

React.js with PHP API REST

8. Validation du formulaire onSubmit et useRef hook

Nous ajoutons la validation du formulaire qu'il était demandé de faire à la fin du TP 7. Voici une proposition de correction pour une validation onSubmit

Nous allons utiliser le hook useRef pour créer une référence vers les éléments du DOM auxquels nous avons besoin d'accéder

```
src > components > contact > Contact.jsx > Contact
1  import { useRef } from "react";
2
3  function Contact() {
4
5      const fullnameInputRef = useRef();
6      const emailInputRef = useRef();
7      const messageInputRef = useRef();
8      const errorOutputRef = useRef();
9      const closeBtnRef = useRef();
```

Ressources :

<https://react.dev/reference/react/useRef>

Nous ajoutons les refs dans le JSX, d'abord sur le bouton close du modal bootstrap pour pouvoir le fermer une fois le formulaire validé, traité, envoyé et la réponse à la requête vers l'API Rest reçue.

```
<div className="modal fade" id="myModal" tabIndex="-1" aria-labelledby="exampleM
  <div className="modal-dialog">
    <div className="modal-content">
      <div className="modal-header">
        <h1 className="modal-title fs-5" id="exampleModalLabel">Contacte
        <button
          type="button" className="btn-close" data-bs-dismiss="modal"
          aria-label="Close" ref={closeBtnRef}>
        </button>
      </div>
      <div className="modal-body">
```

-> voir page suivante

Nous ajoutons ensuite les refs sur les inputs du formulaire

```
<form id="contactForm" onSubmit={handleFormSubmit} noValidate>
  <div className="mb-3">
    <label className="form-label" htmlFor="fullname">Name</label>
    <input
      className="form-control" name="fullname" type="text"
      placeholder="Name" ref={fullnameInputRef}
    />
  </div>
  <div className="mb-3">
    <label className="form-label" htmlFor="email">Email Address</label>
    <input
      className="form-control" name="email" type="email"
      placeholder="Email Address" ref={emailInputRef}
    />
  </div>
  <div className="mb-3">
    <label className="form-label" htmlFor="message">Message</label>
    <textarea
      className="form-control" name="message" type="text"
      placeholder="Message" style={{height: '10rem'}}
      ref={messageInputRef}
    >
  </textarea>
  </div>
  <div className="mb-1 text-center">
    <button className="btn btn-success" name="send" type="submit">
      Envoyer
    </button>
    <br/>
    <div className="text-danger mt-2" ref={errorOutputRef}></div>
  </div>
</form>
```

Nous avons ajouté un div avec une ref en dessous le bouton submit afin d'afficher les messages d'erreurs éventuels.

Nous avons également ajouté l'attribut noValidate pour que le formulaire ne soit plus validé par le navigateur mais par nous même.

-> voir page suivante

Nous allons maintenant créer la méthode `handleFormValidation()` afin de valider notre formulaire

```
src > components > contact > Contact.jsx > Contact
3  function Contact() {
10
11      const handleFormValidation = () => {
12          const isValid = {
13              fullname: false,
14              email: false,
15              message: false
16          }
17          const fullnameValue = fullnameInputRef.current.value;
18          isValid.fullname = fullnameValue.length >= 1;
19          const emailValue = emailInputRef.current.value;
20          const emailPattern = /^[w-]+@[w-]+\.[w-]{2,4}$/;
21          isValid.email = emailPattern.test(emailValue);
22          const messageValue = messageInputRef.current.value;
23          isValid.message = messageValue.length >= 1;
24          const formIsValid = Object.values(isValid).every(v => v == true);
25          let errorMessage = !isValid.fullname ? "Nom incorrect (1 caractère minimum)" :
26                          !isValid.email ? "Email incorrect (format d'email invalide)" :
27                          !isValid.message ? "Message incorrect (1 caractère minimum)" : "";
28          errorOutputRef.current.innerText = errorMessage;
29          return formIsValid;
30      }
31  }
```

Nous créons un objet pour stocker la validité des différents inputs du formulaire (ligne 12 à 16)

Nous récupérons la valeur de l'input fullname (ligne 17) et mettons à jour sa validité (ligne 18)

Ici nous avons décidé qu'il doit contenir au moins un caractère.

