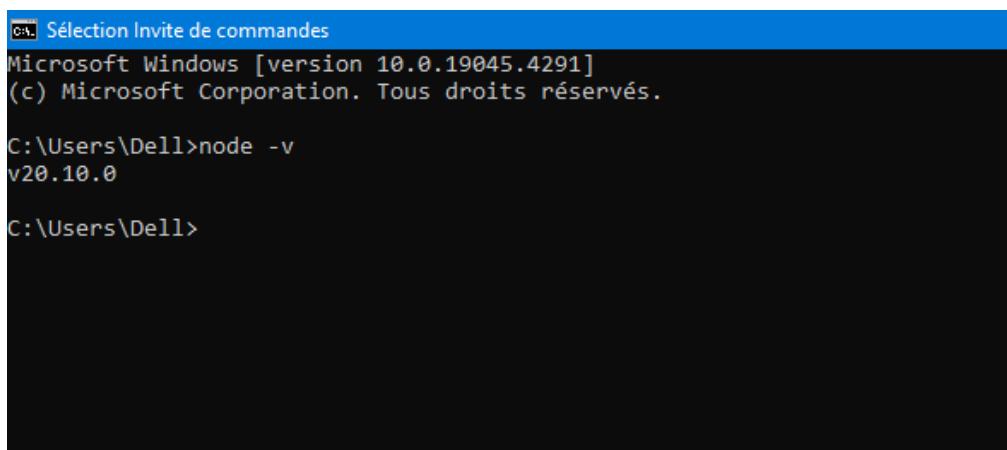


Rapport Vue.js

➤ Crédit à la création d'un projet Vue.js :

Installation de Node.js :

Node.js est une plateforme logicielle permettant d'exécuter du code JavaScript côté serveur. Après avoir installé Node.js via le lien [suivant](#), vous pouvez vérifier l'installation dans n'importe quel terminal (cmd ou terminal VSCode) en utilisant la commande « node -v ». Cette commande affiche la version de Node.js installée, confirmant ainsi une installation réussie.



```
C:\ Sélection Invité de commandes
Microsoft Windows [version 10.0.19045.4291]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\DELL>node -v
v20.10.0

C:\Users\DELL>
```

Installation de vue CLI :

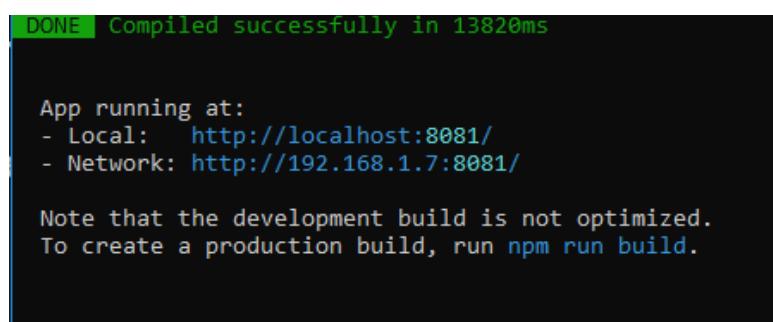
Vue CLI est un outil de ligne de commande pour le développement rapide d'une application Vue.js.

On l'installe en utilisant la commande « npm install -g@vue/cli ». (L'installation est globale à l'aide de l'indicateur -g).

On vérifie l'installation en utilisant la commande « vue --version ».

On peut maintenant créer notre projet vue par la commande « vue create -le nom de projet- »

Ensuite, on lance l'exécution avec la commande « npm run serve » dans le répertoire du projet, ce qui démarre un serveur de développement local sur le port 8081.

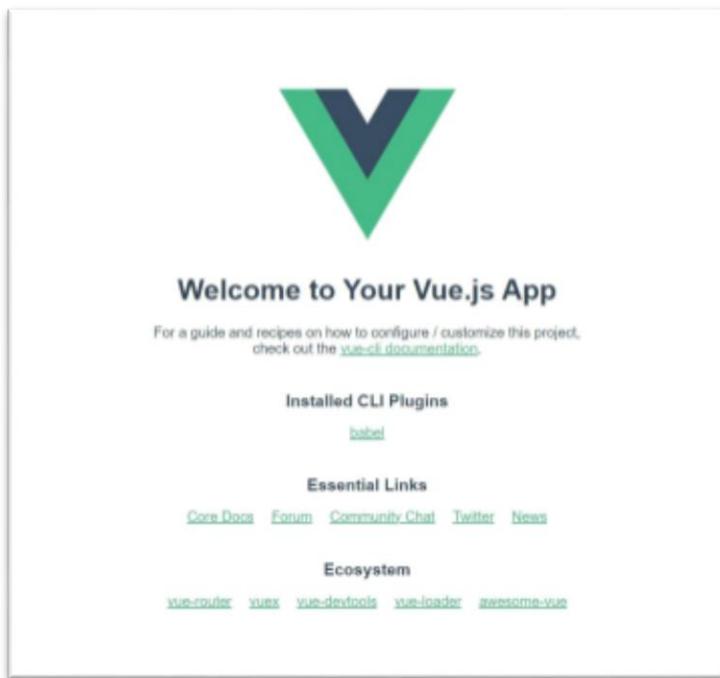


```
DONE Compiled successfully in 13820ms

App running at:
- Local: http://localhost:8081/
- Network: http://192.168.1.7:8081/

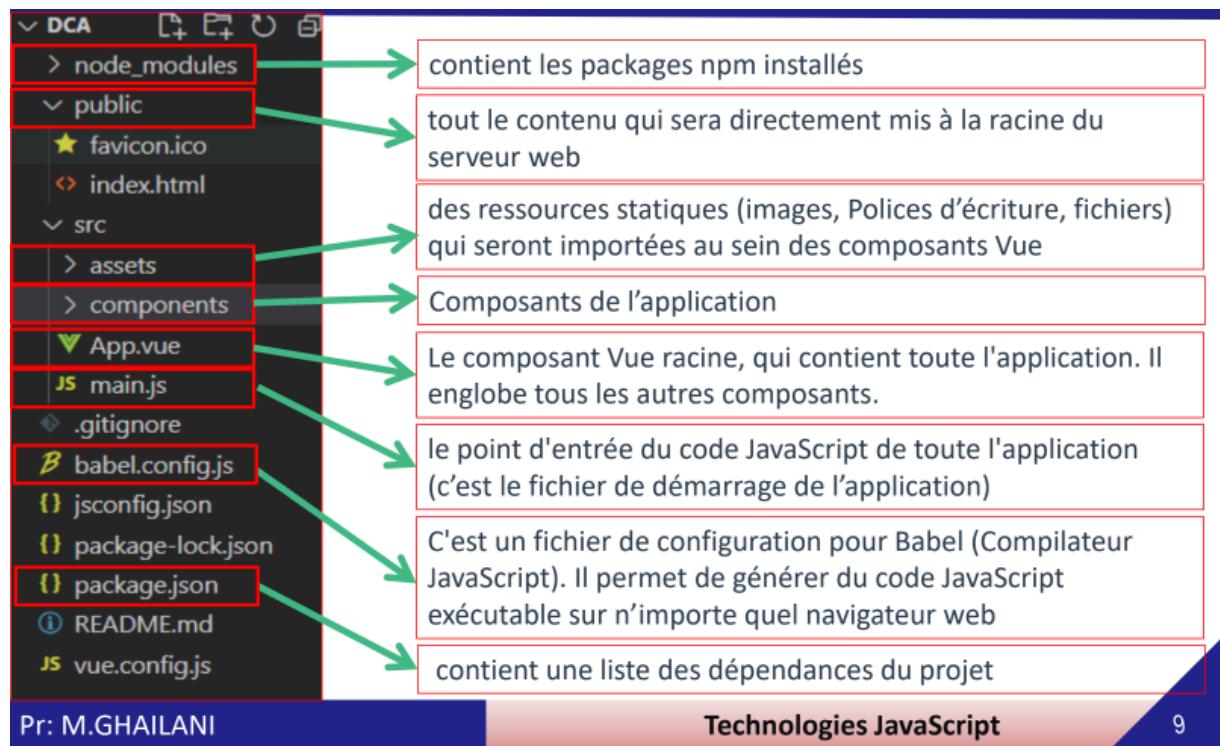
Note that the development build is not optimized.
To create a production build, run npm run build.
```

En accédant à « `localhost:8081` », on trouve la structure de la page qui est générée par défaut.



Structure de Projet par défaut :

Lorsque vous créez un projet Vue.js, les fichiers initiaux sont générés pour vous aider à structurer et configurer votre application.



La partie la plus importante est « components » par ce qu'il va contenir tous les composants de notre application.

➤ Installation d'axios et router :

Installation Router :

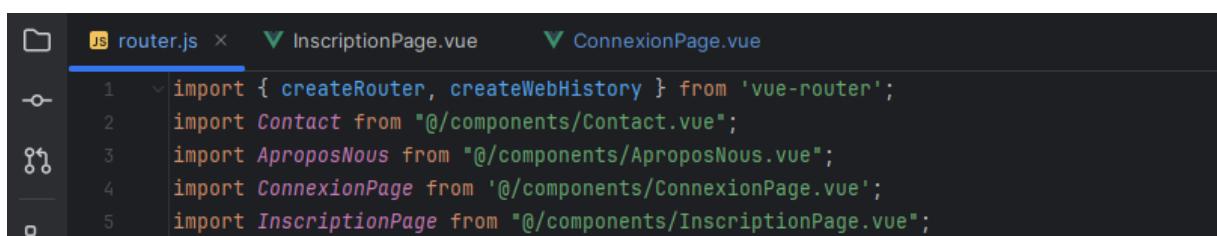
Le fichier router.js définit tous les routes de l'application, chaque route est associée un composant vue.

On l'installe en utilisant la commande « npm install vue-router@4 » dans le fichier de projet.

Après en change le fichier main.js pour qu'il puisse importer l'instance de Vue Router.

```
1
2
3 // Importez `createApp` de Vue
4 import { createApp } from 'vue';
5 import App from './App.vue';
6 import router from './router';
7
8 // Créez l'instance de l'application
9 const app : App<Element> = createApp(App);
10
11 // Utilisez le router avec `app.use()`
12 app.use(router);
13
14 // Montez l'application
15 app.mount( rootContainer: '#app' );
16
```

Et on créer le fichier **router.js** dans le dossier **src** de notre projet :



```
router.js
1 import { createRouter, createWebHistory } from 'vue-router';
2 import Contact from "@/components/Contact.vue";
3 import AproposNous from "@/components/AproposNous.vue";
4 import ConnexionPage from '@/components/ConnexionPage.vue';
5 import InscriptionPage from "@/components/InscriptionPage.vue";
```

En utilisant cette syntaxe on peut importer tous les composants vue.

Et après on peut configurer les routes de notre application dans un tableau **routes []** (**path : /** est la racine).

```
const router : Router = createRouter( options: {
  history: createWebHistory(),
  routes: [
    { path: '/', component: PageAccueil, name: "PageAccueil" },
    { path: '/ConnexionPage', component: ConnexionPage, name: "ConnexionPage" },
    { path: '/InscriptionPage', component: InscriptionPage, name: "InscriptionPage" },
  ]
});
```

Installation d'Axios :

C'est une bibliothèque javascript qui fonctionne comme un client http. Elle permet de communiquer avec des Api en utilisant des requêtes.

Pour l'installation, en utilise la commande « npm install axios ».

Exemple : de l'utilisation avec la méthode Post.

```
  axios.post('http://localhost:8000/api/register-parent', this.user)
    .then(response => {
      alert('Inscription réussie!');
      console.log(response.data);
      this.$router.push('/ConnexionPage');
    })
    .catch(error => {
      console.error(`Erreur lors de l'\`inscription\`:`, error);
      alert(error.response.data.message);
    });
}
```

On utilise catch pour la gestion des erreurs.

➤ **Structure actuelle de notre projet :**

```
✓  └ components
    ✓ AboutUs.vue
    ✓ ActivityList.vue
    ✓ ChangePassword.vue
    ✓ ChildrenSelection.vue
    ✓ ChooseChildren.vue
    ✓ Contact.vue
    ✓ FeaturedOffer.vue
    ✓ Home.vue
    ✓ NavBar.vue
    ✓ NotificationHistory.vue
    ✓ NotificationItem.vue
    ✓ NotificationsPage.vue
    ✓ OfferDetails.vue
    ✓ OffersList.vue
    ✓ OffersPage.vue
    ✓ SignIn.vue
    ✓ SignUp.vue
    ✓ UserProfile.vue
    ✓ UtilisateurParent.vue
    ✓ WelcomePage.vue
    ✓ App.vue
    JS main.js
    JS router.js
    ⚡ .gitignore
```

App.vue :

C'est le fichier principal de l'application qui sert généralement de point d'entrée pour le rendu de l'ensemble de votre application Vue.

Dans la partie Template on a introduit juste la balise de router, parce que c'est l'élément de vue qui va afficher tous les autres composants en fonction de leur route.

On peut ajouter dans **App.vue** un des composants qui sont globaux comme le **navbar**.

```

▼ PageAccueil.vue      ▼ App.vue ×      JS router.js
  1 <template>
  2   <router-view />
  3 </template>
  4 <script>
  5
  6   export default {
  7     name: 'App',
  8     components:{ 
  9       |
 10     }
 11   }
 12 </script>
 13
 14 <style>

```

Page d'accueil :

localhost:8081

Apprenez vos enfants Sans Limites

Commencez dès maintenant, inscrivez vos enfants à des activités enrichissantes, gérez les plannings et les paiements en toute simplicité.

Inscrivez-vous gratuitement Explorez les cours

On a défini le chemin pour la page d'accueil en tant que route par défaut afin qu'elle soit la première page affichée lors de l'accès à l'application.

```
{ path: '/', component: PageAccueil, name: "PageAccueil" },
```

La partie Template du composant **PageAccueil.vue** :

The screenshot shows the template section of the `PageAccueil.vue` component. It includes a `<NavbarElement />`, a container div with classes `conteneurBienvenue` and `sectionTexte`, an `<h1>Apprenez vos enfants Sans Limites</h1>`, descriptive text, and two buttons: `Inscrivez-vous gratuitement` and `Explorez les cours`. An image of a child is also present.

```
1 <template>
2   <NavbarElement />
3   <div class="conteneurBienvenue">
4     <div class="sectionTexte">
5       <h1>Apprenez vos enfants Sans Limites</h1>
6       <p>Commencez dès maintenant, inscrivez vos enfants à des activités enrichissantes, gérez les plannings et les paiements</p>
7       <div>
8         <button type="button" @click="InscriptionMethode" class="boutonInscription">Inscrivez-vous gratuitement</button>
9         <button type="button" @click="ConnexionMethode" class="boutonConnexion">Explorez les cours</button>
10      </div>
11    </div>
12    <div class="sectionImage">
13      
14    </div>
15  </div>
16 </template>
```

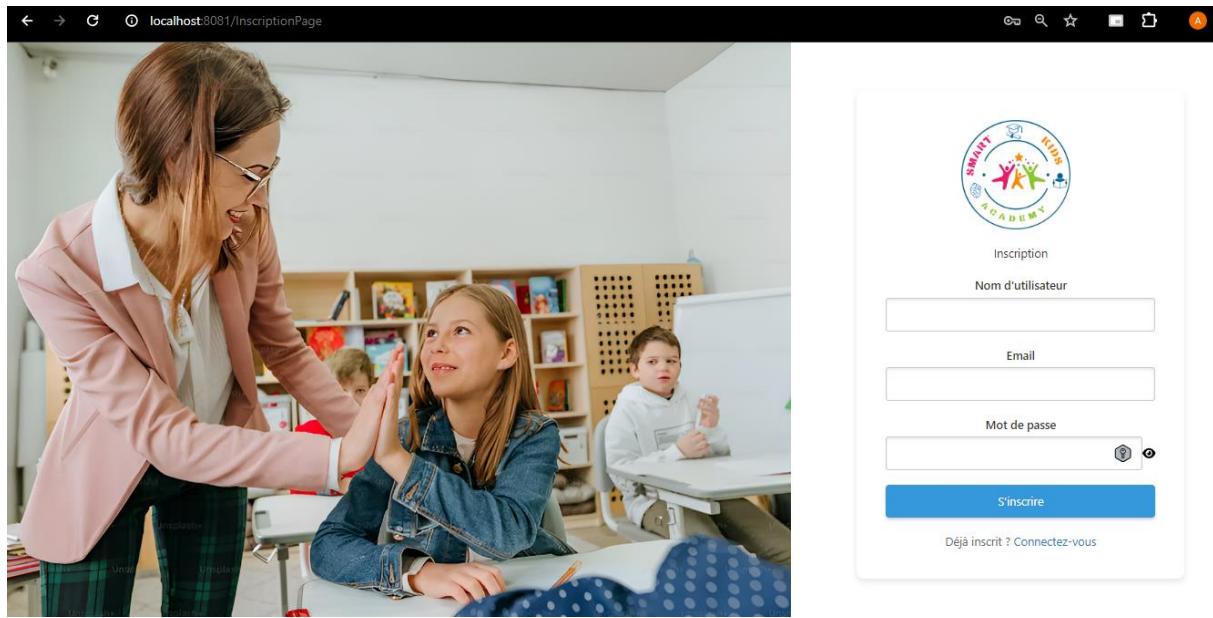
La page d'accueil utilise un template HTML structuré. Elle inclut deux boutons principaux : "Inscrivez-vous gratuitement" et "Explorez les cours". Ces boutons déclenchent, respectivement, les méthodes `InscriptionMethode` et `ConnexionMethode` grâce aux directives `@click`, facilitant ainsi la navigation vers les composants `Inscription` et `Connexion`.

