



GUIDE UTILISATEUR

Projet pizza

Résumé

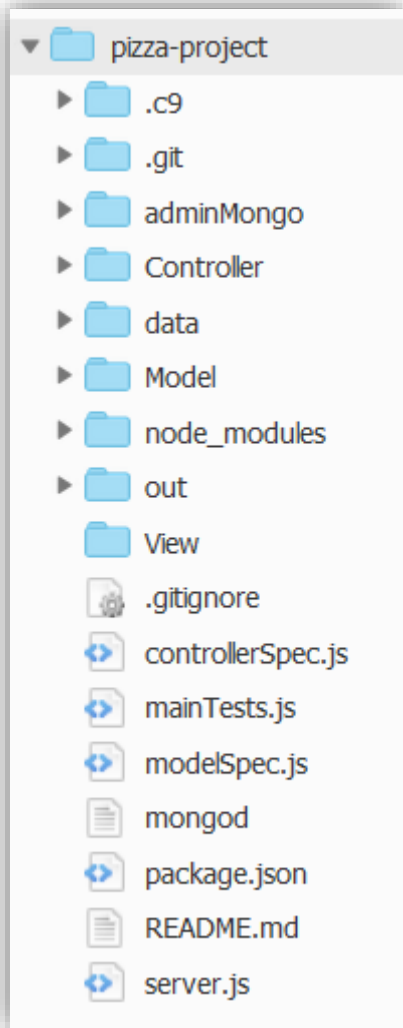
Guide permettant de comprendre l'architecture et le fonctionnement de l'API du site pizza

Surya Elroh

Table des matières

Architecture	2
Routes	3
Pizza	3
Ingrédient	6
Socket.....	10

Architecture



Le site est architecturé de manière à respecté le model MVC (model-view-controller).

Dans le dossier **Model** nous trouvons les représentations de nos objets pour la base de données. C'est dans ces fichier que l'on définit les attributs que contiennent les objets ainsi que leurs propriétés. La définition de schéma permet de contrôler la forme des données inséré en base de données.

Dans le dossier **Controller** nous trouvons les différentes méthodes permettant de communiquer avec la base de données. On y trouve des opérations CRUD (create read update delete) pour chacun des model définis.

Dans ce projet le dossier **View** est vide car il ne s'agit pas d'un site mais juste d'une API.

Si ce n'étais pas le cas nous y trouverions les différentes pages du site.

Le dossier out contient toute la documentation technique du projet. Celle-ci ressense tous les commentaires écrits permettant d'expliquer les différentes méthodes et imports.

A la racine du projet nous pouvons remarquer les fichiers : **mainTest.js**, **controllerSpec.js** et **modelSpec.js**. MainTest.js va être le point d'entrée des tests unitaires de l'API. On trouve à l'intérieur des imports des deux autres fichiers cités précédemment. Ils font eux-mêmes le lien avec les différents fichiers de tests qui se trouvent dans les dossiers : Model et Controller. Le fichier se défini en point d'entré dans le fichier package.json qui permet de paramétrer le projet.

C'est aussi à la racine que se trouve un fichier très important du site, **server.js**. Il permet d'inclure tous les modules nécessaires au bon fonctionnement du l'API. Il défini l'accès à notre base de données mais aussi à nos différents modèles et contrôleurs.

