

Dans l'épisode précédent du fil rouge, nous avons reconçu notre application en utilisant des composants React.

Dans cette partie vous utiliserez les Hook pour lier le front-end de l'application à une api fournissant les données.

L'adresse racine de l'api est la suivante <http://kazaburger.e-mingo.net/api>

Dans la suite de l'exercice ne vous sera précisé que le suffixe(ressource), à ajouter à l'adresse racine, pour récupérer les données spécifiques.

Par exemple avec la ressource **/toto**, vous interrogerez <http://kazaburger.e-mingo.net/api/toto>

I. Le composant <Suggests>

- Dans le FilRouge#02, les données ont été injectées en dur dans la variable data

```
import React from "react";
import { SuggestItem } from "../Suggests/SuggestItem";

const Suggests = () => {
  const data = [
    {
      image: "src/assets/images/classic-burger.jpg",
      imageAlt: "Classic burger",
      title: "Classic burger",
      description:
        "Le classic Burger est un délicieux sandwich à bas",
      price: 6.9,
    },
  ],
};
```

- Nous allons donc modifier le composant pour récupérer les données fournies par la ressource « **/suggest** » de l'api
- Quel que soit le nombre de résultats envoyés par l'api, vous n'afficherez que 3 éléments 😊
- Dans le dossier « src », créez un nouveau dossier « services »
- Ici nous placerons nos appels aux services externes, comme les api, etc ..
- Placez-y le fichier « useFetch.js », cf TP5
- Modifiez le composant pour afficher et récupérer les données
- NB : La structure du json fourni par l'api est différente de celle du TP4
- NB2 : Si l'image n'existe pas, affichez une image générique

Nos suggestions



Tartellettes cantal
chocolat

Goûtez notre Tartellettes cantal chocolat, où le bacon croustillant se marie parfaitement avec des champignons savoureux. Une combinaison gagnante pour tous les palais !

19.66 €



Burger Karadoc (16
ingrédients)

Notre Burger Karadoc (16 ingrédients) est parfait pour une expérience gustative audacieuse. Avec sa sauce piquante, il ravira vos papilles et vous laissera en redemander !

6.85 €



Pizza bacon-mozzarella

Savourez notre Pizza bacon-mozzarella, pleine de légumes frais colorés. Une option saine et délicieuse qui ravira vos papilles et vous apportera du bien-être.

16.22 €

Pour éviter d'avoir un affichage monotone, il est possible de mélanger les données récupérées

```
/**
 * On mélange et on retourne une partie
 * @param {*} array
 * @param {*} length
 * @returns
 */
function shuffleArray(array, length=0) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }

  return length?array.slice(0, length):array;
}

export { shuffleArray };
```

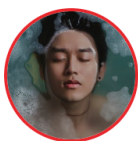
Cette fonction reçoit en paramètres :

- Le tableau contenant les données à mélanger
- Le nombre d'éléments à retourner 😊

II. Le composant <Testimonies>

- Modifiez le composant pour récupérer les données à partir de la ressource « **/testimony** »
- Pour les photos de profil, vous pourrez utiliser des générateurs comme, par exemple, <https://pravatar.cc/> ou <https://avatar-placeholder.iran.liara.run/>
- Quel que soit le nombre de résultats envoyés par l'api, vous n'afficherez que 4 éléments 😊

Nos clients témoignent



Sarah G.
★★★★★

Le burger était bon, mais la viande manquait un peu de jutosité. - Sarah G.



Nicolas L.
★★★★★

Un bon burger, bien garni et savoureux. Le pain était moelleux et les frites étaient bien chaudes. - Nicolas L.



Camille P.
★★★★★

Déçu par la pizza, la pâte était trop cuite et la garniture manquait de saveur. - Camille P.



Thomas B.
★★★★★

Déçu par le burger, la viande était trop cuite et sèche. - Thomas B.

III. Le composant <Featured>

Ce composant utilisera la ressource « **/featured** » pour récupérer les données afin de les afficher sous la forme d'un Carrousel



NB : L'image a été enlevée au profit de l'animation du fond de la section directement

- Utilisez useState pour créer 2 états afin de gérer :
 - l'indice dans le tableau pour la diapo en cours d'affichage : « **currentIndex** »
 - si le carrousel est en mode autoplay : « **isPlaying** »
- Utilisez useRef pour cibler la section « featured » et changer son image de fond
- Créez 2 fonctions, pour changer de diapo:
 - « **nextSlide** » pour passer à la diapositive suivante
 - « **prevSlide** » pour passer à la diapo précédente .
- Créez des fonctions pour gérer les événements :
 - « **slideOver** » : au survol de la zone (mouseenter), arrêt de l'animation
 - « **slideOut** » : A la sortie de la zone (mouseleave), reprise de l'animation
 - « **PrevNextClick** » : Au clic sur les boutons (arrêt de l'animation et changement de la diapo)
- Si les données sont chargées, le carrousel est en mode autoplay. Suivant un intervalle défini par défaut à 5s, il passe à la diapositive suivante.

Exemple de classes CSS utilisées pour le bon positionnement des boutons de navigation

```
.slide@apply relative flex flex-col md:flex-row justify-evenly h-full
mx-auto overflow-hidden px-20 md:items-center bg-black/20 ;
```

```
button {@apply absolute cursor-pointer top-1/2 transform -translate-y-1/2 text-
white/20 hover:text-secondary/50 text-7xl mx-10;}
```

De même pour faire mieux ressortir le texte dans les diapos, le fond des diapos est noir avec 20% d'opacité, le contenu (texte a une légère transparence et une petite ombre portée