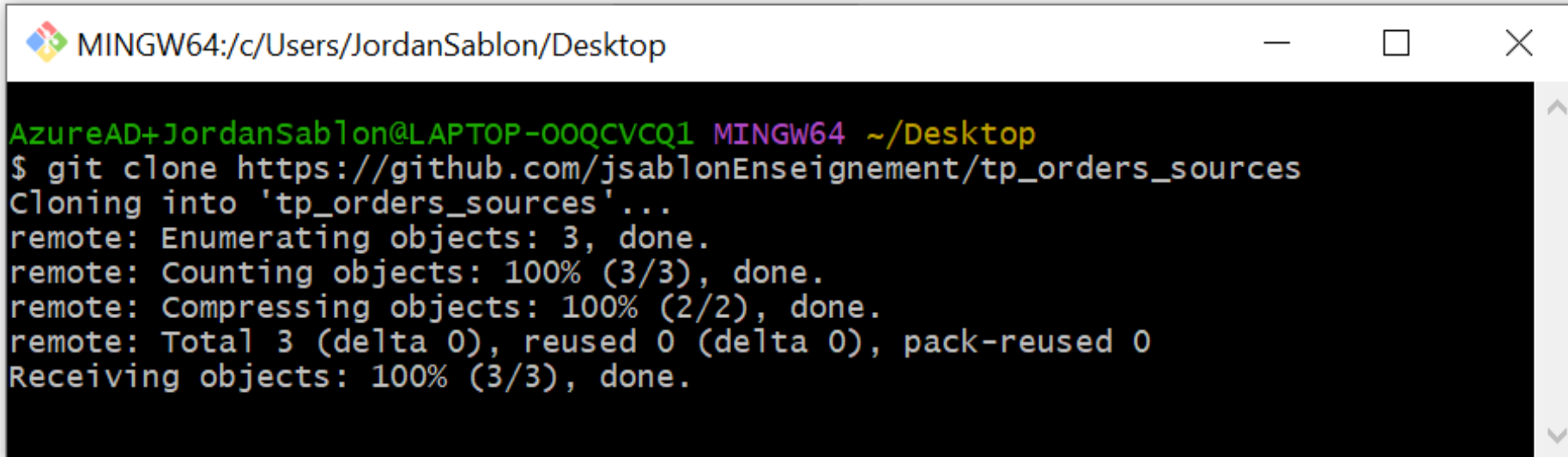
# Récupération du code source du projet & installation des dépendances

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 :



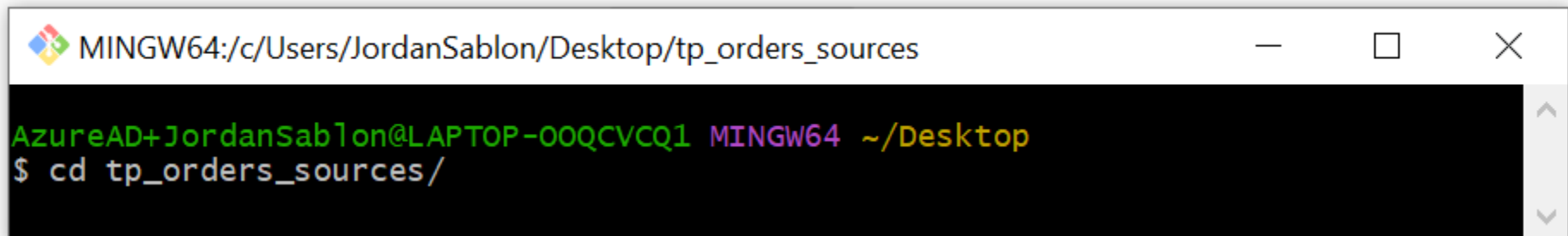
# Récupération du code source du projet & installation des dépendances

3. Cloner le répertoire Git : [https://github.com/jsablonEnseignement/tp\\_orders\\_sources](https://github.com/jsablonEnseignement/tp_orders_sources)



```
MINGW64:/c/Users/JordanSablon/Desktop  
AzureAD+JordanSablon@LAPTOP-00QCVCQ1 MINGW64 ~/Desktop  
$ git clone https://github.com/jsablonEnseignement/tp_orders_sources  
Cloning into 'tp_orders_sources'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.
```