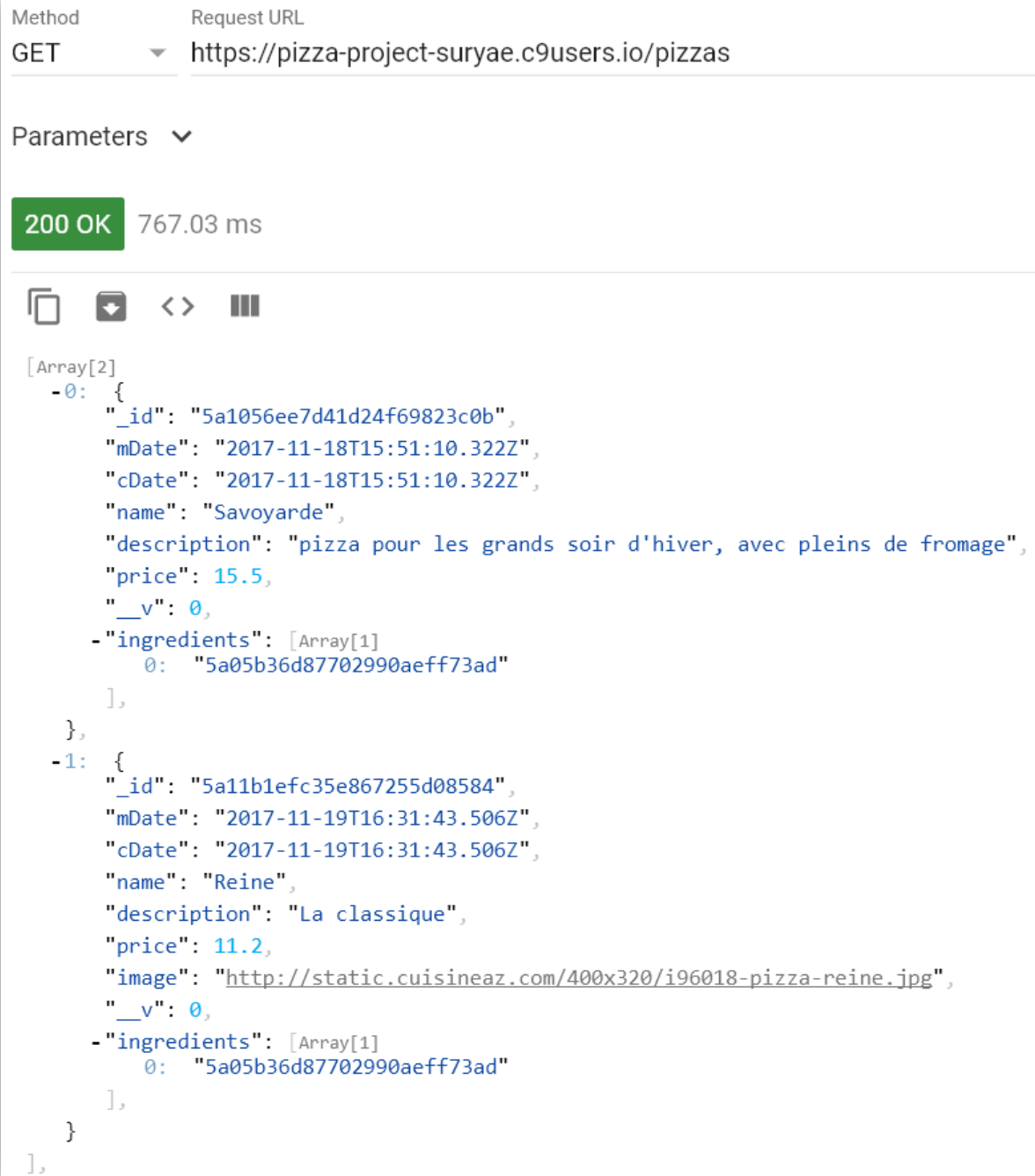
Routes

Les routes permettent de définir quelles sont les actions à effectuer en fonction de l'URL et des paramètres reçus ainsi que les méthodes appelées.

Afin de les tester nous utilisons l'outil Advanced REST client.

Pizza

`Get('.../pizzas/')` permet de récupérer toutes les pizzas stockées en base. La requête ne prend aucun paramètre.



Method: GET Request URL: <https://pizza-project-suryae.c9users.io/pizzas>

Parameters: ▾

200 OK 767.03 ms

[Array[2]]

```
-0: {
  "_id": "5a1056ee7d41d24f69823c0b",
  "mDate": "2017-11-18T15:51:10.322Z",
  "cDate": "2017-11-18T15:51:10.322Z",
  "name": "Savoyarde",
  "description": "pizza pour les grands soir d'hiver, avec pleins de fromage",
  "price": 15.5,
  "__v": 0,
  "ingredients": [Array[1]]
    0: "5a05b36d87702990aef73ad"
  ],
},
-1: {
  "_id": "5a11b1efc35e867255d08584",
  "mDate": "2017-11-19T16:31:43.506Z",
  "cDate": "2017-11-19T16:31:43.506Z",
  "name": "Reine",
  "description": "La classique",
  "price": 11.2,
  "image": "http://static.cuisineaz.com/400x320/i96018-pizza-reine.jpg",
  "__v": 0,
  "ingredients": [Array[1]]
    0: "5a05b36d87702990aef73ad"
  ],
},
]
```