The screenshot shows the script section of the `PageAccueil.vue` component. It imports `NavBarElement`, defines the component name as `'PageAccueil'`, and includes methods `InscriptionMethode` and `ConnexionMethode` that push routes to `/InscriptionPage` and `/ConnexionPage` respectively.

```
18 <script>
19   import NavbarElement from "@/components/NavBar.vue";
20
21   1+ usages  ± Anass El Ouahabi *
22   export default {
23     name:'PageAccueil' ,
24     components: {
25       NavbarElement
26     },
27     methods: {
28       InscriptionMethode() {
29         this.$router.push('/InscriptionPage');
30       },
31       ConnexionMethode() {
32         this.$router.push('/ConnexionPage');
33       }
34     }
35   </script>
36
```

Page d'inscription :

Si l'utilisateur n'est pas encore inscrit sur la plateforme, il sera redirigé vers la page d'inscription en cliquant sur le bouton "Inscrivez-vous gratuitement".



On utilise **@submit.prevent** pour empêcher le rechargement de la page lors de la soumission du formulaire et pour appeler la méthode submitForm.

```
<form @submit.prevent="submitForm">

  <div class="groupeInput">
    <label for="name">Nom d'utilisateur</label>
    <input type="text" id="name" v-model="user.name" required>
  </div>

  <div class="groupeInput">
    <label for="email">Email</label>
    <input type="email" id="email" v-model="user.email" @blur="validerEmail" required>
    <span v-if="emailError" class="erreur">{{ emailError }}</span>
  </div>

  <div class="groupeInput">
    <label for="password">Mot de passe</label>
    <div class="conteneurMotDePasse">
      <input :type="showPassword ? 'text' : 'password'" id="password" v-model="user.password" @blur="validerPassword" required>
      <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
    </div>
    <span v-if="passwordError" class="erreur">{{ passwordError }}</span>
  </div>

  <div class="groupeInput">
    <button type="submit" class="boutonSoumettre">S'inscrire</button>
  </div>
```

La directive `@blur` est utilisée pour valider les champs email et password lorsque l'utilisateur quitte le focus sur ces champ, assurant ainsi une validation en temps réel.

```
62  methods: {
63    validerEmail() {
64      const re = /^[^@\s]+@[^\s]+\.\[^@\s]+$/;
65      if (!re.test(this.user.email)) {
66        this.emailError = 'Veuillez entrer un email valide';
67      } else {
68        this.emailError = '';
69      }
70    },
71    validerPassword() {
72      if (this.user.password.length < 8) {
73        this.passwordError = 'Le mot de passe doit contenir au moins 8 caractères';
74      } else {
75        this.passwordError = '';
76      }
77    },
78  },
```

Inscription

Nom d'utilisateur

Email

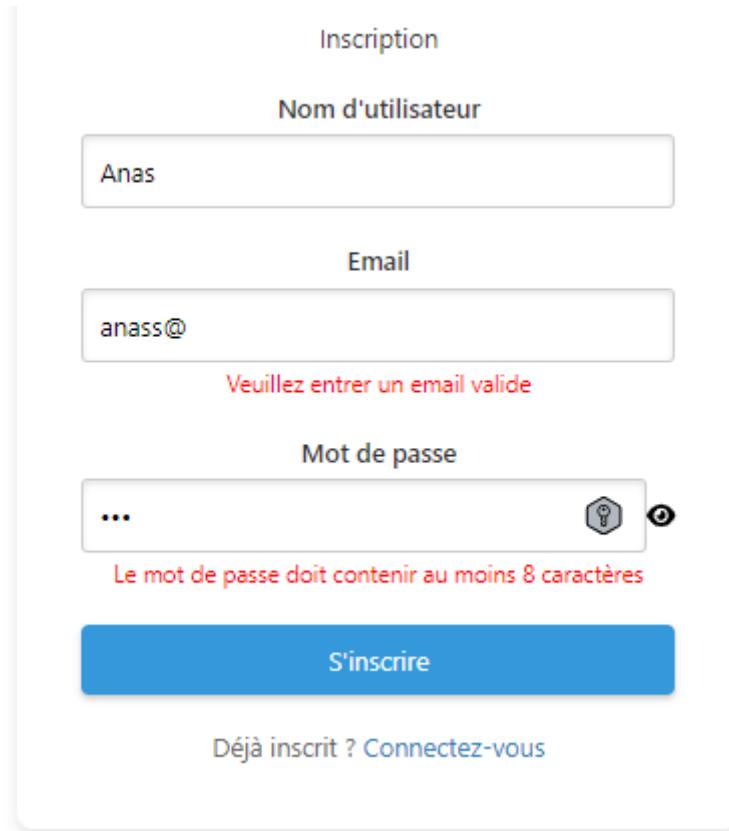
Veuillez entrer un email valide

Mot de passe

Le mot de passe doit contenir au moins 8 caractères

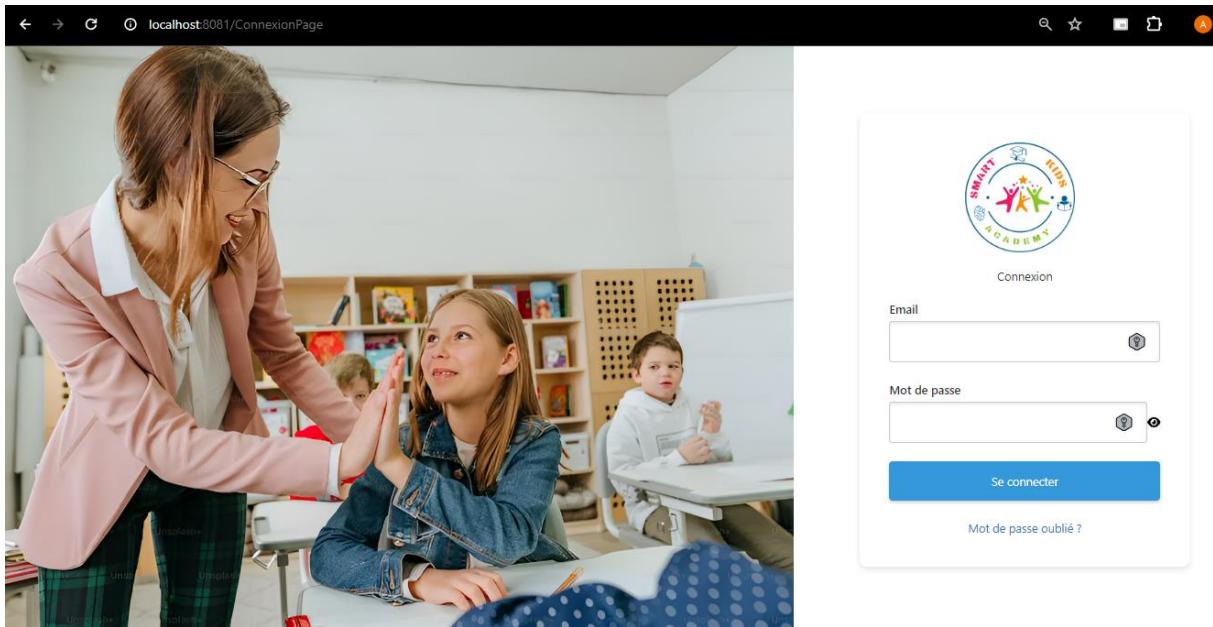
[S'inscrire](#)

Déjà inscrit ? [Connectez-vous](#)



Si l'utilisateur est déjà inscrit, il sera redirigé vers la page de Connexion.

Page de connexion :



Nous importons Axios et configurons les options pour inclure les identifiants et les jetons CSRF dans chaque requête HTTP. La fonction `getCSRFToken` est utilisée pour obtenir le jeton CSRF depuis le serveur.

Cette configuration est essentielle pour sécuriser les requêtes HTTP en protégeant contre les attaques CSRF.

```
36 <script>
37   import axios from '@/axios';
38   axios.defaults.withCredentials = true;
39   axios.defaults.withXSRFToken = true;
40
41   1+ usages  ↗ Anass El Ouahabi
42   async function getCSRFToken() {
43     try {
44       await axios.get(url: 'http://localhost:8000/sanctum/csrf-cookie');
45     } catch (error) {
46       console.error('Failed to fetch CSRF token:', error);
47     }
48   }

```

On ajoute une requête POST qui est envoyée à l'API de connexion avec les données de l'utilisateur.

Le jeton d'authentification et le rôle de l'utilisateur (parent, admin ou animateur) sont stockés dans le `localStorage`, et le jeton est ajouté aux en-têtes des requêtes Axios pour les futures requêtes authentifiées.

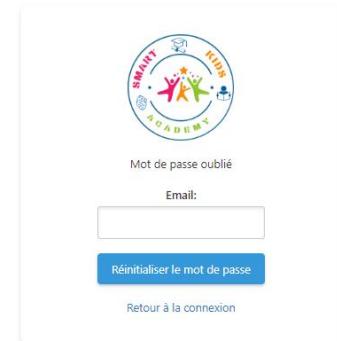
```

        ,
        submitForm() {
            this.validerEmail();
            this.validerPassword();
            if (this.emailError || this.passwordError) {
                return;
            }
            getCSRFToken().then(()=>{
                axios.post( url: 'http://localhost:8000/api/login' , this.user).then(response => {
                    const token = response.data.token;
                    const role = response.data.role ;
                    localStorage.setItem('user_role', role) ;
                    localStorage.setItem('auth_token', response.data.token);
                    axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
                    alert('Connexion réussie!');
                    this.$router.push('/offerspage');
                })
                .catch(error => {
                    console.error('Erreur de connexion:', error);
                    alert(error.response.data.message);
                });
            })
        }
    }
}

```

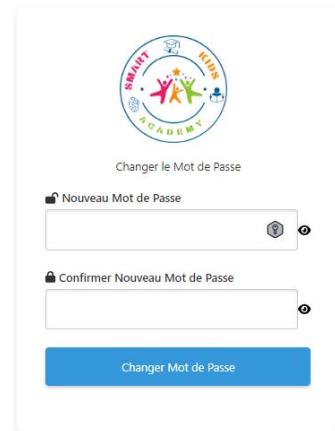
En cas de succès, l'utilisateur est redirigé vers la page des offres.

- L'utilisateur peut récupérer son mot de passe en cas d'oubli.

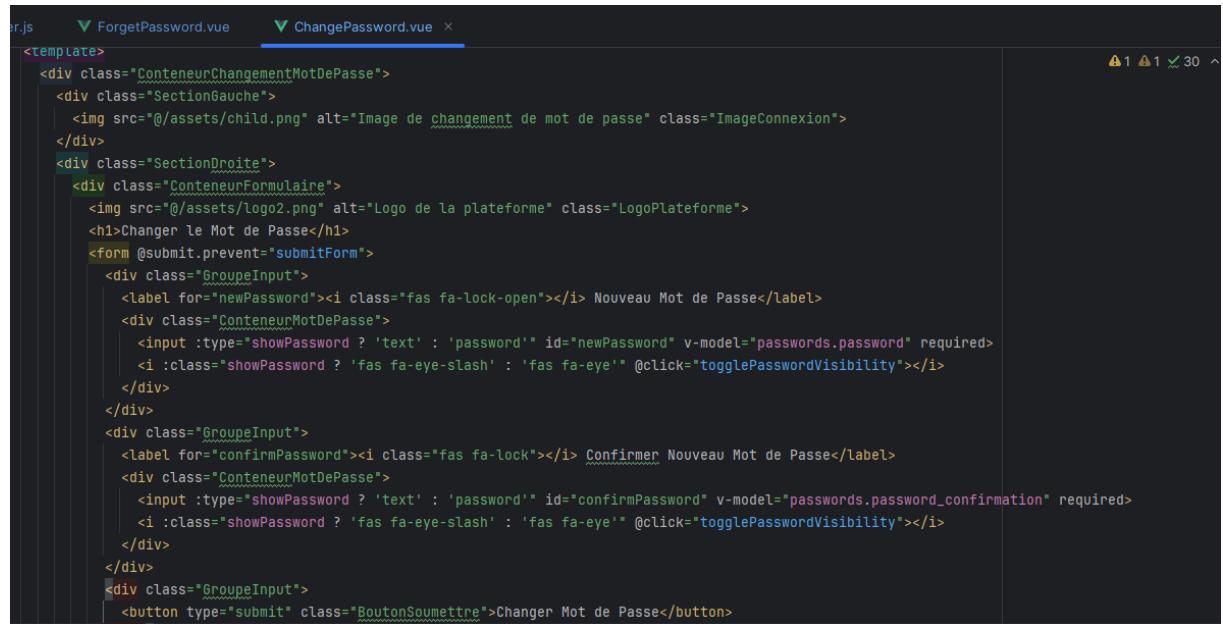


Il faut écrire l'adresse email, et un email sera envoyé à l'adresse email donné.

Après avoir cliqué sur le lien envoyé via email, l'utilisateur sera redirigé vers la page Changepassword.



Lorsque l'utilisateur clique sur « Changer Mot de Passe », il sera redirigé vers la page de connexion afin de se connecter avec son nouveau mot de passe.

A screenshot of a code editor showing the template for the ChangePassword.vue component. The template includes sections for left and right sections, a logo, a title, and a form with two password inputs and a submit button.

```
<template>
<div class="ConteneurChangementMotDePasse">
  <div class="SectionGauche">
    | 
  </div>
  <div class="SectionDroite">
    <div class="ConteneurFormulaire">
      
      <h1>Changer le Mot de Passe</h1>
      <form @submit.prevent="submitForm">
        <div class="GroupeInput">
          <label for="newPassword"><i class="fas fa-lock-open"></i> Nouveau Mot de Passe</label>
          <div class="ConteneurMotDePasse">
            <input :type="showPassword ? 'text' : 'password'" id="newPassword" v-model="passwords.password" required>
            <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
          </div>
        </div>
        <div class="GroupeInput">
          <label for="confirmPassword"><i class="fas fa-lock"></i> Confirmer Nouveau Mot de Passe</label>
          <div class="ConteneurMotDePasse">
            <input :type="showPassword ? 'text' : 'password'" id="confirmPassword" v-model="passwords.password_confirmation" required>
            <i :class="showPassword ? 'fas fa-eye-slash' : 'fas fa-eye'" @click="togglePasswordVisibility"></i>
          </div>
        </div>
        <div class="GroupeInput">
          <button type="submit" class="BoutonSoumettre">Changer Mot de Passe</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Dans la méthode **mounted**, on a extrait le jeton de réinitialisation de mot de passe depuis l'URL actuelle à l'aide de `window.location.href`. Après on a utiliser `substring` et `lastIndexOf` pour obtenir la partie de l'URL après le dernier '/', puis stockons le jeton dans `this.token`. Cela garantit la disponibilité du jeton lors de la soumission du formulaire de changement de mot de passe.

Lors de la soumission du formulaire, la méthode `submitForm` appelle la fonction `changePassword` en passant le jeton extrait de l'URL. La fonction `changePassword` vérifie d'abord si le jeton est présent. Ensuite, elle compare les mots de passe pour s'assurer qu'ils correspondent. Après une requête POST est envoyée à l'API avec le jeton et les nouveaux mots de passe.

