UNIVERSITE ASSANE SECK DE ZIGUINCHOR

UFR : SCIENCES ET TECHNOLOGIES

DEPARTEMENT : INFORMATIQUE

FILIERE : L2I

NIVEAU : LICENCE 3

RAPPORT DE TEST DU FRONT-END DE L'APPLICATION EVENT'S PLANNING

Souleymane DIAGNE | Aliou DIOP

Baye Mor DIOUF | Mareme THIAM

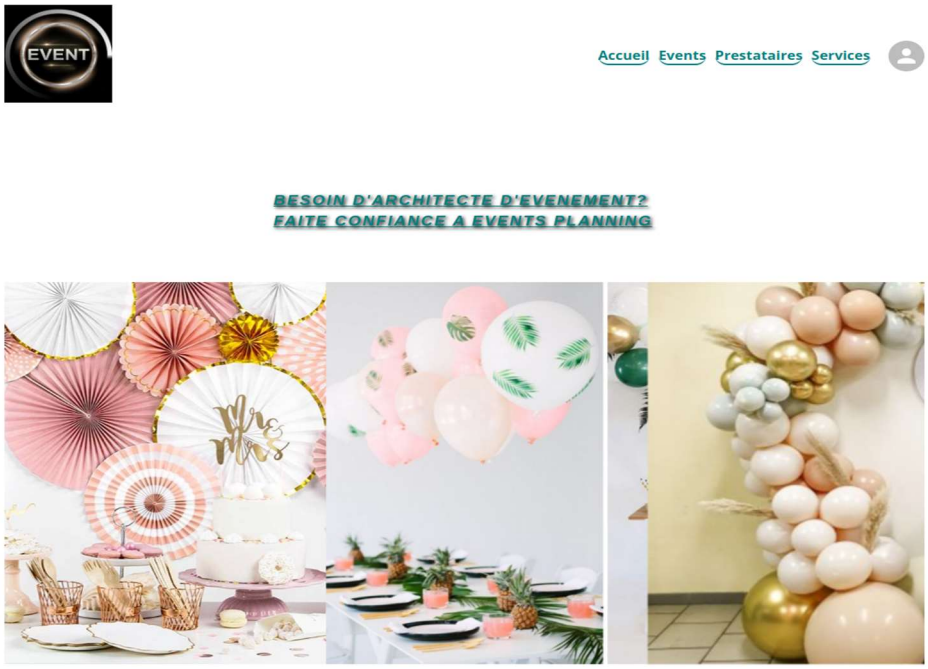
ETUDIANTS

Professeur : Mr Toure

Introduction :

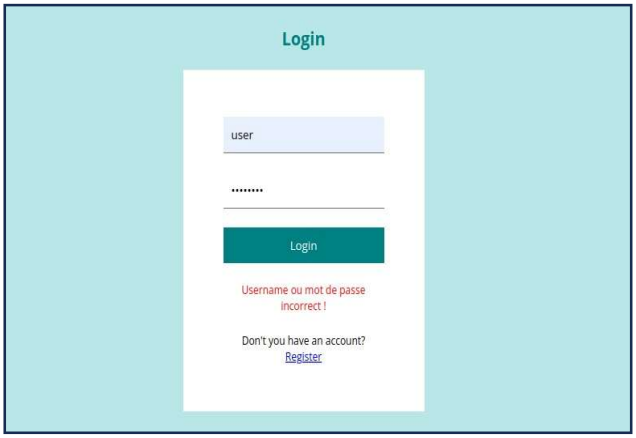
Event's planning est une application visant à faciliter la tâche à toute personne responsable d'un évènement, qu'il soit religieux, familial, professionnel ou même sportif. Le client doit pouvoir créer un évènement et choisir les prestataires dont il a besoin pour la gestion de son évènement.

- **Page d'accueil :**



Cette page renvoie le menu de l'application à savoir les évènements (les types d'évènements gérer par l'application), et les services que proposent les prestataires figurant dans l'application. L'icône à côté de services permet à l'utilisateur de se connecter s'il a déjà un compte sinon de le créer .

