CESI EATS

Documentation API

Thomas DESGRANGES
Frédéric NGUYEN
Yann SUBTS
Antoine SIRE

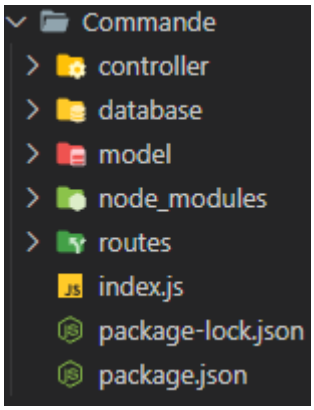
Table des matières

API Gateway

Pour récupérer les informations, c'est l'API Gateway qui va vous être utile.

Il est possible de contacter 5 microservices. 4 récupèrent leurs données de MongoDB, et la dernière depuis phpMyAdmin.

Hiérarchie



Tous les microservices utilisent *index.js* sauf l'API Gateway qui utilise *app.js*.

Article

Il est possible d'accéder aux données du microservice Article via : http://localhost:2000/article.

Les fonctions utilisables :

- GET http://localhost:2000/article : Récupère tous les articles.
- GET http://localhost:2000/article/:id : Récupère l'article en fonction de l'id.
- POST http://localhost:2000/article : Ajoute un nouvel article.
- PUT http://localhost:2000/article/:id : Met à jour un article en fonction de l'id.
- DELETE http://localhost:2000/article/:id : Supprime un article en fonction de l'id.

Un objet Article suit le schéma suivant :

```
idMenu : { type: String },
idRestaurant: { type: String, required: true },
name: { type: String, required: true },
content: { type: String, required: true },
tags: { type : Array, required: true },
imageUrl: { type: String, required: true },
isAvailable: { type: Boolean, required: true },
price: { type: Number, required:true }
```

Commande

Il est possible d'accéder aux données du microservice Commande via : http://localhost:2000/order.

Les fonctions utilisables :

- GET http://localhost:2000/order: Récupère toutes les commandes.
- GET http://localhost:2000/order/:id : Récupère la commande en fonction de l'id.

- POST http://localhost:2000/order: Ajoute une nouvelle commande.
- PUT http://localhost:2000/order/:id : Met à jour une commande en fonction de l'id.
- DELETE http://localhost:2000/order/:id : Supprime une commande en fonction de l'id.

Un objet Commande suit le schéma suivant :

```
idUser: { type: String, required: true },
idDeliver : {type: String },
idRestaurant: { type: String, required: true },
shippingAddress: {
  address: { type: String, required: true },
  city: { type: String, required: true },
  postalCode: { type: String, required: true },
  country: { type: String, required: true },
},
articles: [
  {
    name: { type: String, required: true},
    amount: { type: Number, required: true},
    content: { type: String, required: true},
    imageURL: { type: String, required: true},
    price: { type: Number, required: true},
    product: { type: mongoose.Schema.Types.ObjectId, required: true, ref:
'Product'}
  }
],
status: { type: String, required: true, default: 'Pending' },
dateOrder: { type: Date, required: true },
shippingPrice: { type: Number, required: true },
totalPrice: { type: Number, required: true }
```

Menu

Il est possible d'accéder aux données du microservice Menu via : http://localhost:2000/menu.

Les fonctions utilisables :

- GET http://localhost:2000/menu : Récupère tous les menus.
- GET http://localhost:2000/menu/:id : Récupère le menu en fonction de l'id.
- POST http://localhost:2000/menu : Ajoute un nouveau menu.
- PUT http://localhost:2000/menu/:id : Met à jour un menu en fonction de l'id.
- DELETE http://localhost:2000/menu/:id : Supprime un menu en fonction de l'id.

Un objet Menu suit le schéma suivant :

```
idRestaurant: { type: String, required: true},
name: { type: String, required: true },
content: { type: String, required: true},
articles: [
  {
```

```
    name: { type: String, required: true},
    content: { type: String, required: true},
    imageURL: { type: String, required: true},
    product: { type: mongoose.Schema.Types.ObjectId, required: true, ref:
'Product'}
  }
],
imageUrl: { type: String, required: true},
price: { type: Number, required: true }
```

Restaurant

Il est possible d'accéder aux données du microservice Restaurant via : http://localhost:2000/restaurant.

Les fonctions utilisables :

- GET http://localhost:2000/restaurant : Récupère tous les restaurants.
- GET http://localhost:2000/restaurant/:id : Récupère le restaurant en fonction de l'id.
- POST http://localhost:2000/restaurant : Ajoute un nouveau restaurant.
- PUT http://localhost:2000/restaurant/:id : Met à jour un restaurant en fonction de l'id.
- DELETE http://localhost:2000/restaurant/:id : Supprime un restaurant en fonction de l'id.

Un objet Restaurant suit le schéma suivant :

```
idOwner: {type: String, required : true},
name: { type: String, required: true },
content: { type: String, required: true },
addressRestaurant: {
    address: { type: String, required: true },
    city: { type: String, required: true },
    postalCode: { type: String, required: true },
    country: { type: String },
},
imageURL: { type: String},
openingDays: [
  {
    day: { type: String},
    isOpen: { type: Boolean},
    openHour: { type: String },
    closeHour: { type: String },
  }
]
```

User

Il est possible d'accéder aux données du microservice User via : http://localhost:2000/user.

Les fonctions utilisables :

- GET http://localhost:2000/user : Récupère tous les utilisateurs.
- GET http://localhost:2000/user/:id : Récupère l'utilisateur en fonction de l'id.
- POST http://localhost:2000/user : Ajoute un utilisateur commande.
- PUT http://localhost:2000/user/:id : Met à jour un utilisateur en fonction de l'id.
- DELETE http://localhost:2000/user/:id : Supprime un utilisateur en fonction de l'id.

L'objet Utilisateur n'a pas de schéma comme les autres, effectivement celui-ci est orienté phpMyAdmin.

Voici les tables :

infos	roles	user_roles	users
id name surname userId createdAt updatedAt	id name createdAt updatedAt	roleId userId createdAt updatedAt	id email phone password createdAt updatedAt

Création d'utilisateur

Pour cette partie, il n'y a pas de microservice, c'est directement dans l'API Gateway.

La création se fait à l'aide de : http://localhost:2000/api/auth/signin.

Les informations à fournir dans le JSON :

```
email
phone
password
name
surname
roles: [« »]
```

Connexion

Pour cette partie, il n'y a pas de microservice, c'est directement dans l'API Gateway.

La création se fait à l'aide de : http://localhost:2000/api/auth/signup.

Les informations à fournir dans le JSON :

```
email
password
```