

TP

Création d'une application de gestion de commandes de pizzas



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



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
- 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 screenshot displays a web interface for managing orders. A red box highlights the entire page content. A blue button at the top left says '+ Créer une nouvelle commande'. Below it, the text 'Liste des commandes :' is followed by a dropdown menu showing 'Commande n°1'. A green box highlights a specific order card. The card title is 'Commande n°1 - John DOE' with a red button 'Supprimer la commande'. Below the title, it says 'Réalisée le 01/06/2023 14:00:00' and 'Statut : done'. A yellow box highlights an item in the order: 'Bacon Groovy' with a quantity of 1, a price of 10.9€, and a description 'Crème fraîche légère française, mozzarella, poulet rôti*, oignons, bacon, sauce barbecue.' At the bottom of the card, it shows 'Prix total 10.9€' and a blue button '+ Ajouter un article'. Three arrows point from the annotations to HTML tags: a red arrow from the top blue button to '<Orders />', a green arrow from the order card to '<Order />', and a yellow arrow from the item to '<Item />'.

+ Créer une nouvelle commande

Liste des commandes : Commande n°1

Commande n°1 - John DOE Supprimer la commande

Réalisée le 01/06/2023 14:00:00

Statut : done

Bacon Groovy - 1 + 10.9€ 10.9€ X
Crème fraîche légère française, mozzarella, poulet rôti*, oignons, bacon, sauce barbecue.