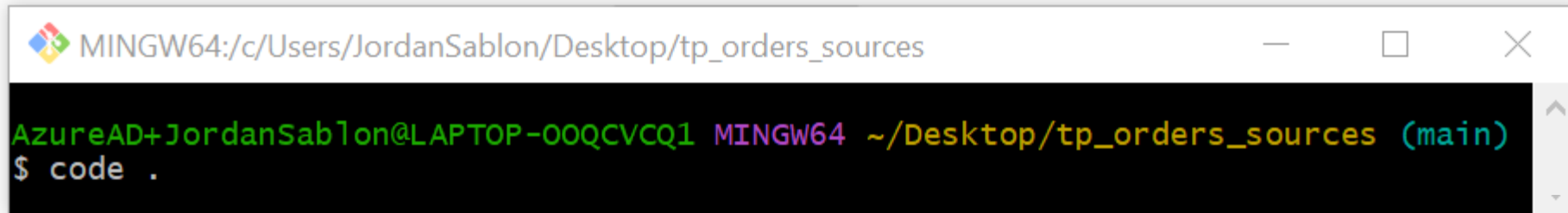
4. Se rendre dans le dossier précédemment cloné :



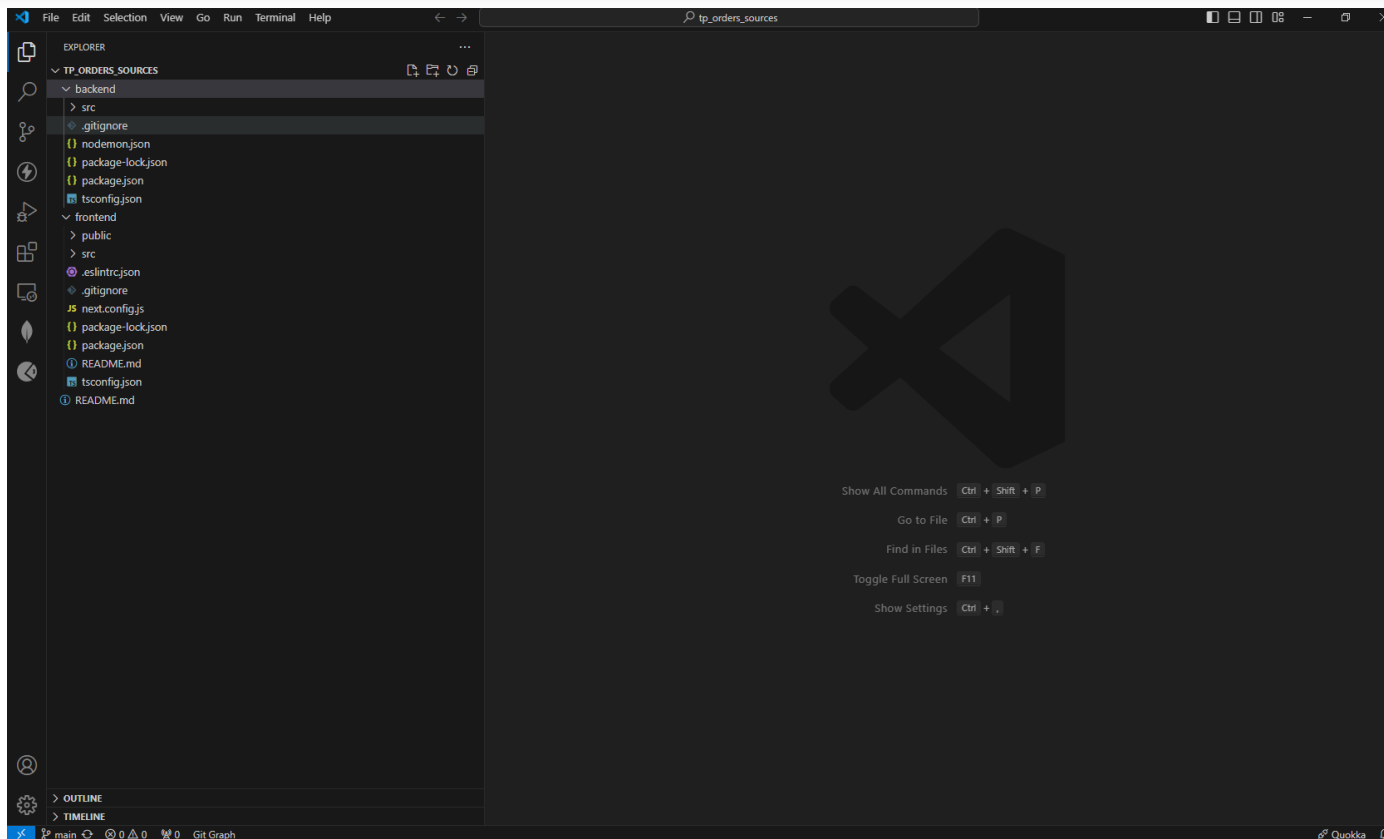
```
MINGW64:/c/Users/JordanSablon/Desktop/tp_orders_sources  
AzureAD+JordanSablon@LAPTOP-00QCVCQ1 MINGW64 ~/Desktop  
$ cd tp_orders_sources/
```