Nous récupérons la valeur de l'input email et vérifions sa validité à l'aide d'un regex (ligne 19 à 21)

Puis nous faisons de même avec la valeur de l'input message (ligne 22 et 23)

Nous vérifions que tous les inputs du formulaire sont valides (ligne 24), c'est cette valeur (true ou false) qui sera renvoyée par la méthode `handleFormValidation()` (ligne 29)

Enfin (entre les lignes 25 et 28) nous affichons un éventuel message d'erreur dans le div prévu à cet effet en fonction de la validité des valeurs des inputs. Nous traitons le formulaire de haut en bas.

-> voir page suivante

Nous modifions ensuite la méthode `handleFormSubmit()` pour valider le formulaire avant envoi.

```
src > components > contact > Contact.jsx > Contact
3  function Contact() {
31 |
32 |     const handleFormSubmit = (event) => {
33 |         event.preventDefault();
34 |         const formData = new FormData(event.target);
35 |         const jsonData = Object.fromEntries(formData.entries());
36 |         console.log(jsonData);
37 |         if(!handleFormValidation()){
38 |             console.log("Formulaire non valide");
39 |             return;
40 |         }
41 |         const submitData = async () => {
42 |             const url = "http://api.php-blog-project.loc/contact";
43 |             const options = {
44 |                 method: "POST",
45 |                 body: JSON.stringify(jsonData),
46 |             }
47 |             const response = await fetch(url, options);
48 |             console.log(response);
49 |             if (!response.ok) {
50 |                 throw new Error('Erreur de réseau');
51 |             }
52 |             const json = await response.json();
53 |             console.log(json.result);
54 |             closeBtnRef.current.click();
55 |             fullnameInputRef.current.value = "";
56 |             emailInputRef.current.value = "";
57 |             messageInputRef.current.value = "";
58 |         }
59 |         submitData();
60 |     };

```

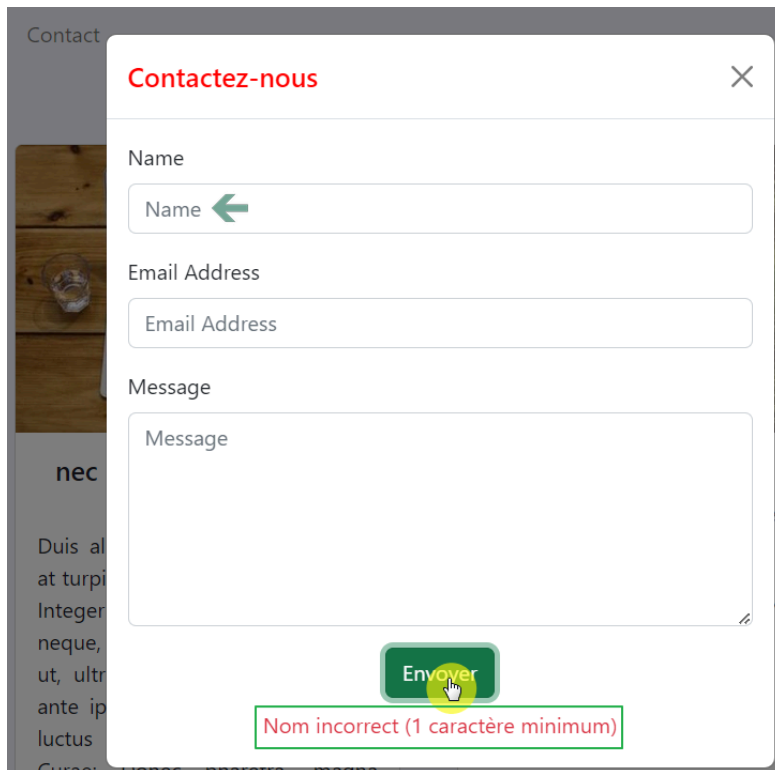
Si la méthode `handleFormValidation()` ne renvoie pas `true` (lignes 37 à 40), nous affichons un message dans la console et arrêtons l'exécution de la méthode `handleFormSubmit()`, les données n'étant pas conformes, rien ne sert de les envoyer vers le serveur.