Prix total 10.9€

+ Ajouter un article

<Orders />

<Order />

<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')
```

BASE DE DONNEES – Présentation des collections

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	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
<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>	Number	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 :

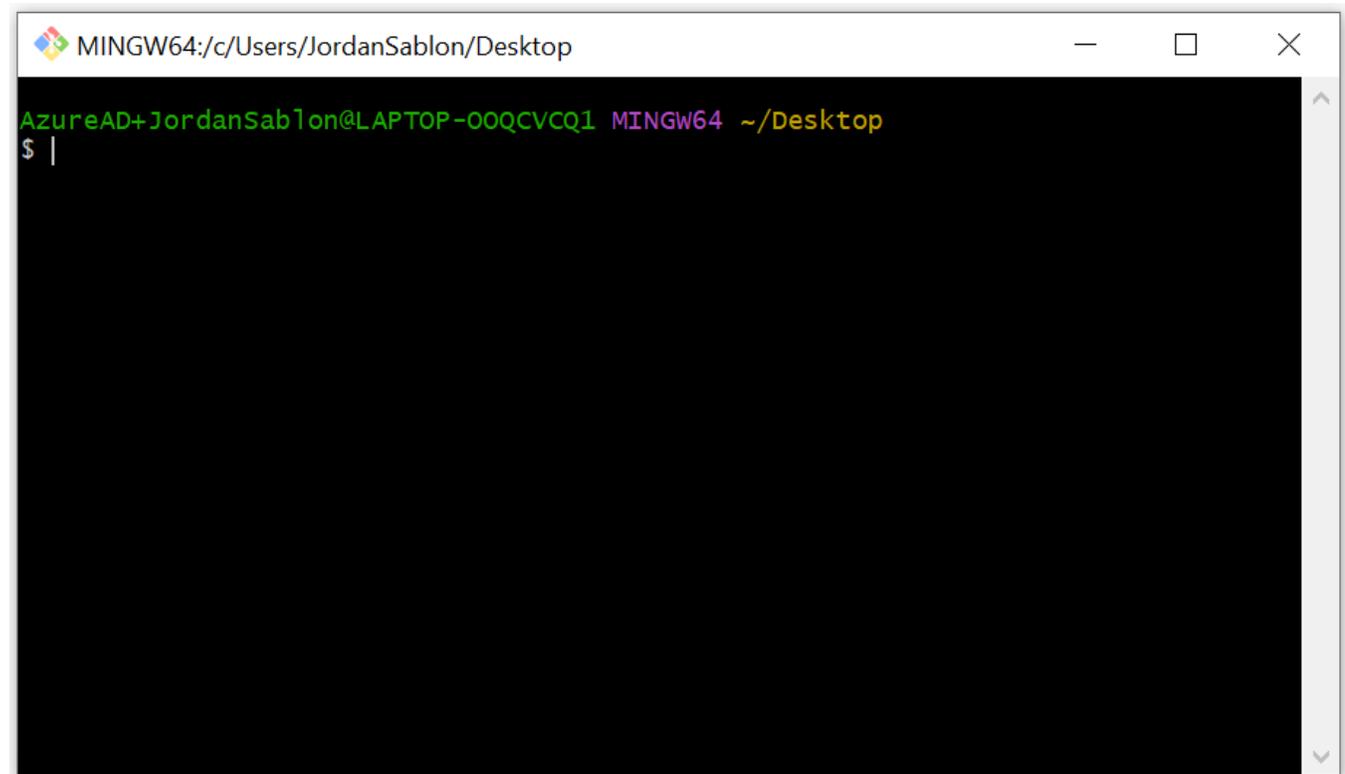
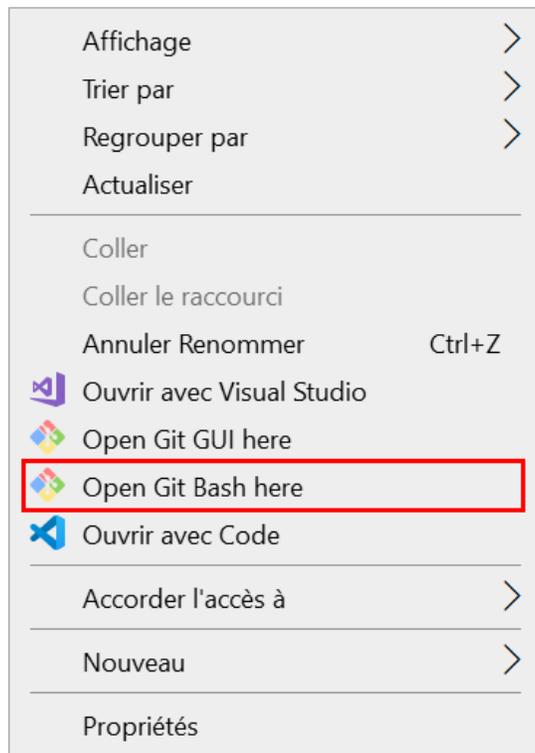
