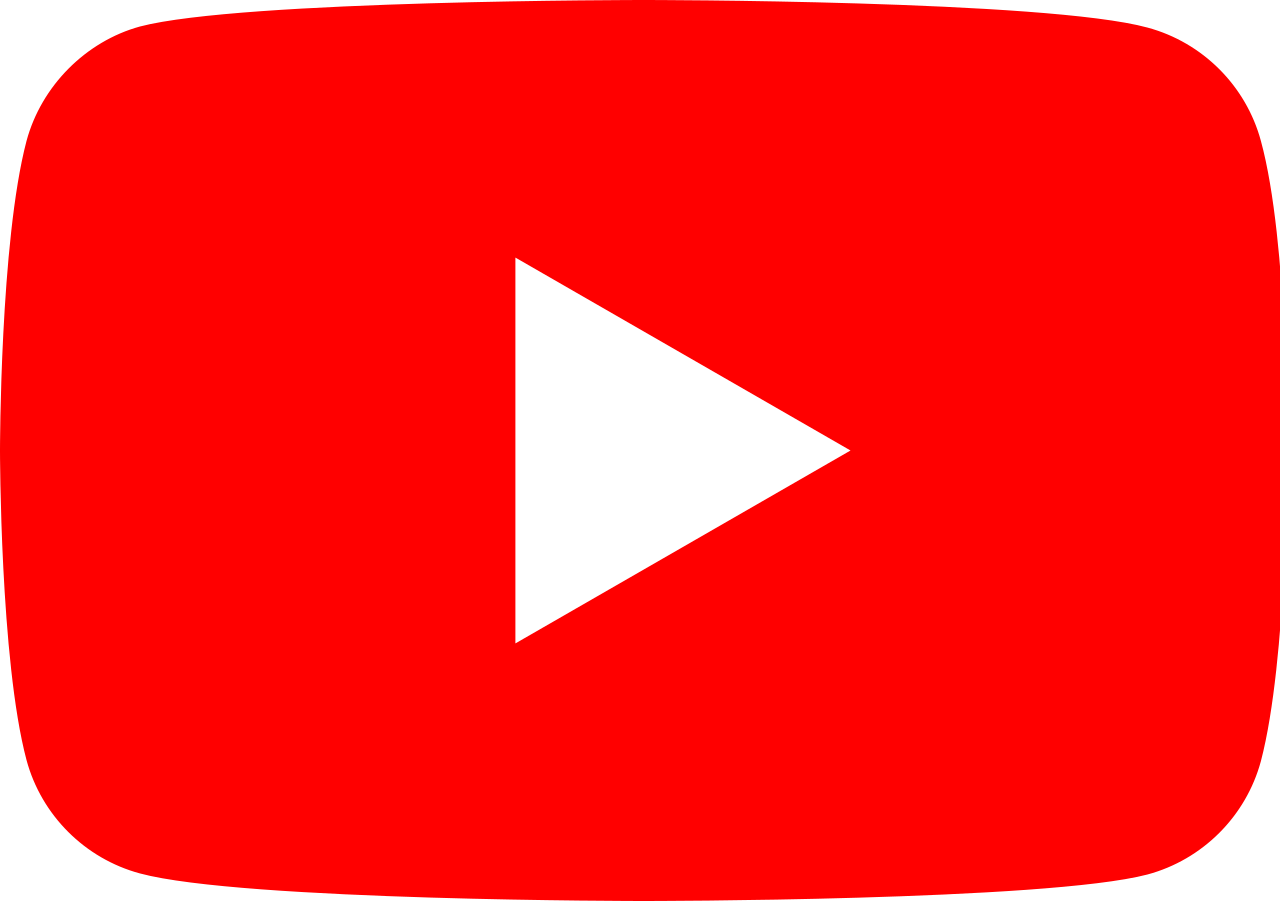
[](https://www.youtube.com/watch?v=g6PDcBhODqo) Cantina

Développeur Front-End / Projet de fin d’études



Sujet

À l’aide du framework JavaScript **React.js**, vous devez réaliser une application de gestion de recettes de cuisine geek.