```

  ForgetPassword.vue      ChangePassword.vue ×
  mounted() {
    const url = window.location.href;
    const token = url.substring(url.lastIndexOf('/') + 1);
    this.token = token;
  },
  methods: {
    togglePasswordVisibility() {
      this.showPassword = !this.showPassword;
    },
    submitForm() {
      this.changePassword(this.token);
    },
    changePassword(token) {
      if (!token) {
        alert("Token non trouvé dans l'URL.");
        return;
      }
      if (this.passwords.password !== this.passwords.password_confirmation) {
        alert("Les mots de passe ne correspondent pas.");
        return;
      }
      axios.post(`http://localhost:8000/api/reset-password/${token}`, this.passwords)
        .then(response => {
          alert(response.data.message);
          this.$router.push('/ConnexionPage');
        })
    }
  }
}

```

Navbar :



On a ajouter dans le code une classe afficher qui depend de l'état de menuOuvert. Si menuOuvert est vrai, la classe afficher est appliquée, ce qui permettant d'afficher le menu sur les petits écrans.

Et pour assurer une navigation responsive, on a utilisé une media query pour cibler les écrans de largeur maximale de 768px.

JS router.js

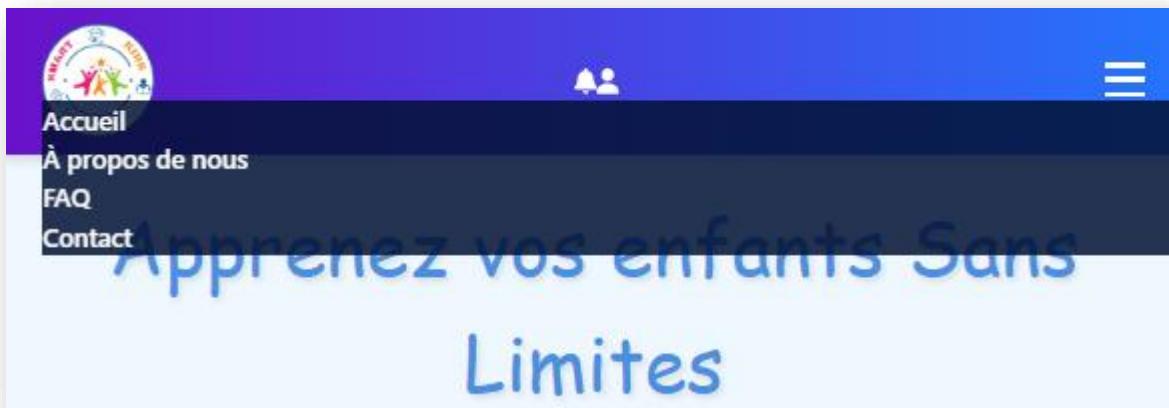
NavBar.vue

```

1 <template>
2   <nav class="BarreDeNavigation">
3     <div class="NavigationGauche">
4       <router-link to="/" class="MarqueNavigation">
5         
6       </router-link>
7     </div>
8     <ul class="NavigationCentre" :class="{ 'afficher': menuOuvert }">
9       <li><router-link to="/">Accueil</router-link></li>
10      <li><router-link to="/AproposNous">À propos de nous</router-link></li>
11      <li><router-link to="/how-to-use">FAQ</router-link></li>
12      <li><router-link to="/contact">Contact</router-link></li>
13    </ul>
14    <div class="NavigationDroite">
15      <router-link to="/notification"><i class="fas fa-bell"></i></router-link>
16      <router-link to="/profile"><i class="fas fa-user"></i></router-link>
17    </div>
18    <button class="ToggleNavigation" @click="basculerMenu">
19      <span class="Barre"></span>
20      <span class="Barre"></span>
21      <span class="Barre"></span>
22    </button>
23  </nav>
24 </template>

```

La Navbar pour les écrans de moins de 768px sera affichée comme suit :



```

    @media (max-width: 768px) {
      .NavigationGauche::after {
        display: none;
      }

      .NavigationCentre {
        position: absolute;
        top: 60px;
        width: 100%;
        background-color: rgba(1, 17, 52, 0.85);
        flex-direction: column;
        display: none;
      }

      .NavigationCentre.afficher {
        display: flex;
      }

      .ToggleNavigation {
        display: flex;
      }
    }
  
```

• Gestion des permission :

On a mis en place une gestion des permissions pour sécuriser l'accès aux différentes routes de l'application. Chaque route est configurée avec une propriété meta indiquant si l'authentification est requise (requiresAuth) et les rôles autorisés à y accéder (roles). Par exemple, la route /offerspage est accessible uniquement aux utilisateurs ayant le rôle de parent. Lors de la connexion, on stocke le rôle de l'utilisateur et le jeton d'authentification dans le localStorage. Ensuite, on ajoute ce jeton aux en-têtes des requêtes Axios pour authentifier les futures requêtes. Cette configuration permet de contrôler efficacement l'accès aux différentes sections de l'application en fonction des rôles des utilisateurs.

```

  { path: '/', component: PageAccueil, name: "PageAccueil" },
  { path: '/ConnexionPage', component: ConnexionPage, name: "ConnexionPage" },
  { path: '/InscriptionPage', component: IncriptionPage, name: "InscriptionPage" },
  { path: '/forgetpassword', component: ForgetPassword , name:"forgetpassword"}, ,
  { path: '/changePassword/:token' , component:ChangePassword , name:"changePassword"}, ,
  { path: '/offerspage' , component:OffersPage , name:"OffersPage" , meta: { requiresAuth: true , roles: ['parent'] } },
  { path: '/offerdetails/:id' , component:OfferDetails , name:"offerdetails" , meta: { requiresAuth: true , roles: ['parent'] }},
  { path: '/activitylist/:offerId' , component:ActivityList , name:"activitylist" , meta: { requiresAuth: true , roles: ['parent'] } },
  { path: '/choosechildren' , component:ChooseChildren , name:"choosechildren" , meta: { requiresAuth: true , roles: [ 'parent' ] } },
  { path: '/selectschedule/:activityId' , component:SelectSchedule , name:"selectschedule" , meta: { requiresAuth: true , roles: [ 'parent' ] } },
  { path: '/submitrequest' , component:SubmitRequest , name:"submitrequest" , meta: { requiresAuth: true , roles: [ 'parent' ] } },
  { path: '/notificationpage' , component:NotificationsPage , name:"notificationpage" , meta: { requiresAuth: true , roles: [ 'parent' , 'admin'] } },
  { path: '/notificationhistory' , component:NotificationHistory , name:"notificationhistory" , meta: { requiresAuth: true , roles: [ 'parent' , 'admin'] } }

```

```

getCSRFToken().then(()=>{

  axios.post( url: 'http://localhost:8000/api/login' , this.user).then(response => {

    const token = response.data.token;
    const role = response.data.role ;
    localStorage.setItem('user_role', role) ;
    localStorage.setItem('auth_token', response.data.token);
    axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
    alert('Connexion réussie!');
    this.$router.push('/offerspage');
  })
  .catch(error => {
    console.error('Erreur de connexion:', error);
    alert(error.response.data.message);
  });
});

```

Pour renforcer la sécurité, on utilise le guard `beforeEach` de Vue Router pour vérifier les permissions avant chaque navigation. On récupère le token JWT et le rôle de l'utilisateur depuis le `localStorage`. Si la route requiert une authentification (`requiresAuth`), on vérifie la présence du token. Si le token est absent, on redirige l'utilisateur vers la page de connexion. Ensuite, on vérifie si l'utilisateur possède les permissions nécessaires pour accéder à la route en comparant son rôle avec les rôles autorisés définis dans les métadonnées de la route. Si l'utilisateur n'a pas les permissions requises, on le redirige vers une page "non autorisée". Ce mécanisme garantit que seules les personnes autorisées peuvent accéder à des sections spécifiques de l'application.

```

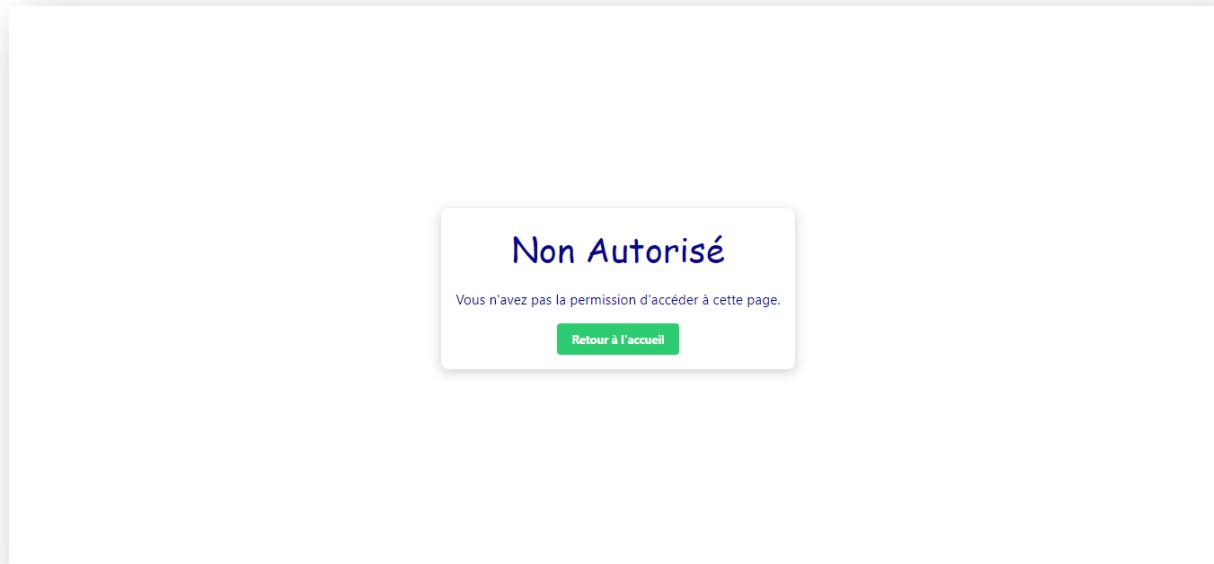
▲ Anass El Ouahabi
router.beforeEach( guard: to : RouteLocationNormalized , from : RouteLocationNormalized , next : NavigationGuardNext ) : void => {
  const token : string  = localStorage.getItem( key: 'auth_token'); // Récupérer le token JWT stocké
  const role : string  = localStorage.getItem( key: 'user_role'); // Récupérer le rôle de l'utilisateur stocké

  // On Vérifie si la route nécessite une authentification
  if (to.matched.some(record : RouteRecordNormalized  => record.meta.requiresAuth)) {

    if (!token) {
      next('/ConnexionPage');
    } else {
      // Vérifiant si l'utilisateur a les permissions nécessaires
      if (to.matched.some(record : RouteRecordNormalized  => record.meta.roles && !record.meta.roles.includes(role))) {
        next('/unauthorized');
      } else {
        next();
      }
    }
  } else {
    next();
  }
};

```

La page « non- autorisée » s'affiche comme ca :



I. Les interfaces

1. Interface de parent

Page principale :

The screenshot shows the main dashboard for parents. At the top, there's a blue header with navigation links: 'BIENVENUE', 'Accueil', 'À propos de nous', 'Contact', 'FAQ', and a user profile icon. The main content area has a white background. On the left, a large card for a 'Robotique avancée' course is displayed. It includes a photo of two students working on a robot, a discount offer of '35.00% de remise', and details about the start and end dates. A blue button says 'Inscrivez-vous à l'offre'. On the right, there's a section titled 'Toutes les Offres' with a search bar and several thumbnail images representing other course offerings.

C'est la page principale qui sera affichée à l'utilisateur lorsqu'il se connecte à la plateforme. Cette page est composée de trois composants qui constituent l'ensemble de l'affichage : FeaturedOffer, OffersList, et ParentUser.

- **featuredoffer :**

Il affiche les trois offres de la plateforme qui ont la remise la plus grande. Cela est utile pour attirer l'attention des utilisateurs sur les meilleures offres disponibles et inciter à profiter des remises les plus avantageuses.

. On a récupéré les données des offres via une requête API :

```
methods: {
  fetchOffers() {
    axios.get('http://localhost:8000/api/show/offers/top/')
      .then(response => {
        this.offers = response.data;
      })
      .catch(error => {
        console.error('Erreur lors de la récupération des offres populaires:', error);
      });
  },
}
```

Ce composant affiche automatiquement les trois offres ayant les plus grandes remises sur la plateforme. Chaque offre est affichée pendant une période de 3 seconds, Ce cycle continue indéfiniment, offrant ainsi une visibilité constante des meilleures offres.



- Apres 3 seconds il affiche cette offre :



```
mounted() {
  this.intervalId = setInterval(this.nextOffer, timeout: 3000);
},
beforeUnmount() {
  clearInterval(this.intervalId);
},
```

On a ajouté la méthode mounted() pour que, chaque fois que le composant est monté, un intervalle soit défini pour appeler la méthode nextOffer toutes les 3 secondes.

La méthode beforeUnmount est utilisée pour effacer l'intervalle défini afin d'éviter les appels après le démontage.

```

1 <template>
2   <div class="offer-container" v-if="offers.length"
3     <button @click="prevOffer" class="nav-btn left-btn"><i class="fas fa-chevron-left"></i></button>
4     <transition name="fade" mode="out-in">
5       <div class="offer-content" :key="currentOffer.id">
6         <div class="text-section">
7           <h2>{{ currentOffer.titre }}</h2>
8           <p class="description">{{ currentOffer.description }}</p>
9           <h3 class="remise">{{ currentOffer.remise }}% de remise</h3>
10          <router-link :to="{ name: 'offerdetails', params: { id: currentOffer.id } }" class="btn">
11            Inscrivez-vous à l'offre
12          </router-link>
13        </div>
14        <div class="date-section">
15          <div class="date">
16            <span class="date-text">Date de début : </span> {{ formatDate(currentOffer.date_debut) }}
17          </div>
18          <div class="date">
19            <span class="date-text">Date de fin : </span> {{ formatDate(currentOffer.date_fin) }}
20          </div>
21        </div>
22        <div class="image-section">
23          
24        </div>
25      </div>
26    </transition>
27    <button @click="nextOffer" class="nav-btn right-btn"><i class="fas fa-chevron-right"></i></button>

```

script > methods > formatDate()

Les utilisateurs peuvent également naviguer manuellement entre les offres à l'aide des boutons de navigation qui appellent les méthodes **nextOffer** et **prevOffer**.

```

nextOffer() {
  this.currentIndex = (this.currentIndex + 1) % this.offers.length;
},
prevOffer() {
  this.currentIndex = (this.currentIndex + this.offers.length - 1) % this.offers.length;
},

```

Et afin de créer une expérience utilisateur fluide et amusante , j'ai ajouté de transitions animée (transition **fade**) entre les offres , incitant les utilisateur a profiter des meilleures remises disponibles.

- **Offerlist :**

Ce composant affiche une liste de toutes les offres disponibles sur la plateforme. Il contient également une barre de recherche qui permet aux utilisateurs de rechercher des offres par titre ou description.

Toutes les Offres

Calcul Différentiel
Cours de calcul différentiel pour les lycéens....

[Voir Détails](#)

Intelligence Artificielle
Cours d'initiation à l'intelligence artificielle et au machine learning....

[Voir Détails](#)

Sécurité Informatique
Introduction à la sécurité informatique et aux techniques de protection des données....

[Voir Détails](#)

Robotique avancée
Cours de robotique avancée pour les adolescents....

[Voir Détails](#)

Impression 3D
Apprentissage de l'impression 3D et du design....

[Voir Détails](#)

Technologie verte
Introduction aux technologies vertes et à l'écologie....

[Voir Détails](#)

Sciences de la Terre
Étude des sciences de la Terre et de la géologie....

[Voir Détails](#)

Cartes et stratégies
Apprentissage des jeux de cartes et des stratégies....

[Voir Détails](#)

[Précédent](#) Page 1 sur 3 [Suivant](#)

- On peut simplement écrire le titre de l'offre souhaitée dans la barre de recherche :

http://localhost:8081/offerspage

Inscrivez-vous à l'offre

Date de début : 2024-06-07
Date de fin : 2024-12-14



Toutes les Offres

Programmation entre 15 et 17 ans
Cours de programmation intermédiaire pour les adolescents de 15 à 17 ans....

[Voir Détails](#)

[Précédent](#) Page 1 sur 3 [Suivant](#)

- Ou en peut rechercher juste avec la description si on ne connaît pas le titre de l'offre , ce qui facilite l'expérience de l'utilisateur :

Toutes les Offres

▼



Robotique avancée

Cours de robotique avancée pour les adolescents...

[Voir Détails](#)

Précédent
Page 1 sur 3
Suivant

La Template de cette composant :

```
<template>
<div class="offers-list">
  <h3>Toutes les Offres</h3>
  <div class="search-bar">
    <i class="fas fa-search search-icon"></i>
    <input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
    <button @click="toggleFilter" class="filter-btn">
      <i class="fas fa-filter"></i>
    </button>
    <div v-if="showFilter" class="filter-dropdown">
      <label for="domaine">Filtrer par domaine :</label>
      <select id="domaine" v-model="selectedDomain" @change="filterByDomain">
        <option value="">Tous les domaines</option>
        <option value="informatique">Informatique</option>
        <option value="sciences">Sciences</option>
        <option value="Mathématiques">Mathématiques</option>
        <option value="Jeux">Jeux</option>
      </select>
    </div>
  </div>
  <div v-if="loading" class="loader">Chargement des offres...</div>
  <div v-else>
    <div v-if="error" class="error-message">
      <p>Une erreur s'est produite lors du chargement des offres : {{ error }}</p>
      <button @click="fetchOffers">Réessayer</button>
    </div>
    <div v-else class="offers-grid">
```

On a ajouté Le champ de recherche par :

```
<input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
```

Ce champ de saisie est lié à la variable **searchQuery** via v-model. Chaque fois que l'utilisateur tape dans ce champ, la méthode **searchOffers** est appelée grâce à l'événement @input.

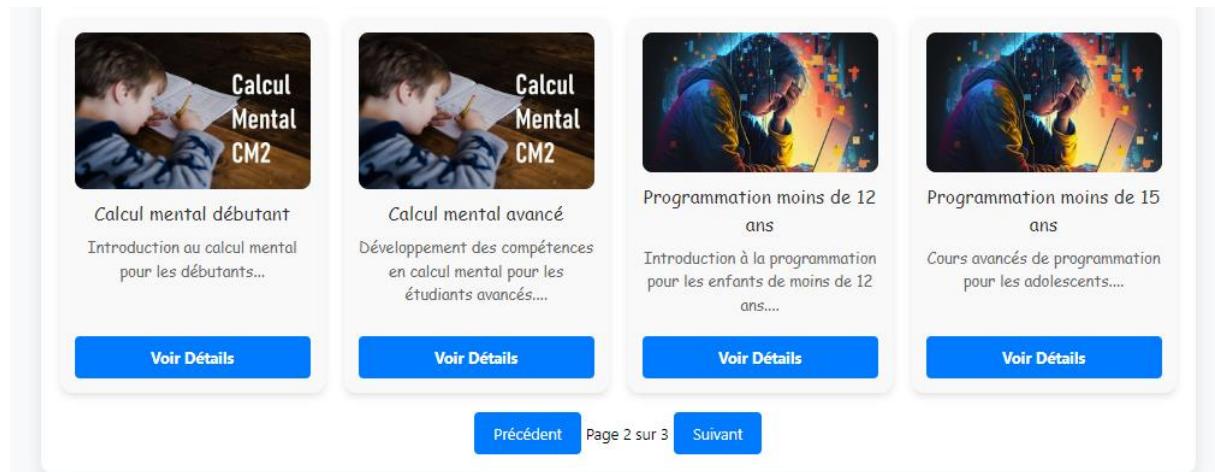
```
    }
    searchOffers() {
      this.currentPage = 1;
    },
  
```

Cette méthode est appelée à chaque fois que l'utilisateur tape dans la barre de recherche. Afin de réinitialisé la page actuelle à 1 pour afficher les résultats depuis le début.