# Récupération du code source du projet & installation des dépendances

## 5. Ouvrir le projet dans VSCode :



```
MINGW64:/c/Users/JordanSablon/Desktop/tp_orders_sources  
AzureAD+JordanSablon@LAPTOP-00QVCVQ1 MINGW64 ~/Desktop/tp_orders_sources (main)  
$ code .
```



# 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 :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE powershell - backend + v [ ] [ ] ... v x
• PS C:\Users\JordanSablon\Desktop\tp_orders_sources\backend> npm i
npm WARN deprecated @types/mongoose@5.11.97: Mongoose publishes its own types, so you do not need to install this package.

added 149 packages, and audited 150 packages in 3s

12 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (5 moderate, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

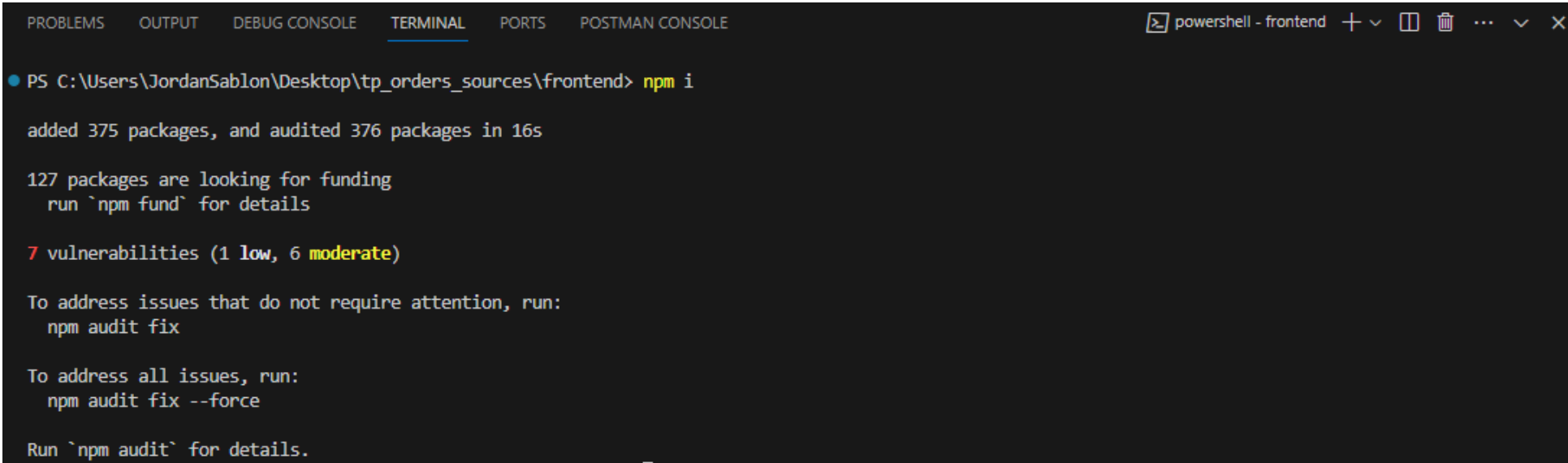
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

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

# 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 :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE powershell - frontend + - [ ] [ ] ... - X
● PS C:\Users\JordanSablon\Desktop\tp_orders_sources\frontend> npm i
added 375 packages, and audited 376 packages in 16s