Votre application devra communiquer avec un serveur web REST et être en mesure de :

* afficher une liste complète des recettes disponibles sur une page
* consulter une recette en détails
* ajouter une nouvelle recette via un formulaire
* modifier une recette existante via un formulaire
* supprimer une recette existante

Le serveur web REST vous est fourni avec ce sujet.

La documentation de cette API se trouve tout en bas de ce brief.

Pour communiquer avec le serveur REST, il est conseillé d’utiliser une bibliothèque tierce comme [Axios](https://www.npmjs.com/package/axios).

Recette

La structure d’une recette dans ce projet est composée des champs suivants :

* Un identifiant (automatiquement généré par le serveur lors d’un ajout)
* Un titre
* Une courte description
* Un niveau de difficulté parmi les possibilités suivantes : “padawan”, “jedi”, “maitre”
* Un nombre de personnes
* Un temps de préparation (en minutes)
* Une liste d’ingrédients nécessaires pour préparer la recette
* Une liste d’étapes pour réaliser la recette
* Une photo de la recette terminée (champs facultatif)

Pages

L’application devra être composée au minimum de 4 pages :

* Page listant les recettes (avec un formulaire offrant des possibilités de filtrage)
* Page affichant le détail d’une recette
* Page d’ajout d’une nouvelle recette (avec un formulaire d’ajout)
* Page de modification d’une recette existante (avec un formulaire d’édition)

1. Page listant les recettes

Cette page devra pouvoir afficher la liste complète des recettes renvoyées par le serveur.

Elle devra comporter un formulaire offrant des possibilités de filtrage.

Ainsi, le formulaire sur cette page devra pouvoir permettre de filtrer les recettes affichées :

* par titre
* par niveau de difficulté
* par nombre de personne (entre X et Y)
* par temps de préparation (moins de X minutes)

Il doit être possible de combiner ces filtres pour faire des recherches précises.

La liste des recettes peut être présentée de la manière qu’il vous convient, cependant il faut au minimum que les champs suivants soient présents :

* titre
* photo (ou une photo générique si le champs n’existe pas)
* niveau de difficulté
* nombre de personne
* temps de préparation
* 2 boutons pour **modifier** ou **supprimer** la recette. Pour la suppression, un mécanisme de confirmation simple devra être proposé (sous forme de boîte de dialogue ou d’une popup).

Chaque recette doit pouvoir être **cliquable** afin d’amener vers la page qui l’affichera en détails.

1. Page affichant une recette

Cette page devra afficher en détail une recette définie. Par exemple, il doit être possible d’y naviguer via une URL du type **/recette/2** (pour afficher la recette portant l’identifiant numéro 2)

Si l’on tente d’afficher une recette avec un identifiant inexistant, un message d’erreur devra être visible sur cette page.

Les champs d’une recette seront affichés de la façon dont il vous plaira, cependant ils devront tous être utilisés.

Veillez à la cohérence de l’affichage de chaque donnée, par exemple :

* temps de préparation : afficher *“1h10“* au lieu de *“70 min”*
* nombre de personnes : afficher *“pour 1 personne”* au lieu de *“pour 1 personne****s****”*

Doivent également être présents sur cette page 2 boutons permettant respectivement de **modifier** et **supprimer** cette recette. Pour la suppression, un mécanisme de confirmation simple devra être proposé (sous forme de boîte de dialogue ou d’une popup).

1. Page d’ajout d’une nouvelle recette

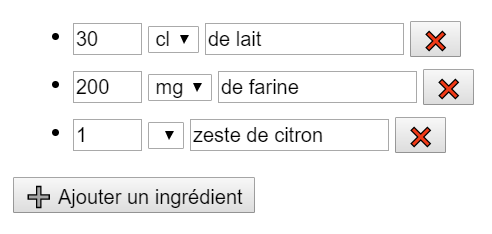
Cette page devra permettre d’ajouter une nouvelle recette au serveur. Elle comprendra un formulaire avec les différents champs pour une recette.

Vous êtes libres de choisir les champs HTML qui vous paraissent les plus adaptés pour le type de données à founir.