[Get\('.../pizzas/:_id'\)](#) permet de récupérer les informations d'une pizza grâce à son id placé en paramètre.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- Request URL:** <https://pizza-project-suryae.c9users.io/pizzas/5a1056ee7d41d24f69823c0b>
- Parameters:** (collapsed)
- Status:** 200 OK
- Response Time:** 622.20 ms
- Response Body (JSON):**

```
{
  "_id": "5a1056ee7d41d24f69823c0b",
  "mDate": "2017-11-18T15:51:10.322Z",
  "cDate": "2017-11-18T15:51:10.322Z",
  "name": "Savoyarde",
  "description": "pizza pour les grands soir d'hiver, avec pleins de fromage",
  "price": 15.5,
  "__v": 0,
  "ingredients": [
    "5a05b36d87702990aeff73ad"
  ]
}
```

[Post\('.../pizzas/'\)](#) permet d'insérer une nouvelle pizza en base. La nouvelle pizza sera placée dans les paramètres afin d'être reçue par la base de données. La requête nous retourne ensuite l'élément qui vient d'être créé.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- Request URL:** <https://pizza-project-suryae.c9users.io/pizzas/>
- Parameters:** (collapsed)
- Body content type:** application/json
- Editor view:** Text input
- Response Body (JSON):**

```
{
  "name": "Végétarienne",
  "description": "L'incontournable du moment",
  "price": 13.5,
  "image": "https://menu-vegetarien.com/wp-content/uploads/2015/09/pizza-vegetarienne.jpg",
  "ingredients": [
    "5a05b36d87702990aeff73ad",
    "5a0ec3e71f554ee9e12925e4"
  ]
}
```

```
200 OK 776.82 ms

{
  "__v": 0,
  "mDate": "2017-11-19T18:43:17.836Z",
  "cDate": "2017-11-19T18:43:17.836Z",
  "name": "Végétarienne",
  "description": "L'incontournable du moment",
  "price": 13.5,
  "image": "https://menu-vegetarien.com/wp-content/uploads/2015/09/pizza-vegetarienne.jpg",
  "_id": "5a11d0c5a94f0889cb115001",
  "ingredients": [Array[2]]
    0: "5a05b36d87702990aeff73ad",
    1: "5a0ec3e71f554ee9e12925e4"
  ],
}
```