```
tableauDeString: {  
  type: Array<String>,  
},
```

OU

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

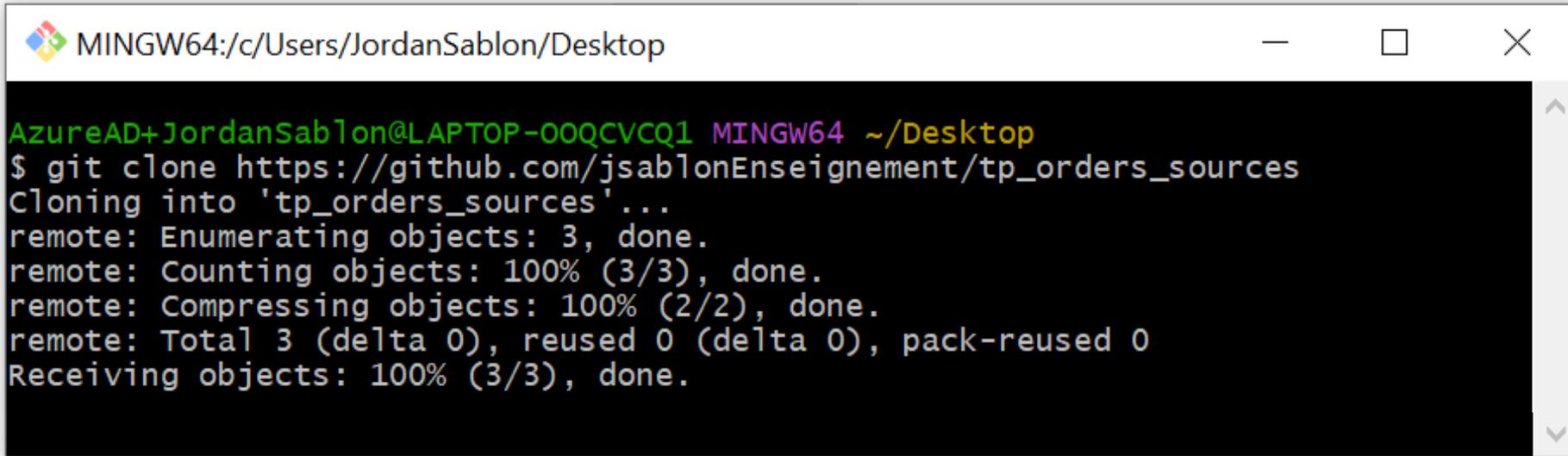
Récupération du code source du projet & installation des dépendances

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 :



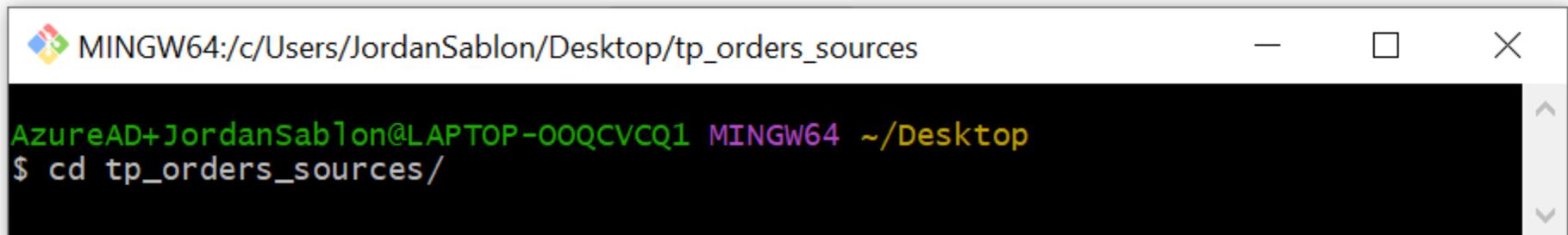
Récupération du code source du projet & installation des dépendances

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