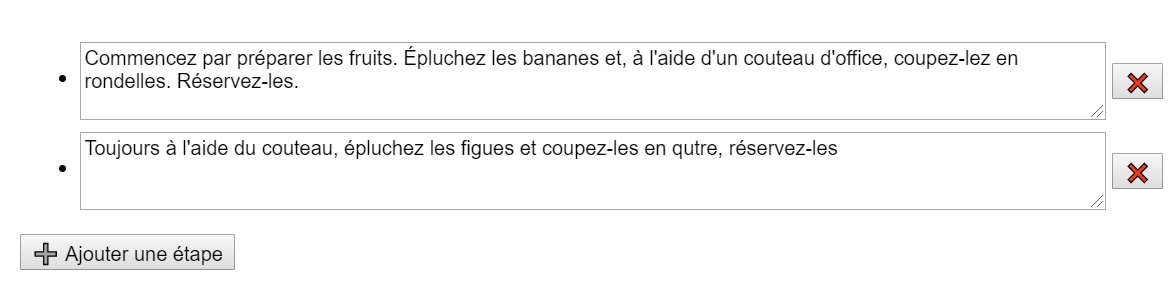
Veillez à bien analyser la structure des données renvoyées par le serveur pour prendre les meilleures décisions.

**Note :** Il doit être possible de renseigner une infinité d’ingrédients et d’étapes. À vous de trouver un système pour cela.

Voici un exemple avec un bouton permettant de gérer une liste d’ingrédients :



Et un exemple pour gérer les étapes :



**Validation des données du formulaire**

À l’aide d’un système de validation de données, vous devez mettre en place sur le formulaire les règles suivantes :

* tous les champs sont obligatoires (sauf le champs “photo” qui est facultatif)
* le champs photo, **s’il est fourni**, doit être une URL commençant par “http://” ou “https://”
* les champs “nombre de personne” et “temps de préparation” doivent être des nombres entiers positifs
* il doit y avoir au moins 1 ingrédient à la recette
* il doit y avoir au moins 1 étape à la recette

1. Page de modification d’une recette existante

Cette page devra permettre de modifier une recette déjà existante. Il devra être possible d’y accéder via une URL du type **/recette/edit/2** (par exemple)

Si la recette n’existe pas, un message d’erreur devra être affiché.

Cette page doit comprendre un formulaire d’édition identique au formulaire d’ajout. Ce formulaire devra être pré-rempli avec les données de la recette que l’on souhaite modifier.

Les validations à mettre en place sont les mêmes que pour l’ajout.

Vous êtes libre de rajouter des fonctionnalités et des options en plus que celles demandées, et de réaliser votre propre design pour l’ensemble de l’application.

Réfléchissez à l’ergonomie de vos pages et à la façon dont les éléments y seront présentés.

Prenez du recul, mais aussi des initiatives (par exemple, des icônes et/ou un code couleur pour donner du contexte aux informations des recettes)

Vous pourrez être amenés à justifier vos choix d’interface lors du Jury.

N’oubliez pas également que chaque potentielle erreur renvoyée par le serveur doit être gérée par l’interface client.

Versionning

Tout au long de votre projet, vous travaillerez avec Git et ferez des commit réguliers et atomiques (= par petites fonctionnalités, pas de gros commits fourre-tout).

Vous veillerez également à pusher régulièrement sur un repository Github public.

L’accès à ce dernier vous sera demandé lors du Jury.

Important : Le dossier du serveur fourni n’est pas à incorporer à votre repository. Vous travaillerez à partir d’un dossier vierge.

Modalités de la semaine de suivi

Ce projet est à réaliser sur une période d’une semaine, en centre de formation, en présence d’un intervenant assurant le suivi technique du projet.

* Vous devez **obligatoirement être présent** durant les heures de suivi.
* La communication et l’échange entre les étudiants est encouragé, cependant le travail à fournir reste **individuel**.