127 packages are looking for funding
  run `npm fund` for details

7 vulnerabilities (1 low, 6 moderate)

To address issues that do not require attention, run:
  npm audit fix

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
```

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



# Récupération du code pour débiter le TP

Dans le fichier `index.ts`, mettre à jour l'URI de connexion à la base de données mongodb en renseignant vos informations de connexion :

- username
- password
- domain
- db\_name

```
mongoose.connect (  
  `mongodb+srv://<username>:<password>@<domain>.mongodb.net/<db_name>?retryWrites=true&w=majority`  
);
```

Télécharger les collections [orders.collection.json](#) et [items.collection.json](#)

Importer les collections dans Thunder Client

Sur Atlas, cliquer sur **Browse collections**

EIL > PROJECT 0

## Database Deployments

Find a database deployment...

+ Create

Cluster0

Connect

View Monitoring

**Browse Collections**

...

FREE

SHA

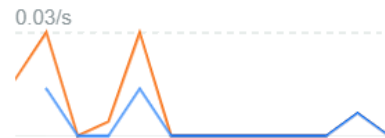
### Visualize Your Data

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

Dismiss

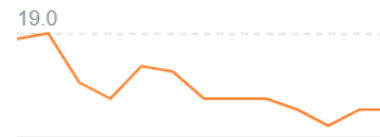
Explore Charts

R 0.007  
W 0.003  
Last 2 hours

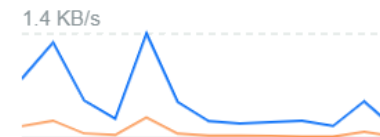


Connections 5.0

Last 2 hours



In 66.4 B/s  
Out 298.6 B/s  
Last 2 hours



Data Size 97.2 KB

Last 12 days

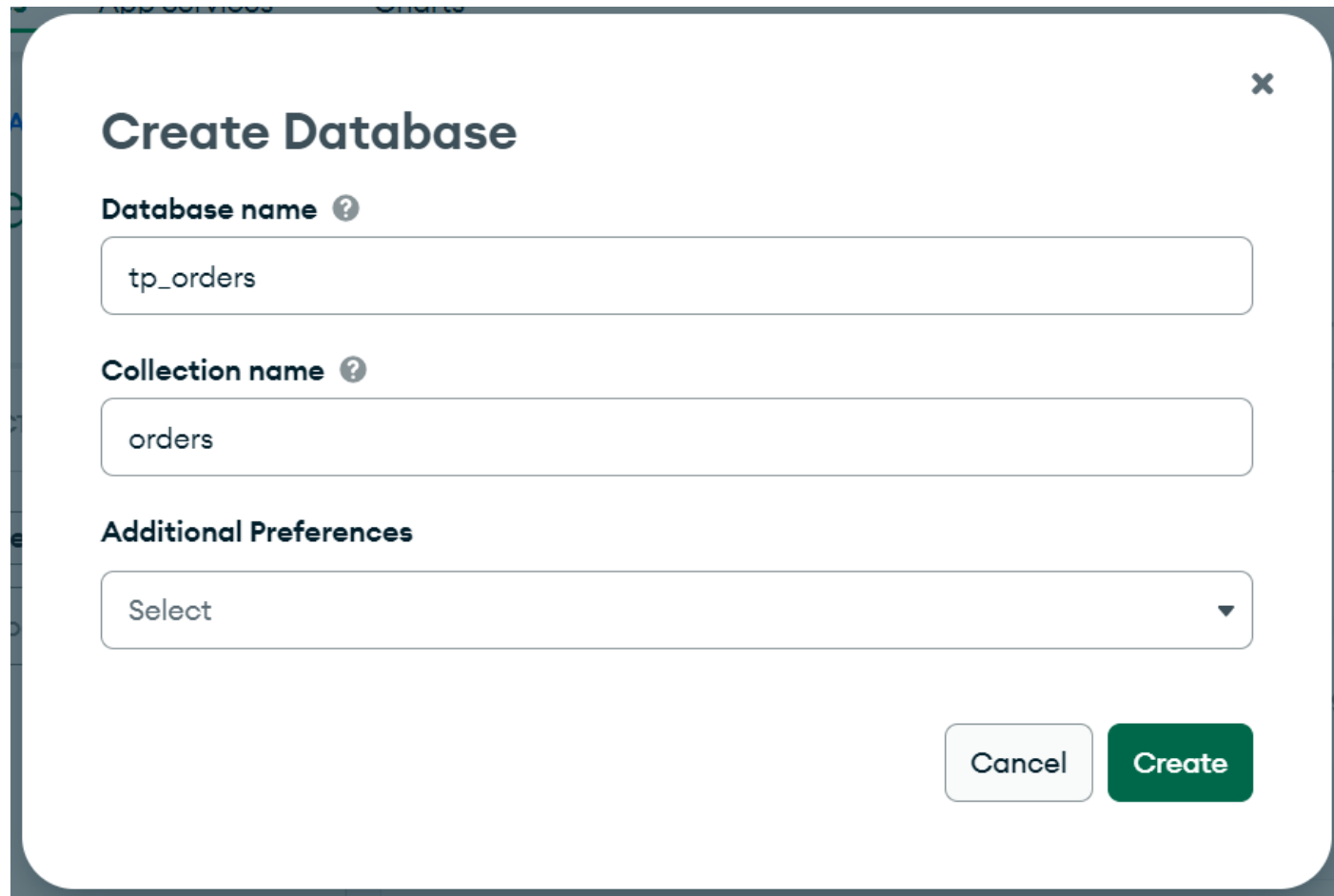
512.0 MB



## Cliquer sur Create Database

The screenshot displays the MongoDB Atlas interface. At the top, the breadcrumb navigation shows 'EIL > PROJECT 0 > DATABASES'. The 'Cluster0' logo is on the left, and the version '6.0.6' and region 'AWS N. Virginia (us-east-1)' are on the right. A navigation bar includes 'Overview', 'Real Time', 'Metrics', 'Collections' (highlighted), 