Put('.../pizzas/ :_id') permet de modifier les informations d'une pizza.

```
Method      Request URL
PUT         https://pizza-project-suryae.c9users.io/pizzas

Parameters ^

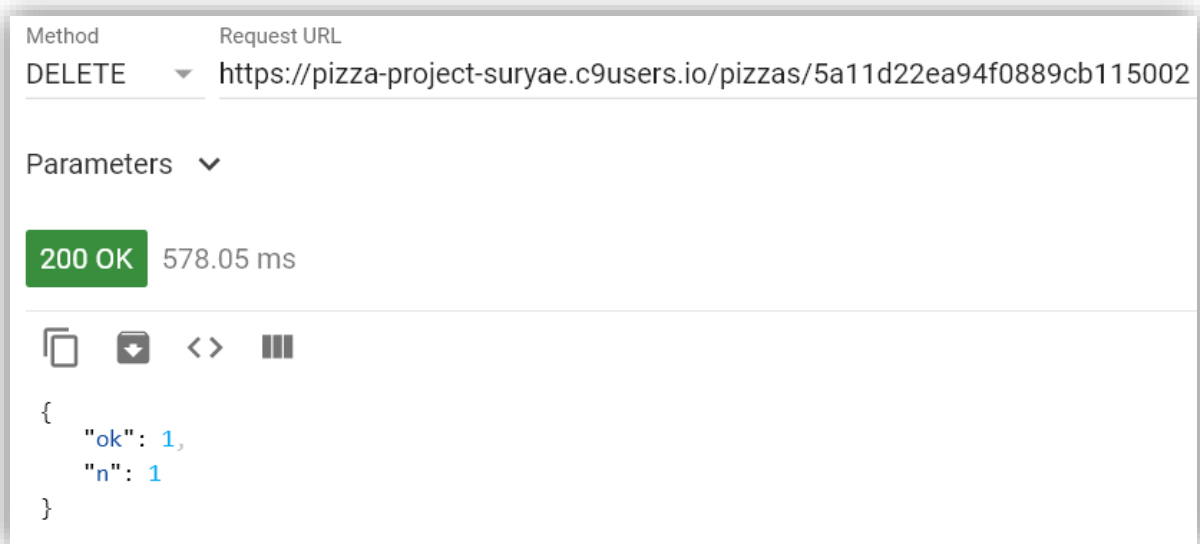
Headers      Body
-----
Body content type  Editor view
application/json   Text input

{
  "_id": "5a11d0c5a94f0889cb115001",
  "name": "Végétarienne",
  "description": "L'incontournable du moment. Légère elle ravira tous les mordus de légumes.",
  "price": 10
}
```

```
200 OK 649.52 ms

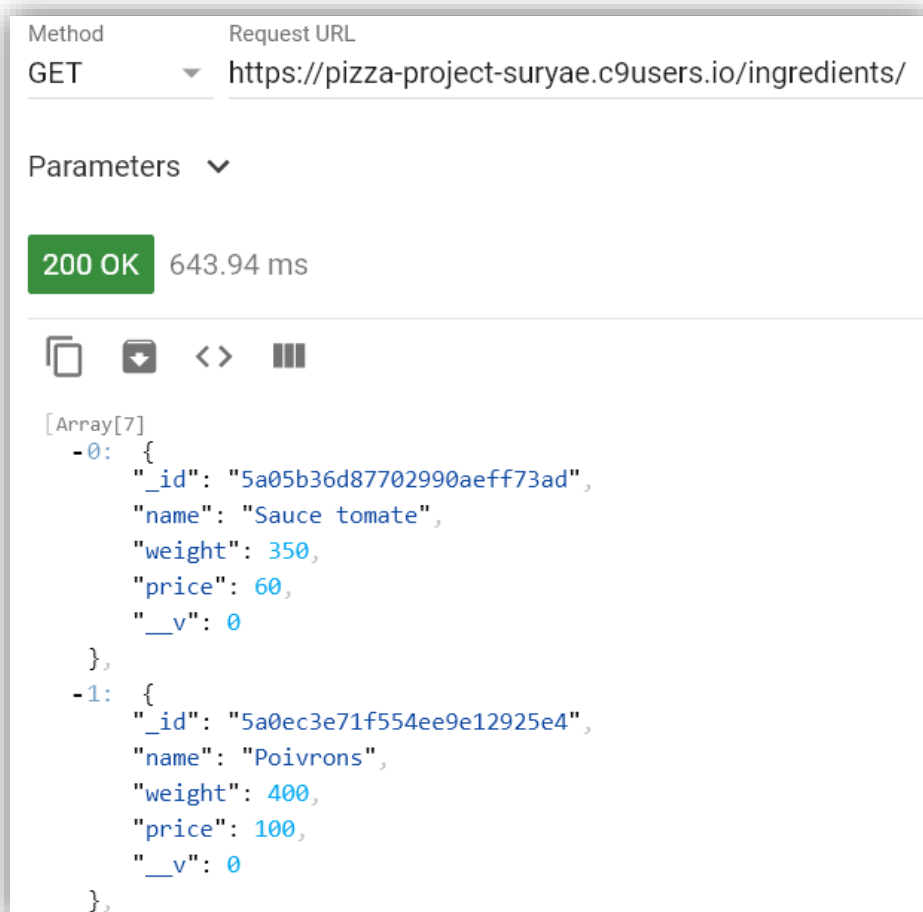
{
  "ok": 0,
  "n": 0,
  "nModified": 0
}
```

[Delete\('.../pizzas/ :_id'\)](#) permet de supprimer une pizza grâce à son id.



Ingrédient

[Get\('.../ingredients/'\)](#) permet de récupérer tous les ingrédients stockés en base. Le requête ne prend aucun paramètre.



[Get\('.../ingredients/ :_id'\)](#) permet de récupérer les informations d'un ingrédient grâce à son id placé en paramètre.

Method: GET Request URL: <https://pizza-project-suryae.c9users.io/ingredients/5a05b36d87702990aeff73ad>

Parameters: ▾

200 OK 618.20 ms

```
{
  "_id": "5a05b36d87702990aeff73ad",
  "name": "Sauce tomate",
  "weight": 350,
  "price": 60,
  "__v": 0
}
```

[Get\('.../ingredients/name/ :name'\)](#) permet de récupérer les informations d'un ingrédient grâce à son nom placé en paramètre.