```
65    computed: {
66      filteredOffers() {
67        const searchLower = this.searchQuery.toLowerCase();
68        return this.offers
69          .filter(offer =>
70            (offer.titre.toLowerCase().includes(searchLower) || offer.description.toLowerCase().includes(searchLower)) &&
71            (this.selectedDomain ? offer.domaine.toLowerCase() === this.selectedDomain.toLowerCase() : true)
72          )
73          .slice(this.startIndex, this.endIndex);
74    },
75  },
```

Cette propriété calculée filtre les offres en fonction de la requête de recherche. Elle convertit la requête en minuscule et vérifie si le titre ou la description de l'offre contient la chaîne de recherche. Les résultats filtrés sont ensuite paginés, ce veut dire que les résultats filtrés sont divisés en pages afin que l'utilisateur puisse les parcourir plus facilement.

Comme ça :



On a ajouté une obtient de filtrer les offres par domaine. L'utilisateur peuvent maintenant sélectionner un domaine spécifique dans un menu déroulant, et seules les offres correspondant à ce domaine seront affichées.

The screenshot shows the 'Toutes les Offres' page with the following elements:

- A header with a search bar containing "Rechercher des offres..." and a dropdown labeled "Filtrer par domaine : Mathématiques".
- A main section titled "Toutes les Offres" displaying three course offerings:

 - Calcul Différentiel**
Cours de calcul différentiel pour les lycéens....
[Voir Détails](#)
 - Calcul Mental CM2**
Calcul mental débutant
Introduction au calcul mental pour les débutants...
[Voir Détails](#)
 - Calcul Mental CM2**
Calcul mental avancé
Développement des compétences en calcul mental pour les étudiants avancés....
[Voir Détails](#)

- Navigation buttons at the bottom: Précédent, Page 1 sur 3, and Suivant.

```

<div class="search-bar">
  <i class="fas fa-search search-icon"></i>
  <input type="text" v-model="searchQuery" placeholder="Rechercher des offres..." @input="searchOffers" />
  <button @click="toggleFilter" class="filter-btn">
    <i class="fas fa-filter"></i>
  </button>
<div v-if="showFilter" class="filter-dropdown">
  <label for="domaine">Filtrer par domaine :</label>
  <select id="domaine" v-model="selectedDomain" @change="filterByDomain">
    <option value="">Tous les domaines</option>
    <option value="informatique">Informatique</option>
    <option value="sciences">Sciences</option>
    <option value="Mathématiques">Mathématiques</option>
    <option value="Jeux">Jeux</option>
  </select>

```

Ce code affiche un bouton avec une icône de filtre. Lorsqu'il est cliqué, il appelle la méthode toggleFilter pour afficher ou masquer le menu déroulant de filtre.



Le sélecteur dans la template est lié à la variable selectedDomain et appelle la méthode filterByDomain chaque fois qu'un domaine est sélectionné.

La méthode filterByDomain sert à réinitialiser la page actuelle à 1 pour afficher les résultats depuis le début.

- On a ajouté aussi le cas de chargement des offres, et le cas erreur lors de récupération des offres depuis l'api .

```

21   <div v-if="loading" class="loader">Chargement des offres...</div>
22   <div v-else>
23     <div v-if="error" class="error-message">
24       <p>Une erreur s'est produite lors du chargement des offres : {{ error }}</p>
25       <button @click="fetchOffers">Réessayer</button>
26     </div>

```

- Le cas de chargement des offres :

le message de chargement s'affiche lorsque la variable **loading** est vraie. Cela indique que les données sont en cours de récupération.

The screenshot shows the application's main page titled "Toutes les Offres". At the top, there is a navigation bar with links for "BIENVENUE", "Accueil", "À propos de nous", "Contact", and "FAQ". On the right side of the header, there are icons for notifications and user profile. Below the header, a search bar contains the placeholder "Rechercher des offres...". Underneath the search bar, a message says "Chargement des offres..." followed by a small loading icon.

- Le cas d'erreur :

Le message d'erreur s'affiche lorsque la variable **error** contient une valeur. Il décrit le problème survenu, et le bouton "Réessayer" permet à l'utilisateur de tenter de nouveau de charger les offres en appelant la méthode `fetchOffers`.

```
created() {
  this.fetchOffers();
},
methods: {
  fetchOffers() {
    this.loading = true;
    this.error = null;
    axios.get(url: 'http://localhost:8000/api/show/offers')
      .then(response => {
        this.offers = response.data;
        this.loading = false;
      })
      .catch(error => {
        console.error('There was an error fetching the offers:', error);
        this.error = 'Impossible de charger les offres. Veuillez réessayer plus tard.';
        this.loading = false;
      });
  },
}
```

The screenshot shows the same "Toutes les Offres" page as before, but now it displays an error message. The loading message "Chargement des offres..." has been replaced by a red error message: "Une erreur s'est produite lors du chargement des offres : Impossible de charger les offres. Veuillez réessayer plus tard." Below this message is a red button labeled "Réessayer". At the bottom of the page, there are navigation buttons for "Précédent", "Page 1 sur 0", and "Suivant".

- **UtilisateurParent :**

Le composant UtilisateurParent combine une navigation supérieure et une navigation latérale pour offrir une expérience utilisateur complète et intuitive.

- Navigation supérieure :



La navigation supérieure inclut un bouton pour basculer la barre latérale (c'est le Button lié à bienvenu), des liens de navigation vers les pages principales, et une icône de notification qui dirige l'utilisateur vers la page de notification et Une icône de profil qui permet d'accéder à un menu déroulant avec des options pour gérer le profil utilisateur et pour se déconnecter.

The screenshot shows a landing page for a robotics course. At the top is a blue header with 'BIENVENUE', 'Accueil', 'À propos de nous', 'Contact', 'FAQ', and user icons. Below the header, the main content area features a course offer for 'Robotique avancée'. It includes a discount message '35.00% de remise', a call-to-action button 'Inscrivez-vous à l'offre', and details about the course start date (2024-06-07) and end date (2024-12-14). To the right is a photograph of two children working on a robotic vehicle. A sidebar on the right contains a dropdown menu with options: 'Profil', 'Mes Enfants', 'Changer mot de passe', 'Mes demandes', and 'Déconnexion'.

```
js router.js  ▼ UtilisateurParent.vue ×
1 <template>
2   <!-- Navigation Supérieure -->
3   <nav class="navbar">
4     <div>
5       <button @click="toggleSidebar" class="navbar-button">
6         <i class="fas fa-bars"></i> BIENVENUE
7       </button>
8     </div>
9     <div class="navbar-center">
10      <router-link to="/offerspage" class="nav-link">Accueil</router-link>
11      <router-link to="/AproposNous" class="nav-link">À propos de nous</router-link>
12      <router-link to="/contact" class="nav-link">Contact</router-link>
13      <router-link to="/FAQ" class="nav-link">FAQ</router-link>
14    </div>
15    <div class="navbar-right">
16      <router-link to="/notificationpage" class="icon-link"><i class="fas fa-bell"></i></router-link>
17      <div class="profile-dropdown">
18        <button @click="toggleProfileMenu" class="icon-link profile-button"><i class="fas fa-user"></i></button>
19        <div v-if="showProfileMenu" class="dropdown-menu">
20          <router-link to="/userprofile">Profil</router-link>
21          <router-link to="/userchildren">Mes Enfants</router-link>
22          <router-link to="/changepassword/${token}">Changer mot de passe</router-link>
23          <router-link to="/parentrequests">Mes demandes</router-link>
24          <button @click="logout">Déconnexion</button>
25        </div>
26      </div>
27    </div>

```

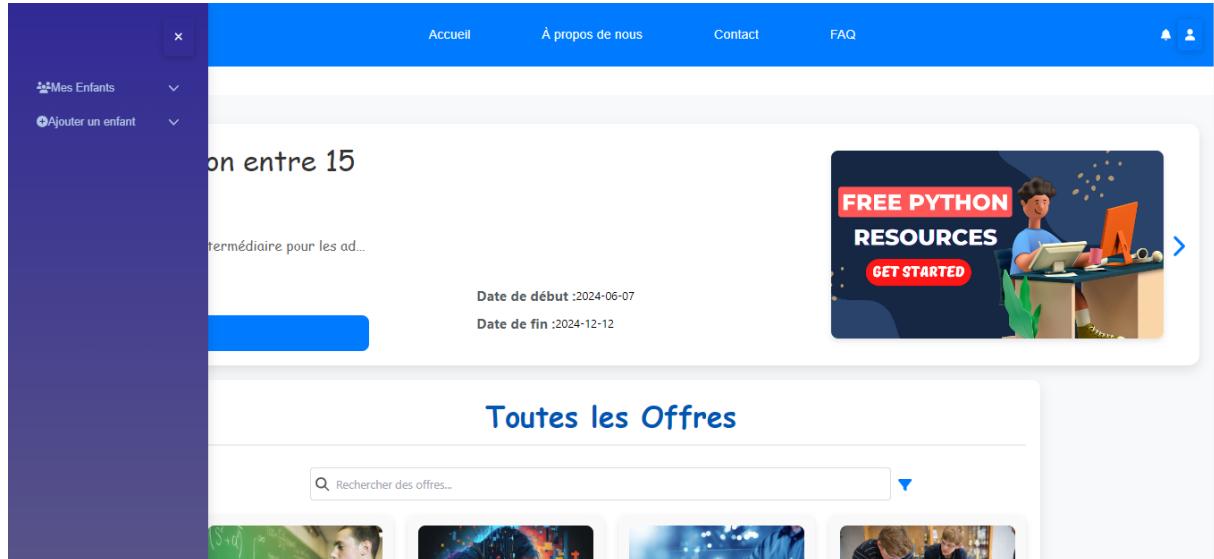
La méthode **logout** est utilisée pour déconnecter l'utilisateur et rediriger vers la page de connexion, avec suppression du token pour sécuriser la déconnexion.

```
146     logout() {
147         localStorage.removeItem(key: 'token');
148         this.$router.push('/ConnexionPage');
149     },

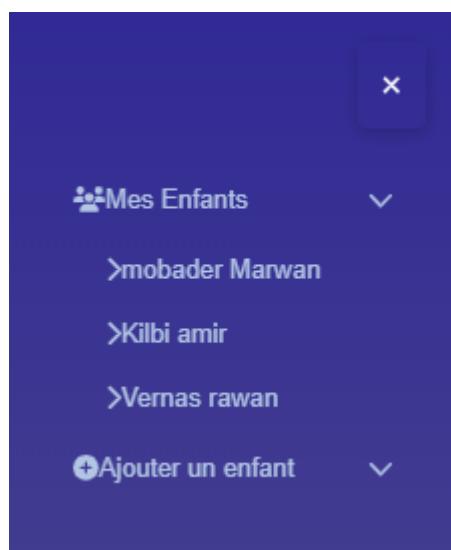
```

- Navigation latérale :

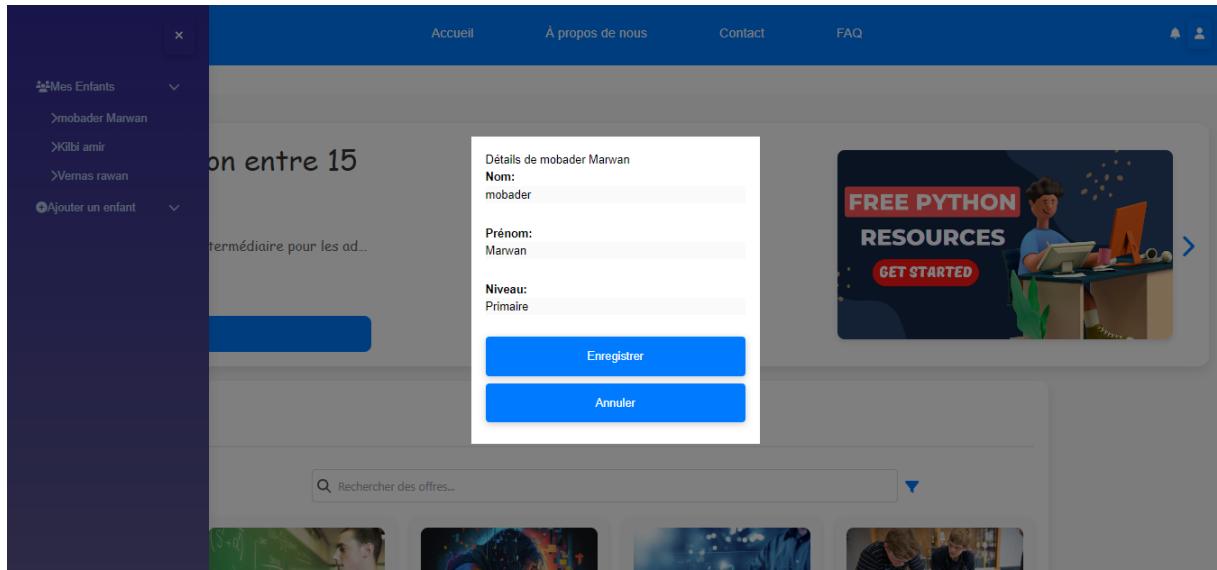
La navigation latérale offre des options pour gérer les enfants de l'utilisateur. Il peuvent ajouter et afficher les enfant de l'utilisateur parent.



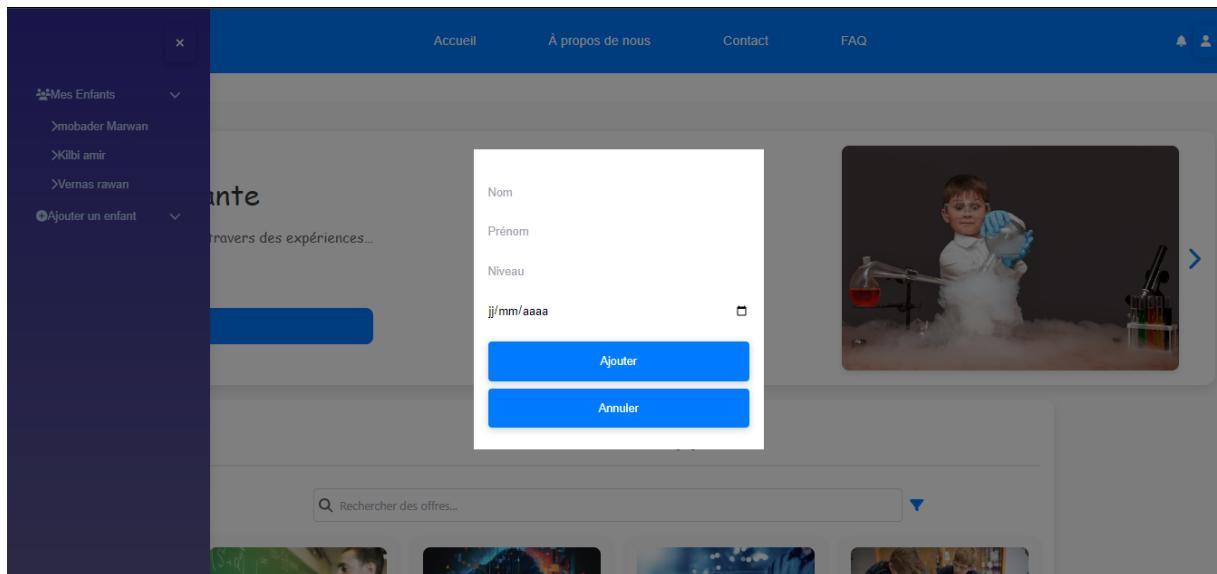
Les enfants de l'utilisateur sont afficher lorsque il clique sur Mes enfants



En cliquant sur un enfant en va recuperer les details de cette enfant :



Et pour ajouter un enfant il suffit de remplir la formulaire :



le code de navigation latérale :

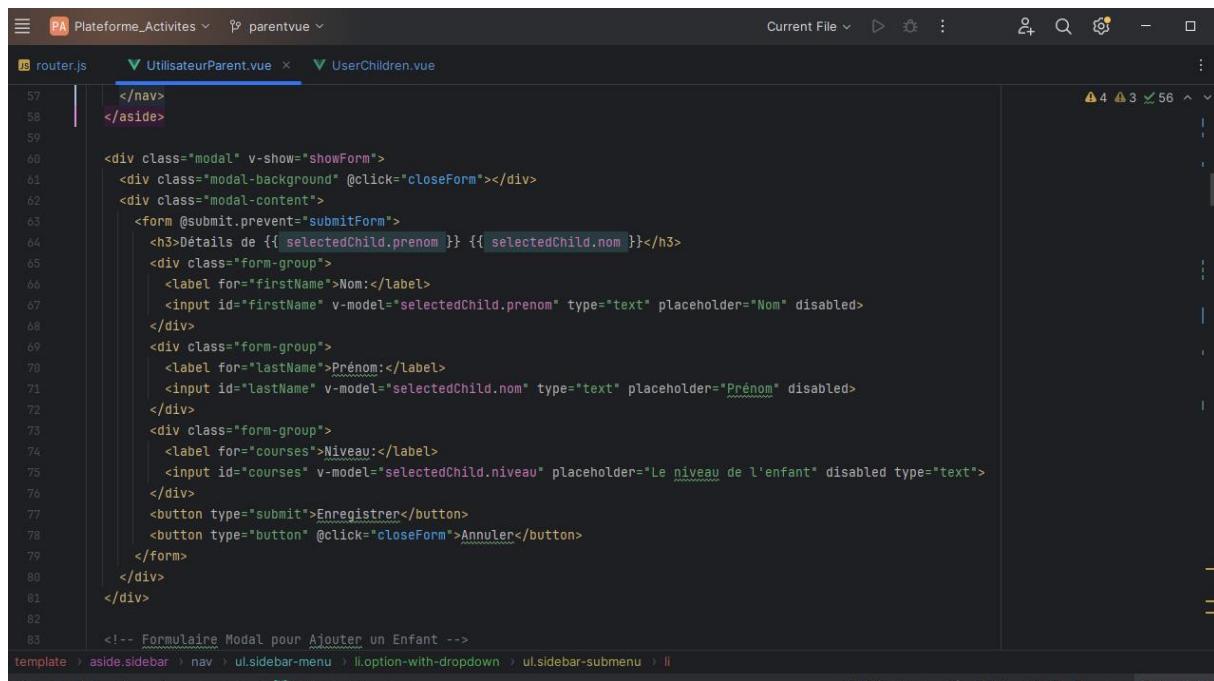
```