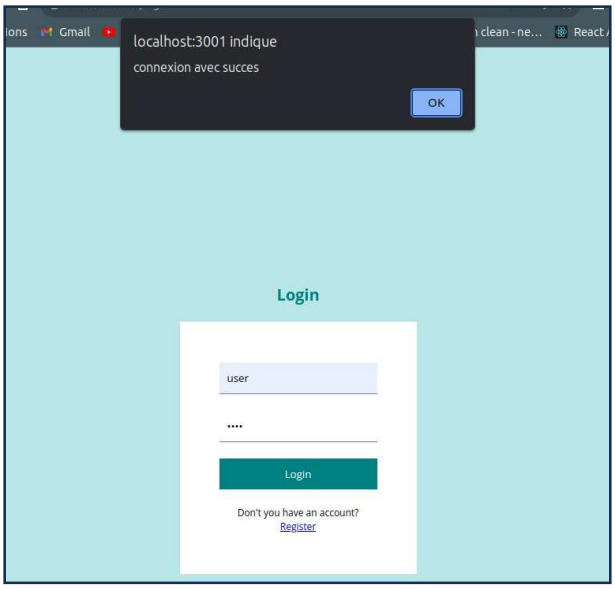
- **Page de connexion :**




```
const addUser = user =>{
  fetch("http://localhost:8080/api/users/create",{
      method: "POST" ,
      headers: {"content-Type" : "application/json"},
      body: JSON.stringify(user) ,
      })
      .then(response => {
        if (response.ok) {
         alert("user creer") ;
        }
        else {

         alert("Cette email est deja utiliser  !") ;
        }

      })
      .catch(err => alert(err)) ;
      alert("s")

};
```

On sait qu'une connexion dans l'application nécessite forcément un login et un mot de passe, d'où l'obligation de créer un compte utilisateur en cliquant sur Register.

```
const [verifier, setVerifier] = useState(true);

  const connexion = async (compte) =>{
     fetch("http://localhost:8080/login",{
        method: "POST" ,
        headers: {"content-Type" : "application/json"},
        body: JSON.stringify(compte) ,
        })
        .then(response => {
           const jwtToken = response.headers.get("Authorization");
           if (jwtToken != null) {
              sessionStorage.setItem("jwt", jwtToken) ;
              alert(jwtToken) ;
           }
           else
           {
             alert("mot de pass incorect !") ;
           }

        })
        .catch(err =>alert(err) ) ;
        alert(verifier) ;
};
```

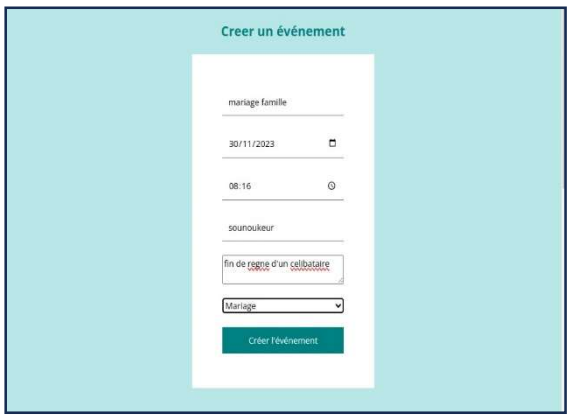


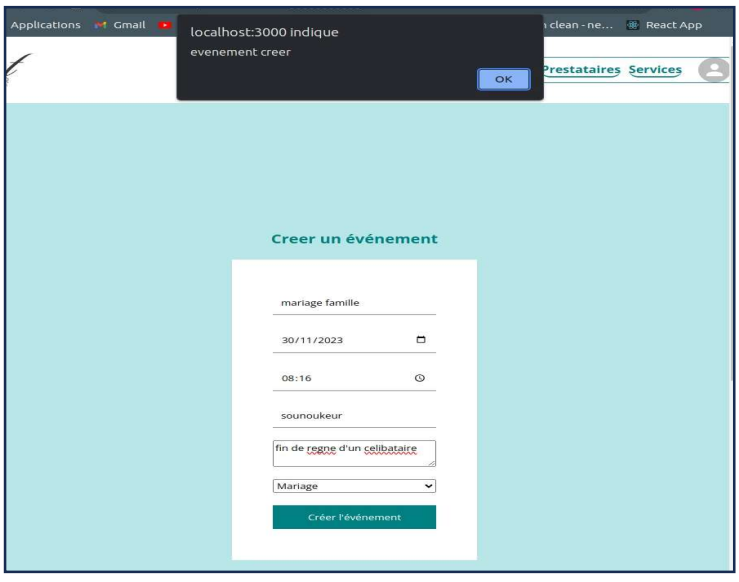
Apres avoir créer son compte le client peut maintenant accéder à l'application en se connectant avec son login et mot de passe.