```
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.
```

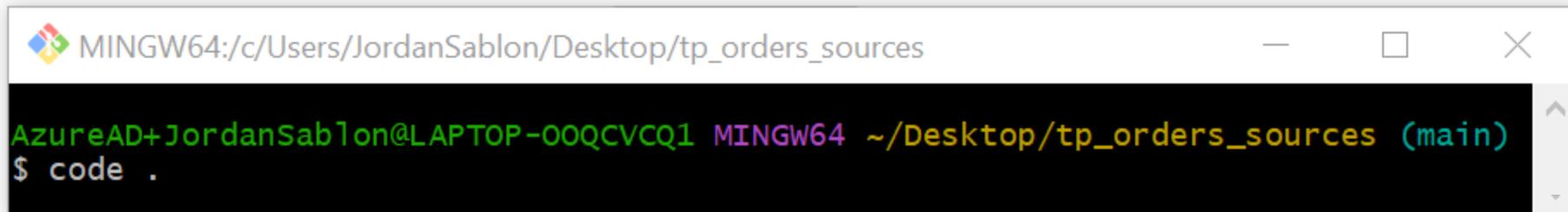
4. Se rendre dans le dossier précédemment cloné :



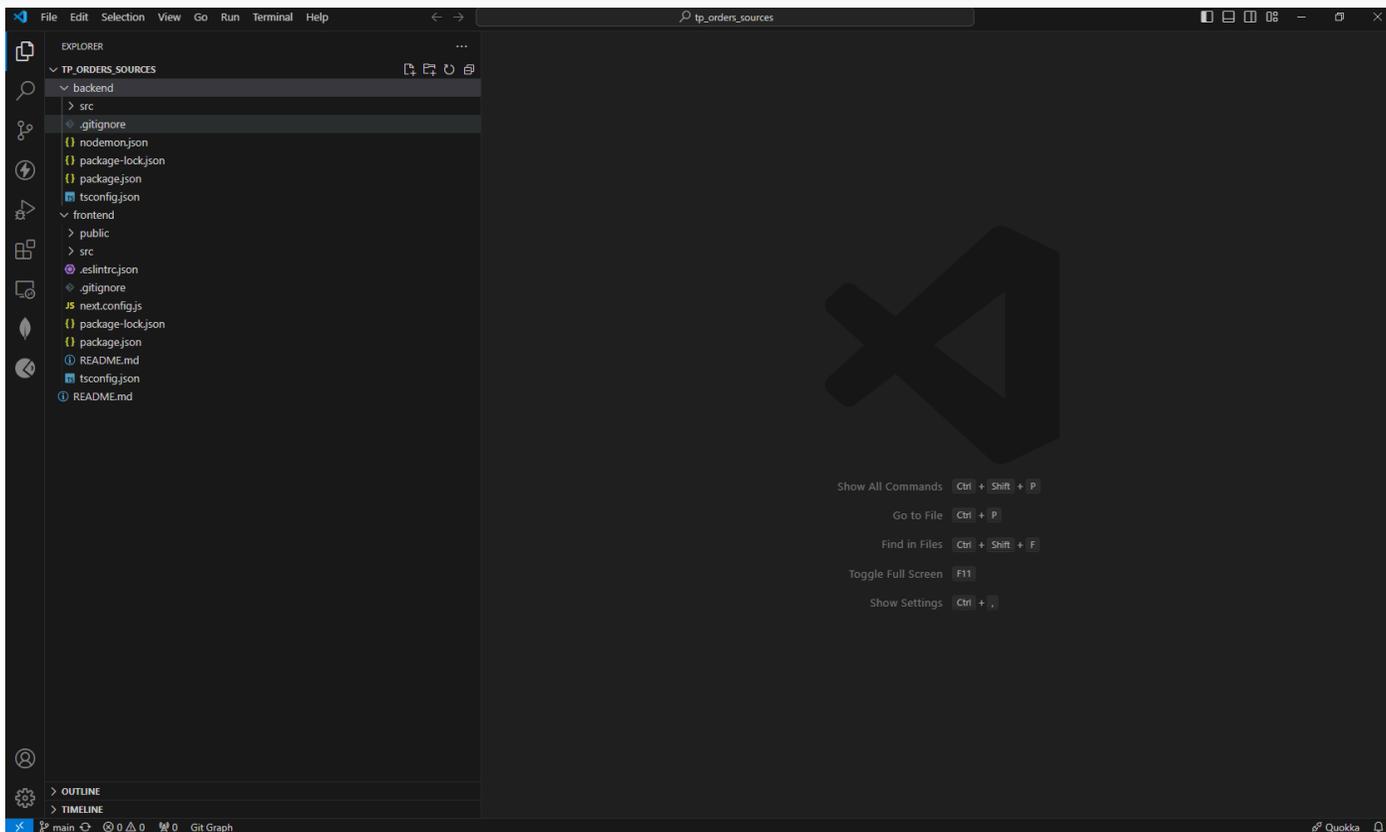
```
MINGW64:/c/Users/JordanSablon/Desktop/tp_orders_sources