30      <!-- Navigation Latérale -->
31      <aside :class="['closed': !isSidebarOpen]" class="sidebar">
32          <div class="sidebar-close">
33              <button @click="closeSidebar" class="sidebar-close-button">
34                  <i class="fas fa-times"></i>
35              </button>
36          </div>
37          <nav>
38              <ul class="sidebar-menu">
39                  <li v-for="(option, index) in opciones" :key="index" class="option-with-dropdown">
40                      <div class="sidebar-item" @click="toggleDropdown(index)">
41                          <div class="sidebar-item-content">
42                              <i :class="option.icon"></i>
43                              <span>{{ option.title }}</span>
44                          </div>
45                          <i class="fas fa-chevron-down"></i>
46                      </div>
47                      <ul v-show="option.isActive" class="sidebar-submenu">
48                          <li v-for="child in children" :key="child.id">
49                              <a href="#" class="sidebar-submenu-item" @click.prevent="selectChild(child)">
50                                  <i class="fas fa-chevron-right"></i>
51                                  {{ child.prenom }} {{ child.nom }}
52                              </a>
53                          </li>
54                      </ul>
55                  </li>

```

template > aside.sidebar > nav > ul.sidebar-menu > li.option-with-dropdown > ul.sidebar-submenu > li

Les enfants associés à l'option "Mes Enfants" sont affichés dans un sous-menu, chaque enfant étant représenté par son prénom et son nom.



The screenshot shows a code editor with the router.js file open. A modal component, UserChildren.vue, is being edited. The modal contains a form for adding a child, with fields for first name, last name, level, and birth date. Buttons for saving and canceling the form are also present.

```

57      </nav>
58  </aside>
59
60  <div class="modal" v-show="showForm">
61      <div class="modal-background" @click="closeForm"></div>
62      <div class="modal-content">
63          <form @submit.prevent="submitForm">
64              <h3>Détails de {{ selectedChild.prenom }} {{ selectedChild.nom }}</h3>
65              <div class="form-group">
66                  <label for="firstName">Nom:</label>
67                  <input id="firstName" v-model="selectedChild.prenom" type="text" placeholder="Nom" disabled>
68              </div>
69              <div class="form-group">
70                  <label for="lastName">Prénom:</label>
71                  <input id="lastName" v-model="selectedChild.nom" type="text" placeholder="Prénom" disabled>
72              </div>
73              <div class="form-group">
74                  <label for="courses">Niveau:</label>
75                  <input id="courses" v-model="selectedChild.niveau" placeholder="Le niveau de l'enfant" disabled type="text">
76              </div>
77              <button type="submit">Enregistrer</button>
78              <button type="button" @click="closeForm">Annuler</button>
79          </form>
80      </div>
81  </div>
82
83  <!-- Formulaire Modal pour Ajouter un Enfant -->

```

template > aside.sidebar > nav > ul.sidebar-menu > li.option-with-dropdown > ul.sidebar-submenu > li

Ces fonctionnalités assurent une interface utilisateur fluide et interactive, permettant aux parents de gérer et visualiser facilement les informations de leurs enfants dans l'application.

- Le code pour ajouter un enfant :

Lorsque showAddChildForm est vrai, le formulaire s'affiche, permettant aux utilisateurs de saisir les informations de l'enfant : prénom, nom, niveau, et date de naissance. Tous les champs sont requis pour garantir des données complètes.

The screenshot shows a code editor with three tabs: router.js, UtilisateurParent.vue, and UserChildren.vue. The UtilisateurParent.vue tab is active, displaying the following code:

```
78     <button type="button" @click="closeForm">Annuler</button>
79   </form>
80 </div>
81 </div>
82
83 <!-- Formulaire Modal pour Ajouter un Enfant -->
84 <div class="modal" v-show="showAddChildForm">
85   <div class="modal-background" @click="closeAddChildForm"></div>
86   <div class="modal-content">
87     <form @submit.prevent="addNewChild">
88       <i class="fas fa-user-circle text-white text-2xl"></i>
89       <input v-model="newChild.prenom" type="text" placeholder="Nom" required>
90       <input v-model="newChild.nom" type="text" placeholder="Prénom" required>
91       <input v-model="newChild.niveau" type="text" placeholder="Niveau" required>
92       <input v-model="newChild.dateOfBirth" type="date" placeholder="Date de naissance" required>
93       <button type="submit">Ajouter</button>
94       <button type="button" @click="closeAddChildForm">Annuler</button>
95     </form>
96   </div>
97 </div>
```

La partie script :

Dans la fonction data, on a initialisé plusieurs propriétés, notamment l'état de la barre latérale, du menu de profil. Et on a défini également des objets pour l'enfant actuellement sélectionné et pour le nouvel enfant à ajouter.

The screenshot shows the script section of the UtilisateurParent.vue component. It defines several properties and methods:

```
1+ usages ± BoukerMohammed +1*
111 export default {
112   name: 'UtilisateurParent',
113   data() {
114     return {
115       isSidebarOpen: false,
116       showProfileMenu: false,
117       showForm: false,
118       showAddChildForm: false,
119       selectedChild: { prenom: '', nom: '', niveau: '' },
120       newChild: { prenom: '', nom: '', niveau: '', dateOfBirth: '' },
121       opciones: [
122         {
123           title: 'Mes Enfants',
124           icon: 'fa fa-users',
125           isActive: false,
126           items: []
127         },
128         {
129           title: 'Ajouter un enfant',
130           icon: 'fas fa-plus-circle',
131           isActive: false,
132           items: []
133         }
134       ],
135       token: '' // Ajouter ici pour stocker le token
136     };
137   }
138 }
```

Dans le hook created, la méthode fetchChildren est appelée pour charger les données des enfants :

```
router.js      ▼ UtilisateurParent.vue × ▼ UserChildren.vue

138
139     created() {
140         this.fetchChildren();
141         this.token = this.$route.params.token; // Initialiser le token à partir des paramètres de la route
142     },
143     methods: {
144         toggleProfileMenu() {
145             this.showProfileMenu = !this.showProfileMenu;
146         },
147         logout() {
148             localStorage.removeItem( key: 'token' );
149             this.$router.push('/ConnexionPage');
150         },
151         toggleSidebar() {
152             this.isSidebarOpen = !this.isSidebarOpen;
153         },
154         closeSidebar() {
155             this.isSidebarOpen = false;
156         },
157         toggleDropdown(index) {
158             if (this.opciones[index].title === 'Ajouter un enfant') {
159                 this.openAddChildForm();
160             } else {
161                 this.opciones[index].isActive = !this.opciones[index].isActive;
162             }
163         }
164     }
165 }
```

Les méthodes openForm et closeForm gèrent l'affichage du formulaire de détails de l'enfant sélectionné, tandis que openAddChildForm et closeAddChildForm gèrent l'affichage du formulaire d'ajout d'un nouvel enfant.

```
router.js      ▼ UtilisateurParent.vue × ▼ UserChildren.vue
166
167     } else {
168         this.opciones[index].isActive = !this.opciones[index].isActive;
169     }
170 },
171 openForm() {
172     this.showForm = true;
173 },
174 closeForm() {
175     this.showForm = false;
176 },
177 openAddChildForm() {
178     this.showAddChildForm = true;
179 },
180 closeAddChildForm() {
181     this.showAddChildForm = false;
182 },
183 async fetchChildren() {
184     try {
185         const response = await axios.get( url: 'http://localhost:8000/api/show/parent/enfant/' );
186         console.log('Response data:', response.data);
187         if (Array.isArray(response.data) && Array.isArray(response.data[0])) {
188             this.children = response.data[0]; // Assigner le tableau imbriqué
189         } else {
190             this.children = response.data;
191         }
192         this.loading = false;
193     } catch (error) {
194     }
195 }
```

Enfin on ajoute la méthode addNewChild qui envoie une requête POST à l'API pour ajouter un nouvel enfant, met à jour la liste des enfants et ferme le formulaire d'ajout.

```

185     } catch (error) {
186       console.error('Erreur lors de la récupération des enfants:', error);
187       this.loading = false;
188       this.error = true;
189     }
190   },
191   selectChild(child) {
192     this.selectedChild = child;
193     this.openForm();
194   },
195   addNewChild() {
196     axios.post('http://localhost:8000/api/children', this.newChild)
197       .then(response => {
198         console.log("Enfant ajouté:", response.data);
199         this.fetchChildren();
200         this.closeAddChildForm();
201       })
202       .catch(error => {
203         console.error("Erreur lors de l'ajout de l'enfant:", error);
204       });
205   },
206   submitForm() {
207     console.log("Updating child:", this.selectedChild);
208     this.closeForm();
209   }
210 }
211 
```

Le profil de l'utilisateur:

On a implémenté une section dédiée à la gestion du profil utilisateur, qui commence par le chargement des données de l'utilisateur. Lors de l'initialisation du composant.

localhost:8081/userprofile



la méthode fetchUserProfile est appelée pour récupérer les informations du profil depuis l'API. Pendant ce temps, un message de chargement est affiché.

```
,  
    created() {  
        this.fetchUserProfile();  
    },  
    methods: {  
        async fetchUserProfile() {  
            try {  
                const response = await axios.get(url: 'http://localhost:8000/api/my-profile');  
                console.log(response.data);  
                this.user = response.data.user;  
                this.loading = false;  
            } catch (error) {  
                console.error('Erreur lors de la récupération du profil:', error);  
                this.loading = false;  
                this.error = true;  
            }  
        },  
    },
```

Pour une expérience utilisateur fluide, on a ajouté des indicateurs visuels pour informer l'utilisateur du statut du chargement des données ou des erreurs éventuelles. Si les données ne sont pas encore disponibles, un message de chargement est affiché. En cas d'erreur lors de la récupération des données, un message d'erreur approprié est présenté.



Une fois les données chargées, on affiche un formulaire permettant à l'utilisateur de voir et de modifier certaines informations de son profil. Les champs email et fonction sont désactivés pour l'édition, tandis que le nom peut être mis à jour. Le formulaire utilise la méthode updateProfile pour envoyer les modifications à l'API.

The screenshot shows the code editor interface with two tabs: 'router.js' and 'UserProfile.vue'. The 'UserProfile.vue' tab is active, displaying the component's template. The template includes a loading state, an error message, and a form for updating the user profile. The code uses Vuetify components like 'v-if', 'v-else-if', and 'v-model'.

```
<template>
<div class="user-profile">
  <h1>Profil Utilisateur</h1>
  <div v-if="loading" class="loader">Chargement du profil...</div>
  <div v-else-if="error" class="error-message">Erreur lors de la récupération du profil. Veuillez réessayer plus tard.</div>
  <div v-else>
    <form @submit.prevent="updateProfile" class="profile-form">
      <div class="form-group">
        <label for="email" class="form-label"><i class="fas fa-envelope"></i> Email:</label>
        <input type="email" id="email" v-model="user.email" disabled class="form-input">
      </div>
      <div class="form-group">
        <label for="name" class="form-label"><i class="fas fa-user"></i> Nom:</label>
        <input type="text" id="name" v-model="user.name" class="form-input">
      </div>
      <div class="form-group">
        <label for="role" class="form-label"><i class="fas fa-briefcase"></i> Fonction:</label>
        <input type="text" id="role" v-model="user.fonction" class="form-input" >
      </div>
      <button type="submit" class="profile-submit">Mettre à jour le profil</button>
    </form>
  </div>
</div>
</template>
<script>
```

Et Pour permettre aux utilisateurs de mettre à jour leur nom, on a ajouté une méthode updateProfile qui envoie une requête PUT à l'API avec les nouvelles données.

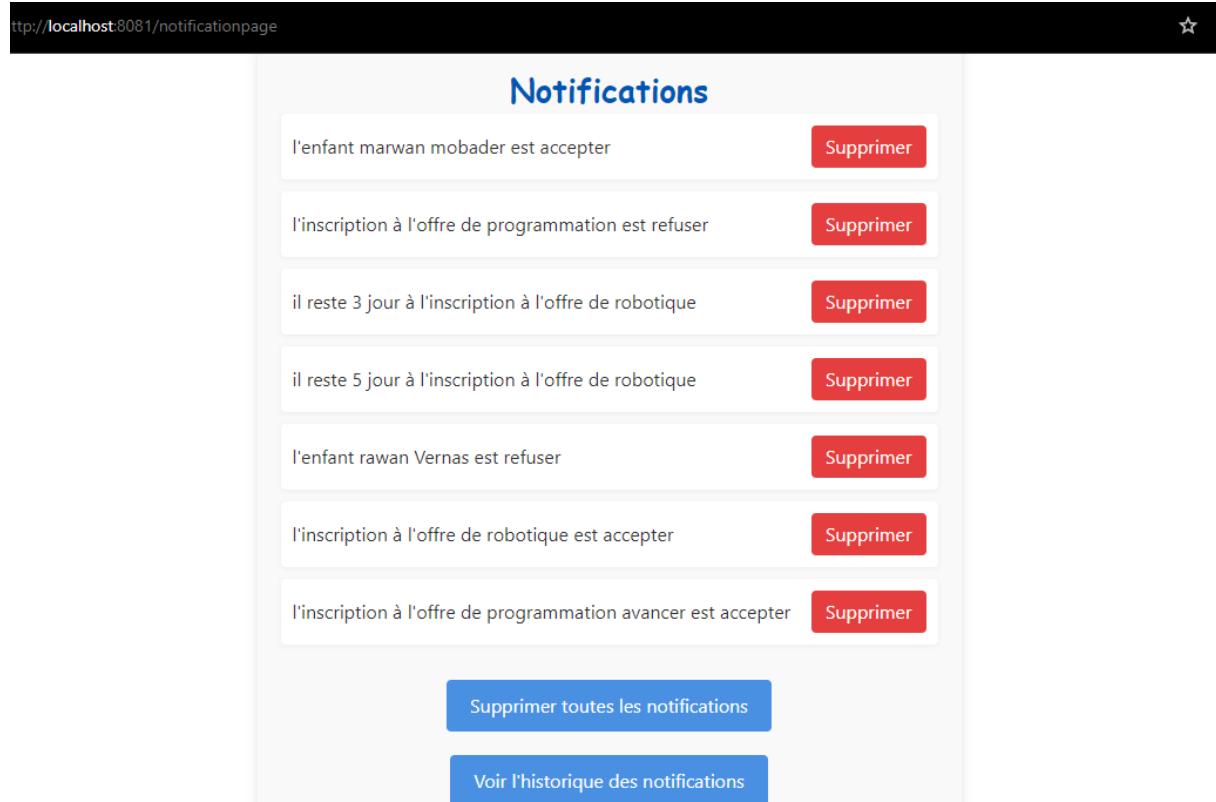
The screenshot shows the code editor interface with the 'UserProfile.vue' component selected. The script section contains an asynchronous function 'updateProfile' that sends a PUT request to 'http://localhost:8000/api/my-profile' with the user's name. It logs a success message to the console and displays an alert. If there is an error, it logs the error and displays an error message.