Notation

Seront pris en compte les critères suivant pour la note du projet :

* Application fonctionnelle (respect des fonctionnalités demandés sur chacune des 4 pages).  
  *(8 points)*
* Ergonomie et qualité de l’interface, cohérence de la structure et des choix graphiques  
  *(3 points)*
* Qualité du code HTML, CSS et JavaScript (respect des normes W3C)  
  *(2 points)*
* Maîtrise du framework JavaScript Vue.js  
  *(4 points)*
* Utilisation conforme des promesses JavaScript pour la programmation asynchrone  
  *(2 points)*
* Utilisation conforme de Git et Github  
  *(1 point)*

Conseils

* Préparer les écrans de votre application (wireframes, mockups)
* Organisez vous et planifiez votre travail : donnez vous des objectifs intermédiaires
* Mettez en oeuvre les bonnes pratiques vues en cours
* Refactorisez pour éviter le code redondant
* Faites de petits commits réguliers plutôt que de gros commits fourre-tout
* Soignez la qualité de votre code (commentaires, indentation)
* Pensez facilité d’utilisation et qualité du résultat.

Documentation API du serveur REST

Le serveur REST permettant de gérer les recettes de cuisine vous est fourni dans le dossier /em-cantina-server/

Pour installer le serveur, ouvrez un terminal et déplacez-vous à l’emplacement de ce dossier. Puis exécutez la commande **npm install** pour installer les dépendances.

Pour lancer le serveur REST, exécutez ensuite la commande **npm run server**.

Le serveur démarrera alors et écoutera toutes les requêtes arrivants sur **http://localhost:9000/api/**

*Note : Le code source de votre projet client doit être indépendant de celui du serveur.*

Récupérer toutes les recettes

| URL | [**http://localhost:9000/api/recipes**](http://localhost:9000/api/recipes) | Méthode | **GET** |
| --- | --- | --- | --- |
| ✅ Réponse en cas de succès : | | | |
| Code HTTP 200 avec un array contenant une liste d’objets “recette”  Exemple :   | [  {  "id": 1,  "titre": "Dustcrepe",  "description": "Croustillant menthe, feta et concombre",  "niveau": "padawan",  "personnes": 4,  "tempsPreparation": 15,  "ingredients": [ ["1","pincée de sel"] , ["200cl","de lait"] ],  "etapes": [  "Commencez par préparer le concombre ...",  "Passez maintenant à l'appareil qui farcira ..."  ],  "photo": "http://localhost:9000/images/dustcrepe.jpg"  },  // … etc.  ] | | --- | | | | |
| ❌ Réponse en cas d’erreur : | | | |
| Code HTTP 500 avec un objet JSON Exemple :   | { errorMessage : "Intitulé de l’erreur" } | | --- | | | | |

Récupérer une seule recette

| URL | **http://localhost:9000/api/recipe/:id** | Méthode | **GET** |
| --- | --- | --- | --- |
| ✅ Réponse en cas de succès : | | | |
| Code HTTP 200 avec un objet représentant la recette demandée  Exemple pour <http://localhost:9000/api/recipe/1> :   | {  "id": 1,  "titre": "Dustcrepe",  "description": "Croustillant menthe, feta et concombre",  "niveau": "padawan",  "personnes": 4,  "tempsPreparation": 15,  "ingredients": [ ["1","pincée de sel"] , ["200cl","de lait"] ],  "etapes": [  "Commencez par préparer le concombre ...",  "Passez maintenant à l'appareil qui farcira ..."  ],  "photo": "http://localhost:9000/images/dustcrepe.jpg"  } | | --- | | | | |
| ❌ Réponse en cas d’erreur : | | | |
| Code HTTP 404 avec un objet JSON Exemple :   | { errorMessage : "Aucune recette trouvée" } | | --- | | | | |