- **Création d'évènement :**

Une fois que le client a accès à l'application il peut maintenant procéder à la création de son évènement. Pour ce, un formulaire de création d'évènement est mis à sa disposition ou il précisera toutes informations caractérisant l'évènement dont la création est notifiée par un message d'alerte.

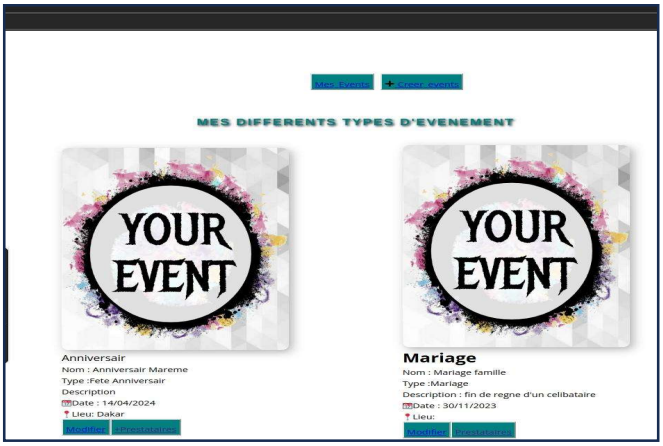
```
const addEvent = event1 =>{
 fetch("http://localhost:8080/api/evenement/create",{
      method: "POST" ,
      headers: {"content-Type" : "application/json"},
      body: JSON.stringify(event1) ,
      })
      .then(response => {
         if (response.ok) {
          alert("evenement creer") ;
         }
         else {
          alert("Quelque chose c'est mal passe !") ;
         }

       })
      .catch(err => alert(err)) ;
      alert(event1.user.username) ;
};

/****************************************/
```





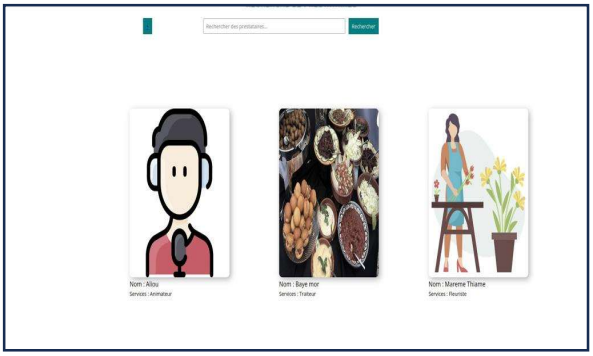
Apres création, l'évènement est relié à l'utilisateur qui peut accéder a une liste ne contenant pour l'instant que de l'évènement qu'il vient de créer mais qui sera incrémenter de nouveaux évènements au fur et à mesure que ce client utilise l'application et en crée de nouveau.

```
const [evenement , setEvenement] = useState([]) ;

const fetchEvennement = () =>{

  fetch('http://localhost:8080/api/users/'+{id}+'/evenements')
    .then(response => response.json())
    .then(data => setCars(data._embedded.evenements))
    .then(err => console.error(err)) ;
};
```



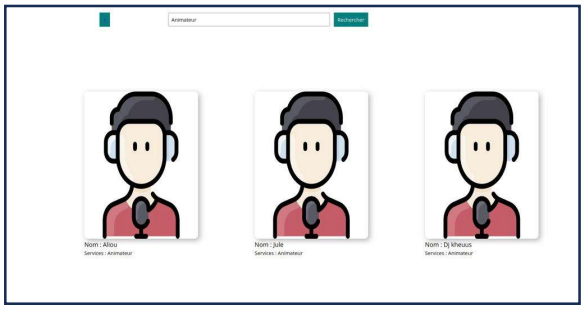
- **Recherche des prestataires :**

Le client dispose d'une interface composé d'un onglet de recherche lui permettant de rechercher et de choisir des prestataires et de la liste des prestataires disponibles.



```
const fetchPrestataire = () =>{

  fetch("http://localhost:8080/api/prestataires/recherche")
    .then(response => response.json())
    .then(data => setPrestataire(data._embedded.evenements))
    .then(err => console.error(err)) ;
    alert("ca passe !")
};
```

Lorsque le client effectue une recherche selon le type de service fourni, une liste des prestataires fournissant ce service est renvoyée.
Par exemple ici on recherche un animateur et ceux disponibles sont Aliou, Jule et Dj kheus .



Lorsqu'on recherche un traiteur on a comme résultats Baye Mor Diouf, Diop et frères et Absa Ndiaye.