AzureAD+JordanSablon@LAPTOP-00QCVCQ1 MINGW64 ~/Desktop
$ cd tp_orders_sources/
```

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

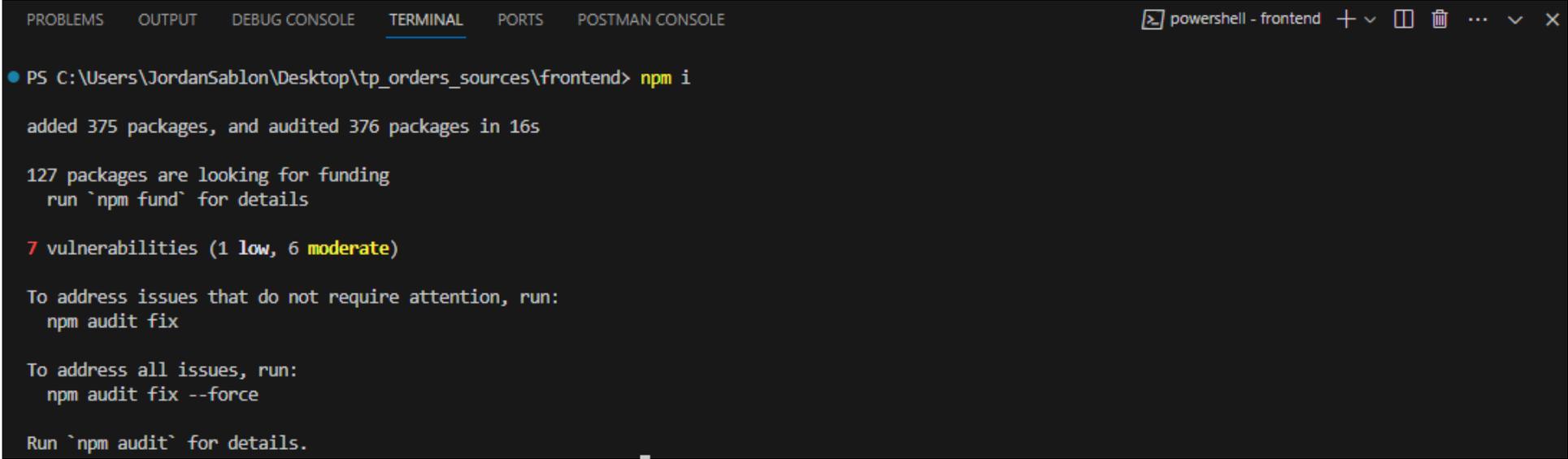
Run `npm audit` for details.
```