Créer une recette

| URL | **http://localhost:9000/api/recipes** | Méthode | **POST** |
| --- | --- | --- | --- |
| Données à envoyer (au format JSON) : | | | |
| * **titre** (String) : Le titre de la recette * **description** (String) : Une courte description de la recette * **niveau** (String) : Un indicateur de difficulté parmi : “padawan”, “jedi” ou “maitre” * **personnes** (Number) : Un nombre indiquant pour combien de personne est la recette * **tempsPreparation** (Number) : Un nombre indiquant le temps de préparation en minutes * **ingredients** (Array<Array<String, String>>) : Un tableau à deux dimensions contenant une liste d’ingrédients. Chaque ingrédient est représenté par un tableau contenant sa quantité et son nom. Par exemple :  | [  [ "1" , "pincée de sel" ],  [ "200cl", "de lait" ],  [ "", "poivre" ] ] | | --- |  * **etapes** (Array<String>) : Un tableau de chaînes de caractères représentants les différentes étapes à suivre pour réaliser la recette. Par exemple :  | [  "Commencez par préparer le concombre ...",  "Passez maintenant à l'appareil qui farcira ..." ] | | --- |  * **photo** (String) : (FACULTATIF) Chaîne de caractères commençant par “http” ou “https” qui pointe vers l’URL absolue de la photo. (Le serveur ne fait pas d’hébergement d’image) | | | |
| ✅ Réponse en cas de succès : | | | |
| Code HTTP 201 avec un message et l’objet représentant la recette nouvellement créée  Exemple :   | {  "message": "Recette créée avec succès !",  "recette": {  /\* Objet recette créée  ... \*/  }  } | | --- | | | | |
| ❌ Réponse en cas d’erreur : | | | |
| Code HTTP 400 avec un objet JSON Exemple :   | { errorMessage : "Champs \"titre\" manquant" } | | --- | | | | |

Modifier une recette

| URL | **http://localhost:9000/api/recipe/:id** | Méthode | **PUT** |
| --- | --- | --- | --- |
| Données à envoyer (au format JSON) : | | | |
| ℹ Il n’est pas nécessaire d’inclure tous les paramètres.  Par exemple, si vous souhaitez ne modifier que la description, n’envoyez que le paramètre “*description*”.  En revanche, au moins un paramètre doit être envoyé pour que la modification ait lieu.   * **titre** (String) : Le titre de la recette * **description** (String) : Une courte description de la recette * **niveau** (String) : Un indicateur de difficulté parmi : “padawan”, “jedi” ou “maitre” * **personnes** (Number) : Un nombre indiquant pour combien de personne est la recette * **tempsPreparation** (Number) : Un nombre indiquant le temps de préparation en minutes * **ingredients** (Array<Array<String, String>>) : Un tableau à deux dimensions contenant une liste d’ingrédients. Chaque ingrédient est représenté par un tableau contenant sa quantité et son nom. Par exemple :  | [  [ "1" , "pincée de sel" ],  [ "200cl", "de lait" ],  [ "", "poivre" ] ] | | --- |  * **etapes** (Array<String>) : Un tableau de chaînes de caractères représentants les différentes étapes à suivre pour réaliser la recette. Par exemple :  | [  "Commencez par préparer le concombre ...",  "Passez maintenant à l'appareil qui farcira ..." ] | | --- |  * **photo** (String) : (FACULTATIF) Chaîne de caractères commençant par “http” ou “https” qui pointe vers l’URL absolue de la photo. (Le serveur ne fait pas d’hébergement d’image) | | | |
| ✅ Réponse en cas de succès : | | | |
| Code HTTP 200 avec un message et l’objet représentant la recette mise à jour.  Exemple :   | {  "message": "Recette mise à jour avec succès !",  "recette": {  /\* Objet recette mis à jour  ... \*/  }  } | | --- | | | | |
| ❌ Réponse en cas d’erreur : | | | |
| Code HTTP 400 ou 404 avec un objet JSON Exemple :   | { errorMessage : "Aucune recette trouvée" } | | --- | | | | |

Supprimer une recette

| URL | **http://localhost:9000/api/recipe/:id** | Méthode | **DELETE** |
| --- | --- | --- | --- |
| ✅ Réponse en cas de succès : | | | |
| Code HTTP 200 avec un message et l’objet représentant la recette supprimée.  Exemple :   | {  "message": "Recette supprimée avec succès !",  "recette": {  /\* Objet recette supprimé  ... \*/  }  } | | --- | | | | |
| ❌ Réponse en cas d’erreur : | | | |
| Code HTTP 400 ou 404 avec un objet JSON Exemple :   | { errorMessage : "Aucune recette trouvée" } | | --- | | | | |