'Search', 'Profiler', 'Performance Advisor', and 'Online Archive'. Below the navigation bar, it shows 'DATABASES: 2' and 'COLLECTIONS: 3' with a 'REFRESH' button. The left sidebar contains a '+ Create Database' button (highlighted with a red box), a 'Search Namespaces' input, and a tree view with 'eilco\_web' expanded to show 'students' and 'tp\_orders'. The main content area shows the 'eilco\_web.students' collection details: 'STORAGE SIZE: 24KB', 'LOGICAL DATA SIZE: 0B', 'TOTAL DOCUMENTS: 0', and 'INDEXES TOTAL SIZE: 24KB'. Below this are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', 'Search Indexes', and 'Charts'. An 'INSERT DOCUMENT' button is visible. At the bottom, there is a 'Filter' section with a query input field containing '{ field: 'value' }', 'Reset', 'Apply', and 'More Options' buttons.

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



**Create Database** ✕

**Database name** ?

**Collection name** ?

**Additional Preferences**

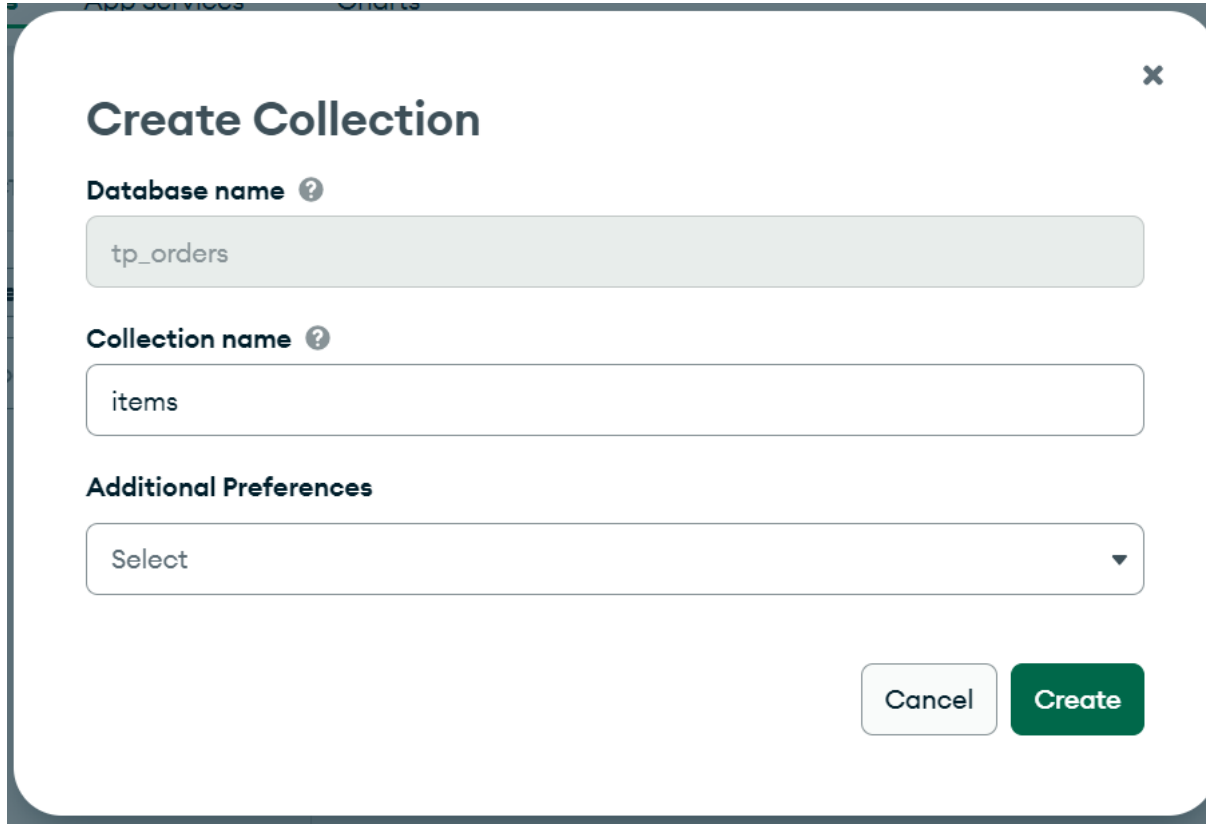
Cancel Create

# Création de la base de données et des collections

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

The screenshot displays the MongoDB Compass interface. At the top, there are navigation tabs: Overview, Real Time, Metrics, Collections (selected), Search, Profiler, Performance Advisor, and Online Archive. Below the tabs, it shows 'DATABASES: 2' and 'COLLECTIONS: 2' with a 'REFRESH' button. On the left sidebar, there is a '+ Create Database' button and a search bar for namespaces. The namespace tree shows 'eilco\_web' and 'tp\_orders' (expanded). Under 'tp\_orders', there is an 'orders' collection. A red box highlights a green '+' icon next to the 'tp\_orders' namespace. The main panel shows details for 'tp\_orders.orders', including storage