```
async updateProfile() {
  try {
    const response = await axios.put('http://localhost:8000/api/my-profile', {
      name: this.user.name,
    });
    console.log("Mise à jour des données de l'utilisateur réussie:", response.data);
    alert('Profil mis à jour avec succès');
  } catch (error) {
    console.error('Erreur lors de la mise à jour du profil:', error);
    alert('Erreur lors de la mise à jour du profil. Veuillez réessayer plus tard.');
  }
}
```

Notification :

J'ai utilisé trois composants pour créer l'interface de notification pour l'utilisateur :

1. NotificationPage: Ce composant gère les dernières notifications de l'utilisateur.
2. NotificationItem: Utilisé par le composant NotificationPage pour afficher chaque notification.
3. NotificationHistory: Ce composant affiche l'historique des notifications de l'utilisateur.



The screenshot shows a web application interface titled "Notifications". At the top left is the URL "http://localhost:8081/notificationpage". At the top right is a star icon. The main content area is titled "Notifications" in blue. It displays a list of seven notifications, each with a "Supprimer" (Delete) button to its right. The notifications are:

- l'enfant marwan mobader est accepter
- l'inscription à l'offre de programmation est refuser
- il reste 3 jour à l'inscription à l'offre de robotique
- il reste 5 jour à l'inscription à l'offre de robotique
- l'enfant rawan Vernas est refuser
- l'inscription à l'offre de robotique est accepter
- l'inscription à l'offre de programmation avancer est accepter

Below the list are two blue buttons: "Supprimer toutes les notifications" (Delete all notifications) and "Voir l'historique des notifications" (View notification history).

La page NotificationPage est dédiée à la gestion des notifications de l'utilisateur. On a implémenté ce composant pour afficher la liste des notifications, permettre leur suppression individuelle ou globale, et naviguer vers l'historique des notifications.

- Affichage des Notifications :

On a utilisé une boucle v-for dans le composant NotificationPage pour afficher chaque notification sous forme d'éléments NotificationItem. Ces éléments reçoivent les données de notification via des props et émettent un événement delete lors de la suppression.

```

1 <template>
2   <div class="notifications-page">
3     <h1>Notifications</h1>
4     <ul>
5       <notification-item
6         v-for="notification in notifications"
7         :key="notification.id"
8         :notification="notification"
9         @delete="deleteNotification"
10      ></notification-item>
11    </ul>
12    <div class="button-container">
13      <button @click="deleteAllNotifications">Supprimer toutes les notifications</button>
14    </div>
15    <div class="button-container">
16      <button @click="viewHistory">Voir l'historique des notifications</button>
17    </div>
18  </div>
19 </template>

```

- Chargement des Notifications :

On a créé une méthode loadNotifications pour récupérer les notifications depuis l'API au chargement du composant. En cas d'échec, un message d'erreur est affiché.

```

55   async loadNotifications() {
56     try {
57       const response = await axios.get(url: 'http://localhost:8000/api/show/notification/parent/top/');
58       this.notifications = response.data;
59       this.loading = false;
60     } catch (error) {
61       console.error("Failed to load notifications:", error);
62       this.loading = false;
63       this.error = true;
64     }
65   },
66   created() {
67     this.loadNotifications();
68   }
69 }

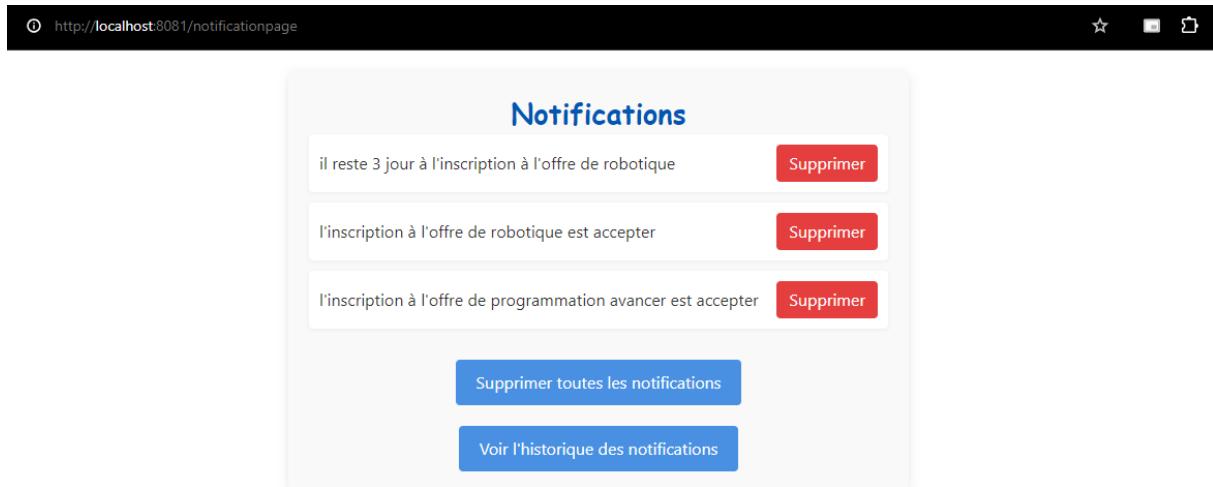
```

Cette page affiche les 7 dernières notifications qui ont été reçues dans l'application.

- Suppression des Notifications :

Pour permettre la suppression des notifications, on a ajouté des méthodes deleteNotification et deleteAllNotifications qui envoient des requêtes à l'API et mettent à jour l'état local des notifications.

La fonction deleteAllNotifications supprime toutes les notifications de l'utilisateur.



```
36      async deleteNotification(id) {
37        try {
38          await axios.delete(`url: 'http://localhost:8000/api/delete/notification/${id}`);
39          this.notifications = this.notifications.filter(n => n.id !== id);
40        } catch (error) {
41          console.error("Failed to delete notification:", error);
42        }
43      },
44      async deleteAllNotifications() {
45        try {
46          await axios.get(`url: 'http://localhost:8000/api/delete/notification/all/'`);
47          this.notifications = [];
48        } catch (error) {
49          console.error("Failed to delete all notifications:", error);
50        }
51      },
52      viewHistory() {
53        this.$router.push('/notificationhistory');
54      },

```

- Élément de Notification :

Le composant `NotificationItem` est utilisé pour afficher une seule notification avec une option pour la supprimer. On a stylisé cet élément pour le rendre visuellement distinct et interactif.

```
router.js  NotificationHistory.vue  NotificationItem.vue  NotificationsPage.vue  UserChildren.vue
2
3  <template>
4    <li class="notification-item">
5      <p>{{ notification.contenu }}</p>
6      <button @click="$emit('delete', notification.id)">Supprimer</button>
7    </li>
8  </template>
9
10 <script>
11   1+ usages  ± Anass El Ouahabi
12   export default {
13     props: {
14       notification: Object
15     }
16   }
17 </script>
18
19 <style scoped>
20   .notification-item {
21     display: flex;
22     align-items: center;
23     justify-content: space-between;
24     padding: 10px;
25     background: white;
26     margin-bottom: 10px;
27     border-radius: 4px;
28     box-shadow: 0 1px 4px rgba(0,0,0,0.05);
29   }
30 
```

```
router.js  NotificationHistory.vue  NotificationItem.vue  NotificationsPage.vue  UserChildren.vue
28
29
30   .notification-item p {
31     margin: 0;
32     color: #333;
33     flex-grow: 1;
34   }
35
36   button {
37     background-color: #e53e3e;
38     color: white;
39     border: none;
40     padding: 6px 12px;
41     border-radius: 4px;
42     cursor: pointer;
43     transition: background-color 0.2s;
44   }
45
46   button:hover {
47     background-color: #c53030;
48   }
49
50   button:focus {
51     outline: none;
52     box-shadow: 0 0 0 2px #fff, 0 0 0 4px #c53030;
53   }
54 </style>
```

- NotificationHistory :

La page NotificationHistory affiche l'historique des notifications. On a utilisé la même structure que pour NotificationPage.

```
<template>
  <div class="notifications-history-page">
    <h1>Historique des Notifications</h1>
    <ul>
      <notification-item
        v-for="notification in historyNotifications"
        :key="notification.id"
        :notification="notification"
        @delete="deleteNotification"
      ></notification-item>
    </ul>
  </div>
</template>
```

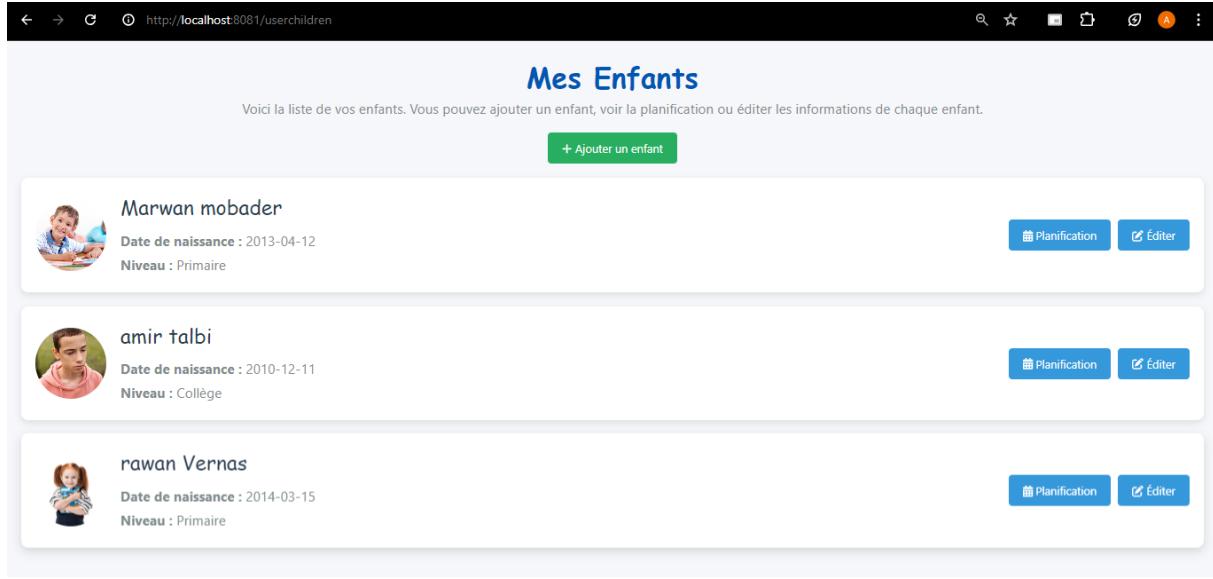
On a utilisé la fonction `loadHistoryNotification` afin de récupérer tous les notifications de l'utilisateur sauf les 7 notifications déjà afficher dans la page `NotificationPage`.

On a également ajouté une méthode pour supprimer des notifications spécifiques de l'historique.

```
created() {
  this.loadHistoryNotifications();
},
methods: {
  async deleteNotification(id) {
    try {
      await axios.delete(`http://localhost:8000/api/delete/notification/${id}`);
      this.historyNotifications = this.historyNotifications.filter(n => n.id !== id);
    } catch (error) {
      console.error("Failed to delete notification:", error);
    }
  },
  async loadHistoryNotifications() {
    try {
      const response = await axios.get('http://localhost:8000/api/show/notification/parent/remaining/');
      this.historyNotifications = response.data;
      this.loading = false;
    } catch (error) {
      console.error("Failed to load notifications:", error);
      alert(error.response.data.message);
      this.loading = false;
      this.error = true;
    }
  }
}
```

Mes enfants :

On a créé le composant UserChildren pour afficher et gérer les informations des enfants d'un utilisateur. Ce composant permet à l'utilisateur de voir la liste de ses enfants, ajouter un enfant, voir la planification et éditer les informations de chaque enfant.

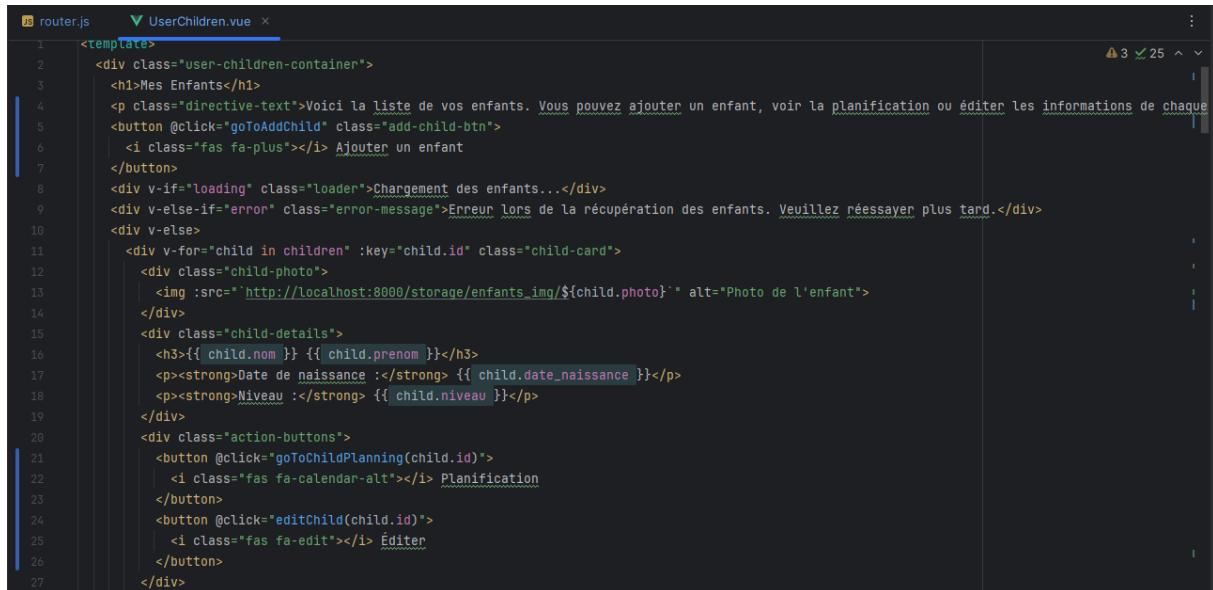


The screenshot shows a web application interface titled "Mes Enfants". At the top, there is a header with a search icon, a star icon, a folder icon, and other navigation icons. Below the header, the title "Mes Enfants" is displayed in bold blue text. A sub-instruction "Voici la liste de vos enfants. Vous pouvez ajouter un enfant, voir la planification ou éditer les informations de chaque enfant." is present. A green button labeled "+ Ajouter un enfant" is located at the top right. The main content area displays three child entries, each in a card format:

- Marwan mobader**: Includes a photo of a boy, date of birth (2013-04-02), level (Primaire), and buttons for "Planification" and "Éditer".
- amir talbi**: Includes a photo of a boy, date of birth (2010-12-11), level (Collège), and buttons for "Planification" and "Éditer".
- rawan Vernas**: Includes a photo of a girl, date of birth (2014-03-15), level (Primaire), and buttons for "Planification" and "Éditer".

■ La partie Template :

Le Template du composant UserChildren comprend plusieurs sections : un titre, un texte explicatif, un bouton pour ajouter un enfant, et une liste des enfants. Chaque enfant est représenté par une carte contenant une photo, des détails, et des boutons pour la planification et l'édition.



```
router.js          UserChildren.vue
```

```
1  <template>
2   <div class="user-children-container">
3     <h1>Mes Enfants</h1>
4     <p>Voici la liste de vos enfants. Vous pouvez ajouter un enfant, voir la planification ou éditer les informations de chaque enfant.</p>
5     <button @click="goToAddChild" class="add-child-btn">
6       <i class="fas fa-plus"></i> Ajouter un enfant
7     </button>
8     <div v-if="loading" class="loader">Changement des enfants...</div>
9     <div v-else-if="error" class="error-message">Erreur lors de la récupération des enfants. Veuillez réessayer plus tard.</div>
10    <div v-else>
11      <div v-for="child in children" :key="child.id" class="child-card">
12        <div class="child-photo">
13          
14        </div>
15        <div class="child-details">
16          <h3>{{ child.nom }} {{ child.prenom }}</h3>
17          <p><strong>Date de naissance :</strong> {{ child.date_naissance }}</p>
18          <p><strong>Niveau :</strong> {{ child.niveau }}</p>
19        </div>
20        <div class="action-buttons">
21          <button @click="goToChildPlanning(child.id)">
22            <i class="fas fa-calendar-alt"></i> Planification
23          </button>
24          <button @click="editChild(child.id)">
25            <i class="fas fa-edit"></i> Éditer
26          </button>
27        </div>
28      </div>
29    </div>
30  </div>
```

■ La partie script :

On introduit la méthode fetchChildren qui est essentielle dans le composant UserChildren afin de récupérer les données des enfants depuis le serveur backend.

Dans cette fonction on a vérifié si les données reçues sont un tableau et si le premier élément de ce tableau est également un tableau. Si c'est le cas, on assigne le premier tableau imbriqué à this.children. Sinon, on assigne directement les données reçues à this.children. Cela permet de gérer différents formats de réponse potentiels du backend.