Method: GET Request URL: <https://pizza-project-suryae.c9users.io/ingredients/name/Poivrons>

Parameters: ▾

200 OK 616.53 ms

```
{
  "_id": "5a0ec3e71f554ee9e12925e4",
  "name": "Poivrons",
  "weight": 400,
  "price": 100,
  "__v": 0
}
```


[Post\('.../ingredients/'\)](#) permet d'insérer un nouvel ingrédient en base. Celui-ci sera placé dans les paramètres afin d'être reçu par la base de données. La requête nous retourne ensuite l'élément qui vient d'être créé.





Method	Request URL
POST	https://pizza-project-suryae.c9users.io/ingredients/

Parameters ^

Headers	Body
Body content type application/json	Editor view Text input

```
{  
  "name": "Olives",  
  "weight": 300,  
  "price": 0.6  
}
```

200 OK 621.93 ms

```
{  
  "__v": 0,  
  "name": "Olives",  
  "weight": 300,  
  "price": 0.6,  
  "_id": "5a11d425a94f0889cb115003"  
}
```

Put('.../ingredients') permet de modifier les informations d'un ingrédient.

Method	Request URL
PUT	https://pizza-project-suryae.c9users.io/ingredients/
Parameters ^	
Headers	
Body	
Body content type	Editor view
application/json	Text input
<pre>{ "_id": "5a11d425a94f0889cb115003", "weight": 150 }</pre>	

200 OK	647.81 ms
<pre>{ "ok": 1, "nModified": 1, "n": 1 }</pre>	

Delete('.../ingredients/ :_id') permet de supprimer un ingrédient grâce à son id.

Method	Request URL
DELETE	https://pizza-project-suryae.c9users.io/ingredients/5a11d425a94f0889cb115003
Parameters v	
200 OK 626.24 ms	
<pre>{ "ok": 1, "n": 1 }</pre>	

Socket

Afin de permettre aux sites utilisant l'API d'être au courant en temps réel des changements apporté à la base de données des sockets ont été mises en place dans les contrôleurs.

Pour cela nous paramétrons tout d'abord socket.io pour qu'il puisse écouter et savoir lorsque des utilisateurs sont connectés à lui.

```
/**
 * -----
 * Socket.io
 * -----
 */

/**
 * Require and set socket.io
 */
const io = require('socket.io').listen(server);

/**
 * listen to connection
 */
io.on('connection', function(socket){
  console.log('user connected');
  socket.on('disconnect', function(){
    console.log('user disconnected');
  });
});

/**
 * Set io global in order to access it in all the project
 */
global.io = io;
```

Ensuite dans chaque méthode CRUD créée les sockets sont présents afin d'envoyer les informations à tous les utilisateurs étant connectés.

```
/**
 * Post a new ingredient
 * And emit it to informe there is a new one
 */
router.post('/', (req, res, next) => {
  let ingredientS = new ingredientSchema(req.body);
  return ingredientS.save()
    .then((ingredient)=>{
      global.io.emit('new ingredient toast', ingredient);
      global.io.emit('new ingredient available', ingredient);
      return res.json(ingredient);
    })
    .catch((err)=>{
      res.send(err);
      console.log(err);
    });
});
```

Par exemple ici à chaque fois que nous créons un nouvel ingrédients le site reçoit une évènement 'new ingredient toast' et 'new ingredient available'.

Le premier permet de notifier à n'importe quel utilisateur sur le site qu'un nouvel ingrédient vient d'être rajouté.

The screenshot shows a web application interface for managing ingredients. At the top, there is a dark header with the text "Les Pizzas" and "Crée ta Pizza" on the left, and "Gérer les ingrédients" on the right. Below the header, there is a form with three input fields: "Name :", "Weight:", and "Price : (en centimes)". The "Name" field contains the text "Nouvel Ingrédient". The "Weight" field contains the number "653". The "Price" field contains the number "65". A blue "Save" button is located at the bottom of the form. A white toast notification is displayed in the center of the screen, containing the text "Un nouvel ingredient !!!! woaaaaa" and an "OK" button.

Le deuxième permet de notifier aux personnes consultant la liste des ingrédients qu'un nouveau vient d'être ajouté.