size (4KB), logical data size (0B), total documents (0), and index size (4KB). There are tabs for Find, Indexes, Schema Anti-Patterns (with a '0' badge), Aggregation, Search Indexes, and Charts. An 'INSERT DOCUMENT' button is visible. A query filter bar contains the text 'Type a query: { field: 'value' }' with 'Reset', 'Apply', and 'More Options' buttons. At the bottom, it shows 'QUERY RESULTS: 0'.

Renseigner le nom de la seconde collection `items`



**Create Collection** ✕

**Database name** ?

tp\_orders

**Collection name** ?

items

**Additional Preferences**

Select ▼

Cancel Create

# Création de la base de données et des collections

Les deux collections sont maintenant créées

The screenshot displays the MongoDB Compass interface. At the top, there are navigation tabs: Overview, Real Time, Metrics, Collections (selected), Search, Profiler, Performance Advisor, Online Archive, and Clusters. Below the tabs, it shows 'DATABASES: 2' and 'COLLECTIONS: 3' with a 'REFRESH' button. On the left sidebar, there is a '+ Create Database' button and a 'Search Namespaces' search bar. The sidebar lists namespaces: 'eilco\_web', 'tp\_orders' (expanded), and 'orders'. Under 'tp\_orders', the 'items' collection is highlighted. The main area shows the details for 'tp\_orders.items': STORAGE SIZE: 4KB, LOGICAL DATA SIZE: 0B, TOTAL DOCUMENTS: 0, INDEXES TOTAL SIZE: 4KB. Below this are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', 'Search Indexes', and 'Charts'. An 'INSERT DOCUMENT' button is visible. A query filter bar contains the text 'Type a query: { field: 'value' }' with 'Reset', 'Apply', and 'More Options' buttons. At the bottom, it shows 'QUERY RESULTS: 0'. A chat icon is in the bottom right corner.



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