Si tout se passe bien (pas d'exception déclenchée ligne 50) nous pouvons fermer le formulaire à l'aide de la ref sur le bouton close (ligne 54) et réinitialiser les valeurs des inputs du formulaire (ligne 55 à 57)

Nous pouvons maintenant tester le formulaire et vérifier dans la console.

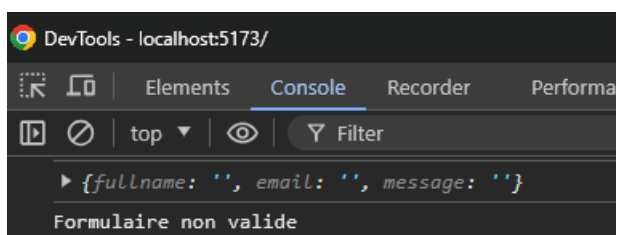
-> voir page suivante

Avec un nom incorrect (aucun caractère)

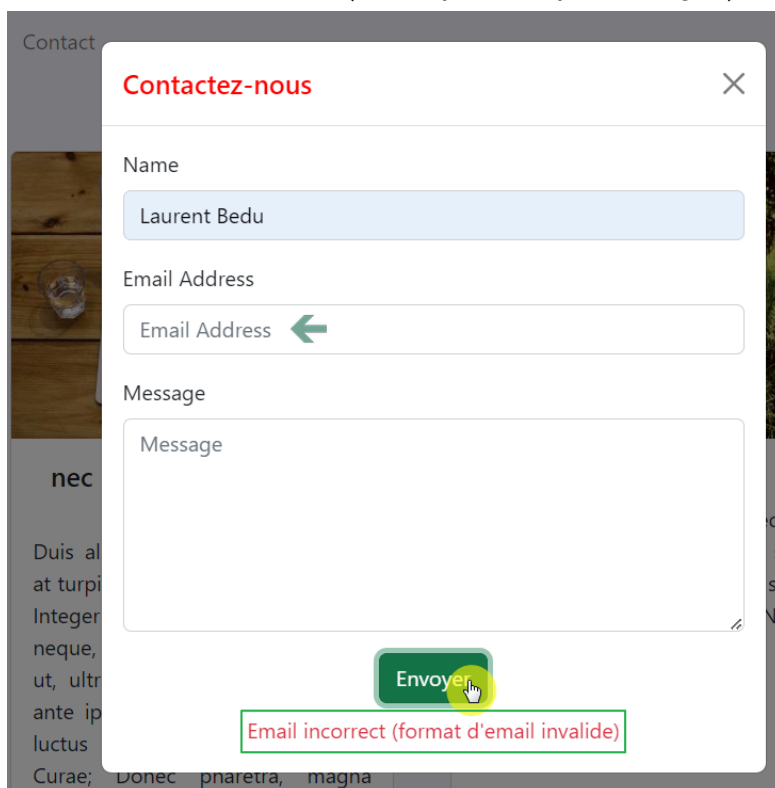


The image shows a contact form titled "Contactez-nous" with a close button (X). It contains three input fields: "Name", "Email Address", and "Message". The "Name" field is empty and has a green arrow pointing left, indicating a validation error. Below the "Envoyer" button, a red error message is displayed: "Nom incorrect (1 caractère minimum)".

Résultat en console

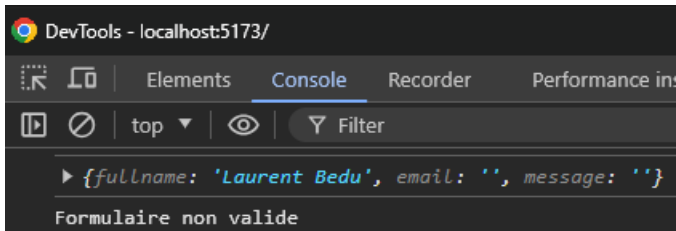


Avec un email incorrect (ne respectant pas le regex)