```
router.js          UserChildren.vue ×
42      error: false
43  };
44  },
45  created() {
46    this.fetchChildren();
47  },
48  methods: {
49    async fetchChildren() {
50      try {
51        const response = await axios.get(url: 'http://localhost:8000/api/show/parent/enfant/');
52        console.log('Response data:', response.data);
53        if (Array.isArray(response.data) && Array.isArray(response.data[0])) {
54          this.children = response.data[0]; // Assigné le tableau imbriqué
55        } else {
56          this.children = response.data;
57        }
58        this.loading = false;
59      } catch (error) {
60        console.error('Erreur lors de la récupération des enfants:', error);
61        this.loading = false;
62        this.error = true;
63      }
64    },

```

▪ Ajouter un enfant :

Dans cette section, nous avons utilisé un composant Vue.js pour créer une interface permettant d'ajouter un nouvel enfant. Le formulaire d'ajout comprend des champs pour le nom, le prénom, la date de naissance, le niveau et la photo de l'enfant.

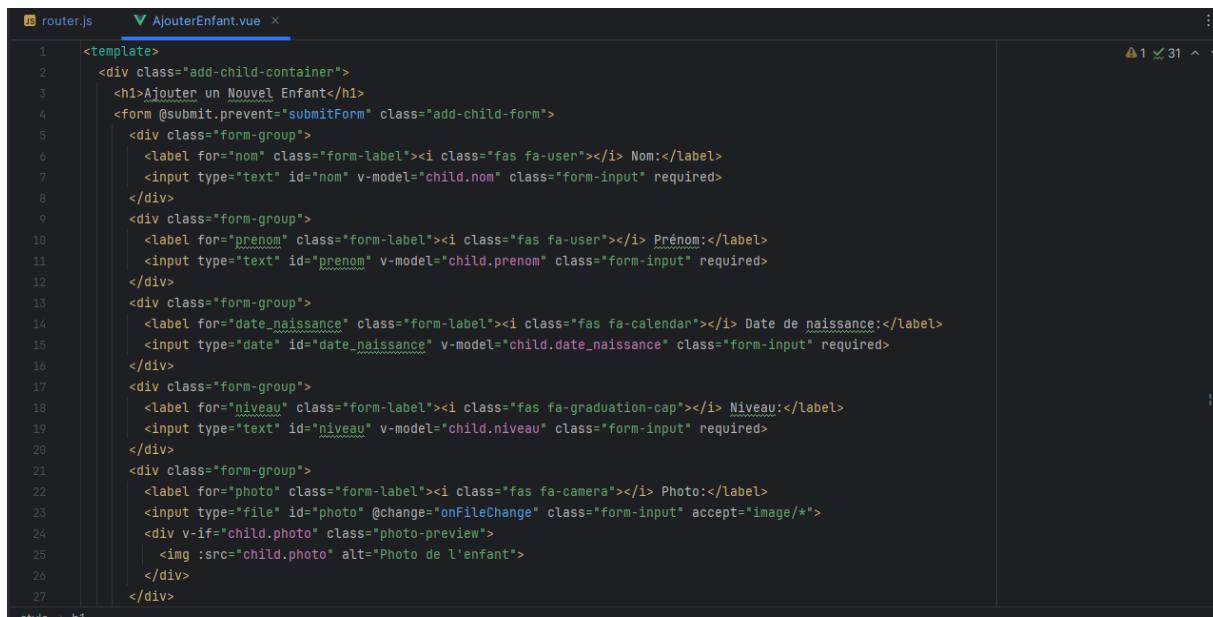
The screenshot shows a web browser window with the URL <http://localhost:8081/AjouterEnfant>. The page title is "Ajouter un Nouvel Enfant". The form contains five fields:

- Nom:** A text input field.
- Prénom:** A text input field.
- Date de naissance:** A date input field with a placeholder "jj/mm/aaaa" and a calendar icon.
- Niveau:** A text input field.
- Photo:** A file input field with a placeholder "Choisir un fichier" and a message "Aucun fichier choisi".

At the bottom of the form is a blue button labeled "Ajouter l'enfant".

- La template :

Le template HTML inclut un formulaire avec plusieurs champs d'entrée pour les informations de l'enfant. Chaque champ est lié au modèle de données `child` à l'aide de la directive `v-model`.



```

1 <template>
2   <div class="add-child-container">
3     <h1>Ajouter un Nouvel Enfant</h1>
4     <form @submit.prevent="submitForm" class="add-child-form">
5       <div class="form-group">
6         <label for="nom" class="form-label"><i class="fas fa-user"></i> Nom:</label>
7         <input type="text" id="nom" v-model="child.nom" class="form-input" required>
8       </div>
9       <div class="form-group">
10        <label for="prenom" class="form-label"><i class="fas fa-user"></i> Prénom:</label>
11        <input type="text" id="prenom" v-model="child.prenom" class="form-input" required>
12      </div>
13      <div class="form-group">
14        <label for="date_naissance" class="form-label"><i class="fas fa-calendar"></i> Date de naissance:</label>
15        <input type="date" id="date_naissance" v-model="child.date_naissance" class="form-input" required>
16      </div>
17      <div class="form-group">
18        <label for="niveau" class="form-label"><i class="fas fa-graduation-cap"></i> Niveau:</label>
19        <input type="text" id="niveau" v-model="child.niveau" class="form-input" required>
20      </div>
21      <div class="form-group">
22        <label for="photo" class="form-label"><i class="fas fa-camera"></i> Photo:</label>
23        <input type="file" id="photo" @change="onFileChange" class="form-input" accept="image/*">
24        <div v-if="child.photo" class="photo-preview">
25          
26        </div>
27      </div>
28    </div>
29  </template>

```

- La partie script :

On a ajouté une méthode `onFileChange` qui est déclenchée lorsque l'utilisateur sélectionne une photo. Elle utilise un `FileReader` pour lire le fichier et convertir l'image en une URL de données (data URL). L'URL de données est ensuite assignée à la propriété `photo` de l'objet `child`.



```

40   },
41
42   methods: {
43     onFileChange(e) {
44       const file = e.target.files[0];
45       if (file) {
46         const reader = new FileReader();
47         reader.onload = (e) => {
48           this.child.photo = e.target.result;
49         };
50         reader.readAsDataURL(file);
51       }
52     },
53   },

```

La méthode `submitForm`: est déclenchée lors de la soumission du formulaire. Elle crée un objet `FormData` et y ajoute les propriétés de l'enfant. Si une photo est présente, elle est convertie en un `Blob` (via la méthode `dataURLtoBlob`) et ajoutée aux `FormData`. Les données sont ensuite envoyées au backend via une requête POST avec `axios`. En cas de succès, l'utilisateur est redirigé vers la liste des enfants.

```

59 },
60     async submitForm() {
61         try {
62             // Préparez les données du formulaire pour l'envoi
63             const formData = new FormData();
64             formData.append('nom', this.child.nom);
65             formData.append('prenom', this.child.prenom);
66             formData.append('date_naissance', this.child.date_naissance);
67             formData.append('niveau', this.child.niveau);
68             if (this.child.photo) {
69                 formData.append('photo', this.dataURLtoBlob(this.child.photo));
70             }
71
72             // Envoyez les données au backend
73             await axios.post('http://localhost:8000/api/children', formData, { config: {
74                 headers: {
75                     'Content-Type': 'multipart/form-data'
76                 }
77             });
78
79             // Redirigez vers la liste des enfants après l'ajout
80             this.$router.push({ name: 'userchildren' });
81         } catch (error) {
82             console.error('Erreur lors de l\'ajout de l\'enfant:', error);
83         }
84     },
85     dataURLtoBlob(dataURL) {

```

On a utilisé aussi une méthode **dataURLtoBlob** qui sert à convertir une URL de données en un objet Blob. C'est nécessaire pour envoyer des fichiers binaires (comme les images) via FormData.

```

        },
        dataURLtoBlob(dataURL) {
            const arr = dataURL.split(',');
            const mime = arr[0].match(/:(.*?);/)[1];
            const bstr = atob(arr[1]);
            let n = bstr.length;
            const u8arr = new Uint8Array(n);
            while (n--) {
                u8arr[n] = bstr.charCodeAt(n);
            }
            return new Blob([u8arr], { type: mime });
        }
    };
}

```

▪ Planification de l'enfant :

Ce composant Vue.js affiche la planification des activités pour un enfant spécifique. Il utilise l'API pour récupérer les données et les affiche dans une interface utilisateur attrayante.

Veuillez trouver ci-dessous les activités planifiées pour votre enfant.

dimanche	dimanche
Introduction à Python 10:00:00 - 12:30:00 Animé par : Sami Towne	Introduction à Python 12:30:00 - 15:00:00 Animé par : Sami Towne

- La partie Template :

```

<template>
  <div class="child-planning">
    <h1>Planification de l'enfant</h1>
    <p class="instruction-text">Veuillez trouver ci-dessous les activités planifiées pour votre enfant.</p>
    <div v-if="loading" class="loader">Chargement de la planification...</div>
    <div v-else-if="error" class="error-message">Erreur lors de la récupération de la planification. Veuillez réessayer plus tard.</div>
    <div v-else class="calendar">
      <div class="week">
        <div v-for="(plan, index) in planning" :key="index" class="day">
          <h2>{{ plan.jour }}</h2>
          <div class="activity">
            <p><strong>{{ plan.titre }}</strong></p>
            <p>{{ plan.heure_debut }} - {{ plan.heure_fin }}</p>
            <p><span class="animer_par">Animé par :</span> {{ plan.name }}</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

```

- La partie script :

On a utilisé `this.$route.params.id` pour obtenir l'ID de l'enfant depuis les paramètres de la route.

```

  { path: '/children/:id', component: UserChildren, name: "userchildren" , meta: { requiresAuth: true , roles: ['parent'] }},
  { path: '/childplanning/:id', component: ChildPlanning, name: "childplanning" , meta: { requiresAuth: true , roles: ['parent'] }},
  { path:'editChild/:id' , component:EditChild , name:"editChild" , meta: { requiresAuth: true , roles: ['parent'] }},
```

```
js router.js    ChildPlanning.vue ×
22 import axios from '@/axios';
23 1+ usages  Anass El Ouahabi *
24 export default {
25   name: 'ChildPlanning',
26   data() {
27     return {
28       planning: [],
29       loading: true,
30       error: false
31     };
32   },
33   created() {
34     this.fetchPlanning();
35   },
36   methods: {
37     async fetchPlanning() {
38       const enfantId = this.$route.params.id;
39       try {
40         const response = await axios.get(`http://localhost:8000/api/show/enfant/planning/${enfantId}`);
41         console.log(`La réponse est`, response.data);
42         this.planning = response.data;
43         this.loading = false;
44       } catch (error) {
45         console.error('Erreur lors de la récupération de la planification:', error);
46         this.loading = false;
47         this.error = true;
48       }
49     }
50   }
51 }
```

- La partie style :

```
js router.js    ChildPlanning.vue ×
52 <style scoped>
53 .child-planning {
54   padding: 20px;
55   text-align: center;
56   background: #f5f7fa;
57   color: #333;
58 }
59
60 h1 {
61   font-size: 2.5rem;
62   margin-bottom: 20px;
63   font-family: 'Baloo Bhaijaan 2', cursive;
64   color: #0056b3;
65   font-weight: bold;
66 }
67
68 .instruction-text {
69   font-size: 1.2rem;
70   color: #4e6267;
71   margin-bottom: 20px;
72 }
73
74 .loader {
75   font-size: 1.5rem;
76   color: #3498db;
77 }
```

```
77 }
78
79 .error-message {
80   color: red;
81   font-size: 1.2rem;
82 }
83
84 .calendar {
85   display: flex;
86   flex-direction: column;
87   gap: 20px;
88 }
89
90 .week {
91   display: flex;
92   justify-content: space-around;
93   flex-wrap: wrap;
94   margin-top: 20px;
95   gap: 20px;
96 }
97
98 .day {
99   background: #ffffff;
100  padding: 20px;
101  border-radius: 10px;
102  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
103  flex: 1;
```

```

127 margin-bottom: 10px;
128 transition: background-color 0.3s;
129 }
130
131 .activity p {
132   margin: 10px 0;
133   font-size: 1.1rem;
134   color: #34495e;
135 }
136
137 .activity strong {
138   color: #3498db;
139   font-size: 1.2rem;
140 }
141
142 .activity:hover {
143   background-color: #d0e4f7;
144 }
145
146 .animer_par {
147   font-size: 1.2rem;
148   color: #7f8c8d;
149   font-weight: bold;
150 }
151 </style>
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370

```

▪ [Editer un enfant:](#)

Le composant `EditChild` permet aux utilisateurs de modifier les informations d'un enfant existant. Il offre une interface utilisateur pour mettre à jour le nom, le prénom, la date de naissance, le niveau et la photo de l'enfant. Les données de l'enfant sont chargées lors de la création du composant en utilisant une requête API. Les modifications apportées sont soumises via un formulaire multipart/form-data, permettant également le téléchargement de fichiers d'image. Un bouton de suppression est également disponible pour permettre la suppression

http://localhost:8081/editChild/1

Éditer l'Enfant

Nom:

Prénom:

Date de naissance:

Niveau:

Photo: Choisir un fichier Aucun fichier choisi

Enregistrer les modifications **Supprimer l'enfant**

- Mes demandes :

Programmation entre
15 et 17 ans

Cours de programmation intermédiaire pour...

En accédant sur la partie de Mes demande en trouve tous Les demandes soumises par le parents, avec leur date de demande .

http://localhost:8081/parentrequests

Mes Demande

Date de création : 27/05/2024	Voir Détails
Date de création : 27/05/2024	Voir Détails
Date de création : 17/04/2024	Voir Détails

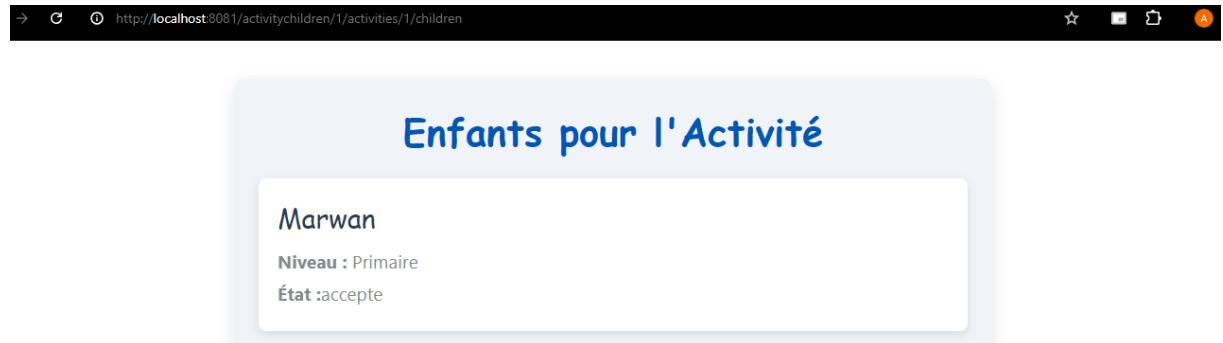
- Le details de chaque demande :

Activités pour la Demande

Introduction à Python Objectif : Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples. Domaine : Programmation	Voir Enfants
Algorithmique et Structures de Données Objectif : Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes. Domaine : Structures de Données	Voir Enfants

Cette page affiche toutes les activités sélectionnées par le parent lors de l'inscription de ses enfants à une offre spécifique.

En cliquant sur le bouton pour voir les enfants, on peut accéder à la liste de tous les enfants que le parent a choisis pour inscrire à une activité. Ces enfants sont affichés avec leur état, indiquant s'ils ont été acceptés ou refusés.



- Ca partie template :

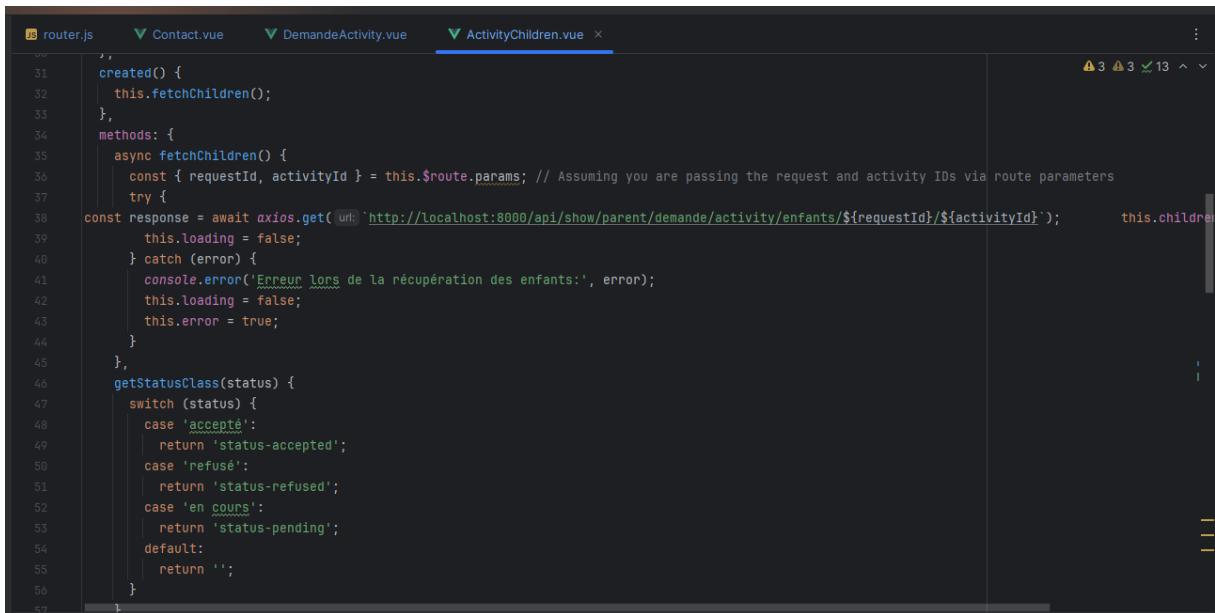
Dans cette section du code, nous avons structuré notre template avec des éléments HTML pour afficher la liste des enfants. Le v-for est utilisé pour itérer sur le tableau des enfants et afficher chaque enfant individuellement. La directive conditionnelle v-if gère l'affichage du chargement et des messages d'erreur.