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

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 :



```
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.

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

Récupération du code pour débiter 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
```

Création de la base de données et des collections

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

0.03/s



i

Connections 5.0

Last 2 hours

19.0



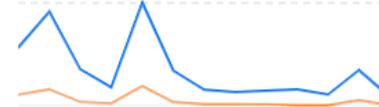
i

In 66.4 B/s

Out 298.6 B/s

Last 2 hours

1.4 KB/s



i

Data Size 97.2 KB

Last 12 days

512.0 MB



Cliquer sur Create Database

EIL > PROJECT 0 > DATABASES

Cluster0 VERSION 6.0.6 REGION AWS N. Virginia (us-east-1)

Overview Real Time Metrics **Collections** Search Profiler Performance Advisor Online Archive

DATABASES: 2 COLLECTIONS: 3 REFRESH

+ Create Database

Search Namespaces

- eilco_web
 - students**
 - tp_orders

eilco_web.students

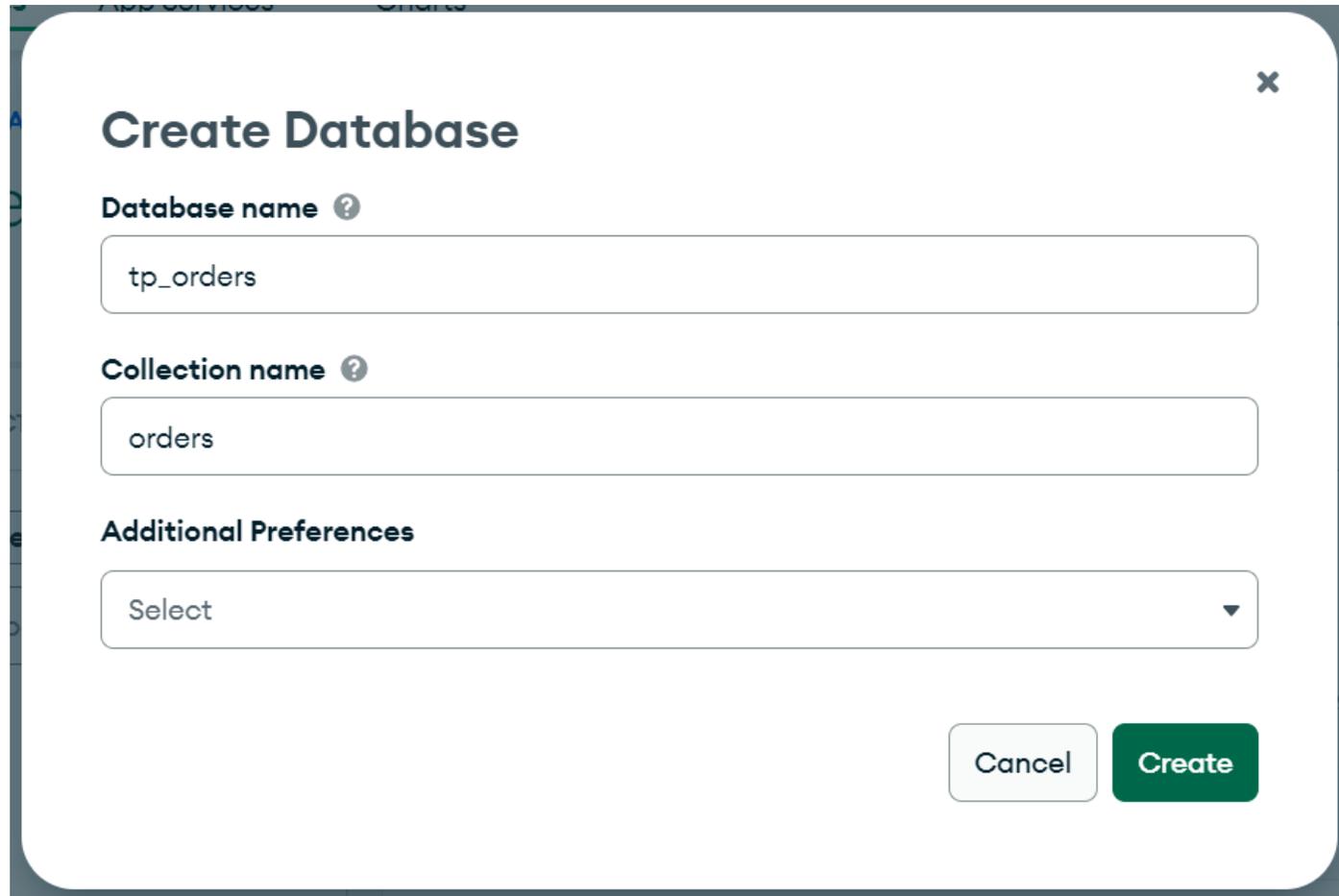
STORAGE SIZE: 24KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 24KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes Charts

INSERT DOCUMENT

Filter Reset Apply More Options

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



Create Database ✕

Database name ?

Collection name ?