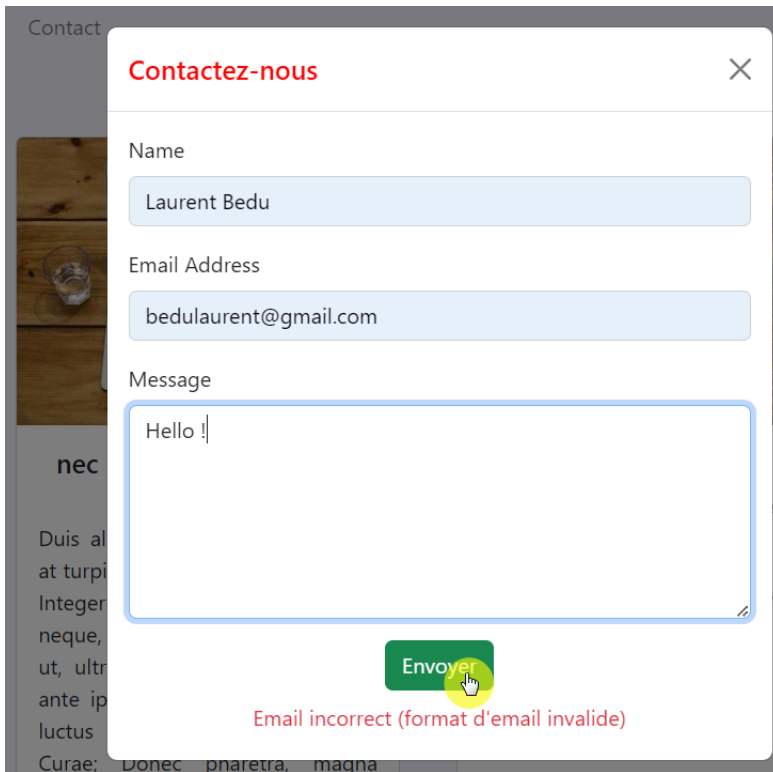


The image shows the same contact form, but now the "Name" field is filled with "Laurent Bedu". The "Email Address" field is empty and has a green arrow pointing left, indicating a validation error. Below the "Envoyer" button, a red error message is displayed: "Email incorrect (format d'email invalide)".

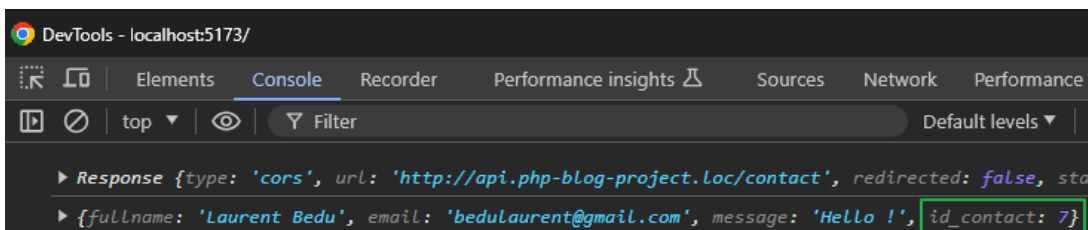
Résultat en console



Avec tous les champs valides



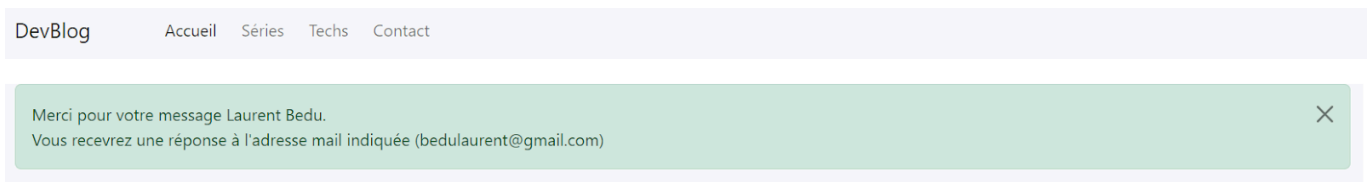
Résultat en console



La modale s'est fermé et la ligne a bien été insérée en DB avec l'id 7

Si vous ouvrez à nouveau le formulaire, vous constaterez qu'il a bien été réinitialisé.

Pour améliorer encore le traitement de notre formulaire, il serait bien d'afficher un message à l'utilisateur quand tout s'est bien passé (comme nous l'avons fait sur l'application PHP MVC)



A vous de coder cette fonctionnalité.

git : https://github.com/DWWM-23526/REACT_BLOG_PROJECT/tree/Step08