Additional Preferences

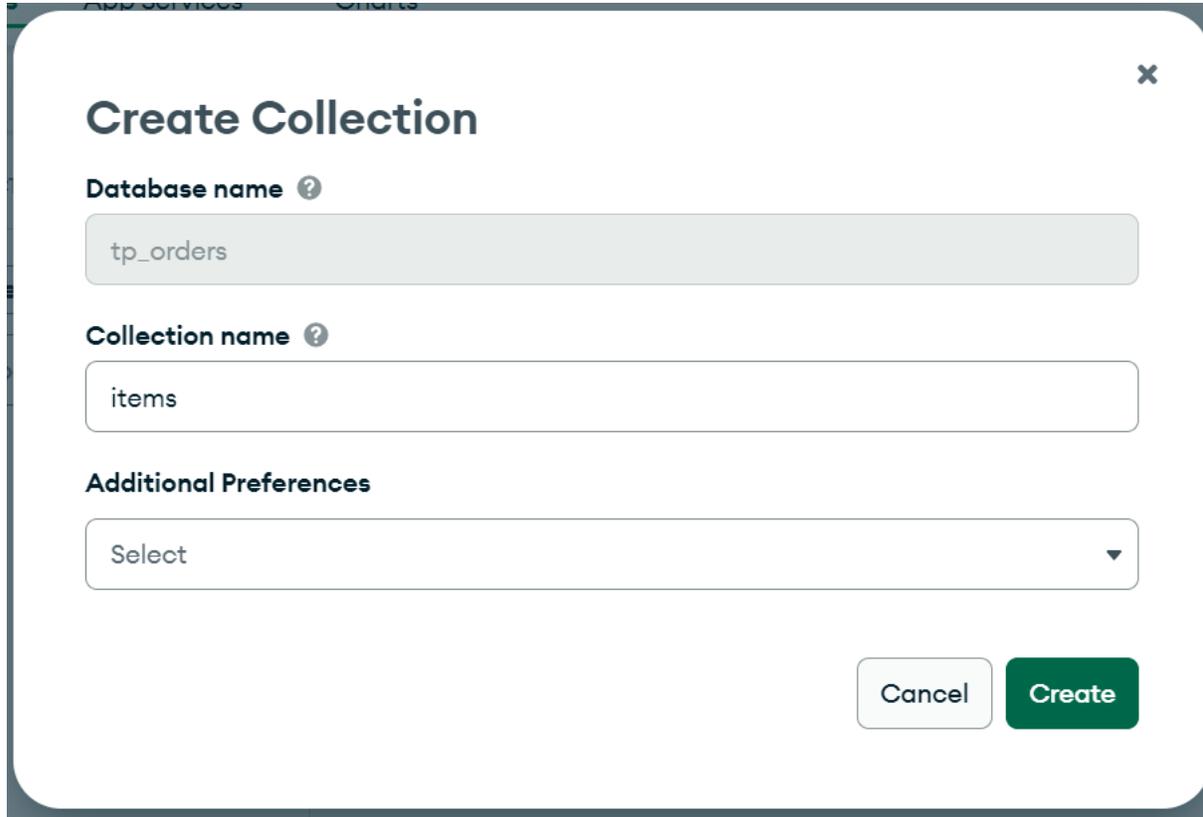
Cancel Create

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 sidebar lists two databases: 'eilco_web' and 'tp_orders'. The 'tp_orders' database is expanded, showing a collection named 'orders'. A red square highlights a green '+' icon next to the 'tp_orders' namespace. The main content area shows the details for the 'tp_orders.orders' collection, 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. Below this is a query filter section with a 'Filter' button, a text input containing '{ field: 'value' }', and buttons for 'Reset', 'Apply', and 'More Options'. 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 box for namespaces. The namespace tree shows 'eilco_web' expanded, and 'tp_orders' expanded to show 'items' and 'orders'. The main panel displays the 'tp_orders.items' collection with statistics: STORAGE SIZE: 4KB, LOGICAL DATA SIZE: 0B, TOTAL DOCUMENTS: 0, and INDEXES TOTAL SIZE: 4KB. Below the statistics are tabs for Find, Indexes, Schema Anti-Patterns (with a '0' indicator), Aggregation, Search Indexes, and Charts. An 'INSERT DOCUMENT' button is visible. A query filter box 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