```
1 <template>
2   <div class="activity-children">
3     <h1>Enfants pour l'Activité</h1>
4     <div v-if="loading">Chargement des enfants...</div>
5     <div v-else-if="error" class="error-message">Erreur lors de la récupération des enfants. Veuillez réessayer plus tard.</div>
6     <div v-else>
7       <div v-for="child in children" :key="child.id" class="child-item">
8         <div class="child-details">
9           <n2>{{ child.nom }}</n2>
10          <p><strong>Niveau </strong> {{ child.niveau }}</p>
11          <p><strong>État </strong></p>
12          <span :class="getStatusClass(child.etat)">{{ child.etat }}</span>
13        </div>
14      </div>
15    </div>
16  </div>
17 </template>
18 <script>
19 import axios from '@axios';
20
```

- La partie script :

Dans la méthode fetchChildren, nous utilisons axios pour envoyer une requête GET à notre backend afin de récupérer les données des enfants pour une activité spécifique. Les paramètres de la route (requestId et activityId) sont utilisés pour construire l'URL de l'API. En cas de succès, les données des enfants sont assignées à this.children et l'état de chargement est mis à jour. En cas d'échec, une erreur est affichée.

La méthode `getStatusClass` attribue une classe CSS en fonction de l'état de l'enfant (accepté, refusé ou en cours), ce qui permet de styliser dynamiquement l'affichage de l'état.



```
js router.js    Contact.vue    DemandeActivity.vue    ActivityChildren.vue x
30
31  ' created() {
32    this.fetchChildren();
33  },
34  methods: {
35    async fetchChildren() {
36      const { requestId, activityId } = this.$route.params; // Assuming you are passing the request and activity IDs via route parameters
37      try {
38        const response = await axios.get(`http://localhost:8000/api/show/parent/demande/activity/enfants/${requestId}/${activityId}`);
39        this.loading = false;
40      } catch (error) {
41        console.error('Erreur lors de la récupération des enfants:', error);
42        this.loading = false;
43        this.error = true;
44      }
45    },
46    getStatusClass(status) {
47      switch (status) {
48        case 'accepté':
49          return 'status-accepted';
50        case 'refusé':
51          return 'status-refused';
52        case 'en cours':
53          return 'status-pending';
54        default:
55          return '';
56      }
57    }
58  }
59
```

➤ Parcours de l'utilisateur pour créer une demande :

Cherchant d'abord l'offre qu'on souhaite inscrire nos enfant :

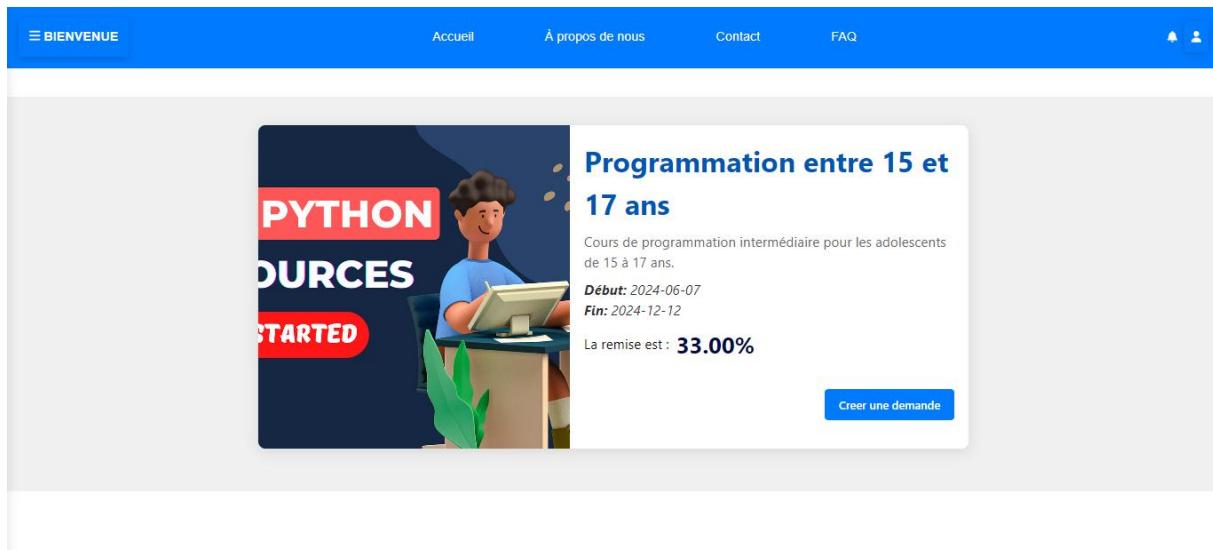


The screenshot shows a web interface titled "Toutes les Offres". A search bar contains the text "cours de progra". Below the search bar, there is a card for a programming offer. The card features an image of a person at a computer, the text "FREE PYTHON RESOURCES", and a "GET STARTED" button. Below the image, it says "Programmation entre 15 et 17 ans" and "Cours de programmation intermédiaire pour les adolescents de 15 à 17 ans...". A blue "Voir Détails" button is at the bottom of the card. At the bottom of the page, there are navigation buttons for "Précédent", "Page 1 sur 3", and "Suivant".

En cliquant sur voir détail en va diriger vers la page qui affiche le détail de chaque offre :

- Offre Détail :

Le composant est structuré de manière à afficher l'image de l'offre, son titre, une description, les dates de début et de fin, ainsi que le pourcentage de remise si applicable. L'interface utilisateur est soigneusement conçue pour offrir une expérience agréable et intuitive.



- La partie Template :

```
Offer.js      OfferDetails.vue
```

```
<template>
<UtilisateurParent />
<div class="offer-details-containier">
  <div class="offer-card" v-if="offer">
    <div class="offer-image">
      
    </div>
    <div class="offer-info">
      <div class="text-content">
        <h1>{{ offer.titre }}</h1>
        <p class="offer-description">{{ offer.description }}</p>
        <div class="date-info">
          <p><strong>Début:</strong> {{ formatDate(offer.date_debut) }}</p>
          <p><strong>Fin:</strong> {{ formatDate(offer.date_fin) }}</p>
        </div>
        <div class="remise-container">
          <p class="remise-label">La remise est :</p>
          <span v-if="offer.remise > 0" class="offer-price">{{ offer.remise }}%</span>
          <span v-else class="no-remise">Pas de remise pour cette offre</span>
        </div>
      </div>
      <div class="button-container">
        <button @click="GoToActivities">Créer une demande</button>
      </div>
    </div>
  </div>
</div>
```

La méthode **formatDate** prend une date au format string et retourne une date formatée en ne gardant que la partie avant le 'T' (partie de la date sans l'heure). C'est utile pour afficher les dates de début et de fin de l'offre de manière plus lisible.

```

router.js
OfferDetails.vue
  
```

```

43     loading: true,
44     error: false
45   },
46 },
47 created() {
48   this.fetchOfferDetails();
49 },
50 methods: {
51   async fetchOfferDetails() {
52     const offerid = this.$route.params.id; // Récupérer l'ID de l'offre depuis les paramètres de route
53     try {
54       const response = await axios.get(url: 'http://localhost:8000/api/show/offer/${offerid}');
55       this.offer = response.data;
56       this.loading = false;
57     } catch (error) {
58       console.error('Erreur lors de la récupération des détails de l\'offre:', error);
59       this.loading = false;
60       this.error = true;
61     }
62   },
63   formatDate(dateString) {
64     return dateString.split('T')[0];
65   },
66   GoToActivities() {
67     this.$router.push({ name: 'activitylist', params: { offerId: this.offer.id }, query: { offerTitre: this.offer.titre } });
68   }
69 }
  
```

En cliquant sur « créer une demande », l'utilisateur sera dirigé vers le composant affichant toutes les activités incluses dans cette offre.

- **List des activités :**

http://localhost:8081/activitylist/3?offerTitre=Programmation+entre+15+et+17+ans

Activités Disponibles

Choisissez les activités auxquelles vous souhaitez inscrire vos enfants. Cliquez sur 'Choisir les enfants' pour sélectionner les enfants pour une activité spécifique.

Introduction à Python

Cette activité introduit les bases de la programmation en Python, couvrant les concepts fondamentaux tels que les variables, les boucles, et les fonctions.

Objectifs : Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples.

Domaine : Programmation

Tarif : 157.45 €

[Show More](#) [Choisir les enfants](#)

Algorithmique et Structures de Données

Cette activité se concentre sur les concepts d'algorithmique et les structures de données fondamentales telles que les listes, les piles, et les arbres.

Objectifs : Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes.

Domaine : Structures de Données

Tarif : 67.00 €

[Show More](#) [Choisir les enfants](#)

En cliquant sur « Show More », l'utilisateur pourra afficher des informations supplémentaires concernant l'activité, telles que l'âge minimum, l'âge maximum, le nombre de séances, le volume horaire et les options de paiement.

Activités Disponibles

Choisissez les activités auxquelles vous souhaitez inscrire vos enfants. Cliquez sur 'Choisir les enfants' pour sélectionner les enfants pour une activité spécifique.

Introduction à Python

Cette activité introduit les bases de la programmation en Python, couvrant les concepts fondamentaux tels que les variables, les boucles, et les fonctions.

Objectifs : Apprendre les bases de Python, comprendre les concepts fondamentaux de la programmation, créer des programmes simples.

Domaine : Programmation

Tarif : 157.45 €

Show More **Choisir les enfants**

Âge Minimum : 15
Âge Maximum : 17
Nombre de Séances : 30
Volume Horaire : 60 heures
Option de Paiement : Cash
Vidéo : Voir la vidéo

Algorithmique et Structures de Données

```
        },
        toggleDetails(activityId) {
            const activity = this.activities.find(act => act.id === activityId);
            if (activity) {
                activity.showDetails = !activity.showDetails;
            }
        },
        goToActivityDetails(activity) {
            this.$router.push({
                name: 'choosechildren',
                query: {
                    activityId: activity.id,
                    activityTitre: activity.titre,
                    offerId: this.$route.params.offerId,
                    offerTitre: this.$route.query.offerTitre
                }
            });
        },
    }
},
```

toggleDetails : Cette méthode permet de basculer l'affichage des détails supplémentaires pour une activité spécifique.

Elle prend l'ID de l'activité en paramètre et trouve l'activité correspondante dans le tableau activities.

Ensuite, elle inverse la valeur de la propriété showDetails de cette activité, ce qui permet d'afficher ou de masquer les détails supplémentaires.

goToActivityDetails : Cette méthode est appelée lorsqu'un utilisateur clique sur le bouton "Choisir les enfants".

Elle utilise le router de Vue pour naviguer vers la route choosechildren, en passant l'ID de l'activité, le titre de l'activité, l'ID de l'offre et le titre de l'offre en tant que paramètres de route et query, respectivement.

Cela permet de diriger l'utilisateur vers le composant où il peut choisir les enfants pour une activité spécifique.

- **Selectioner les enfants :**

Le composant ChooseChildren permet aux parents de sélectionner les enfants à inscrire dans une activité spécifique

Lorsqu'il clique sur un de ses enfants, il est dirigé vers le composant des horaires qui affiche les horaires disponibles pour cette activité. Une fois qu'il a terminé la sélection des horaires pour tous les enfants qu'il souhaite inscrire dans cette activité, il clique sur "Terminer" et est redirigé vers la page des activités. Il peut alors choisir une autre activité et est à nouveau dirigé vers cette page pour sélectionner les enfants et leurs horaires. Ce processus se répète jusqu'à ce qu'il ait terminé l'inscription de tous ses enfants dans les activités souhaitées avec les horaires de leur choix.

Mes enfants

Choisissez les enfants auxquels vous souhaitez inscrire dans l'activité : [Introduction à Python](#)

Marwan mobader	Niveau: Primaire
amir talbi	Niveau: Collège
rawan Vernas	Niveau: Primaire

[Terminer](#)

The screenshot shows a web browser window with the URL <http://localhost:8081/selectschedule/1?activityTitre=Introduction+à+Python&offerId=3&offerTitre=Programmation+entre+15+et+17+ans&childId=1...>. The title of the page is "Les horaires disponibles". It displays three available time slots for Marwan mobader:

dimanche 10:00:00 - 12:30:00 Eff. Min: 13 Eff. Max: 25	<input checked="" type="checkbox"/>	Places Restantes: 3
dimanche 12:30:00 - 15:00:00 Eff. Min: 5 Eff. Max: 15	<input type="checkbox"/>	Places Restantes: 8
mardi 20:00:00 - 22:30:00 Eff. Min: 20 Eff. Max: 35	<input type="checkbox"/>	Places Restantes: 10

[Terminer](#)

Lorsque l'utilisateur termine l'inscription de tous ses enfants, il peut cliquer sur le bouton « Envoyer la demande » situé sur la page des activités de l'offre.

DONNÉES ET ALGORITHMES
COMPÉTENCES NUMÉRIQUES POUR TOUS
Apprendre les algorithmes

Algorithmique et Structures de Données

Cette activité se concentre sur les concepts d'algorithmique et les structures de données fondamentales telles que les listes, les piles, et les arbres.

Objectifs : Comprendre l'importance des algorithmes, apprendre à résoudre des problèmes complexes, manipuler des structures de données pour optimiser les programmes.

Domaine : Structures de Données

Tarif : 67.00 €

[Show More](#) [Choisir les enfants](#)

APPRENDRE LE DÉVELOPPEMENT WEB
HTML, CSS, JS

Développement Web avec HTML, CSS et JavaScript

Les participants apprendront à créer des sites web interactifs en utilisant HTML, CSS pour le design, et JavaScript pour la fonctionnalité.

Objectifs : Comprendre les bases du développement web, créer des pages web interactives, apprendre à utiliser les technologies front-end.

Domaine : Programmation

Tarif : 100.50 €

[Show More](#) [Choisir les enfants](#)

Si vous avez terminé l'ajout de tous les enfants que vous souhaitez inscrire dans les activités, vous pouvez envoyer la demande en cliquant sur le bouton ci-dessous.

[Envoyer la demande](#)

L'utilisateur va donc diriger vers un composant submitrequest, ce composant permet aux utilisateurs de finaliser et de soumettre leur demande d'inscription pour diverses activités. Après avoir sélectionné les activités et les enfants à inscrire, l'utilisateur peut revoir un récapitulatif de ses choix.

Chaque activité sélectionnée est affichée avec des détails sur l'offre, l'activité, l'enfant choisi et l'horaire correspondant. Le composant propose également des options de packs supplémentaires que l'utilisateur peut sélectionner avant de soumettre sa demande.

Lorsque l'utilisateur clique sur le bouton « Soumettre la Demande », les informations sont envoyées au backend pour traitement et la sélection est enregistrée et nettoyée du stockage local (Car tous les choix et les sélections effectués par l'utilisateur sont stockés dans le local storage, cela permet d'envoyer toutes les inscriptions de leurs enfants en une seule requête au backend.).

Récapitulatif de la Demande

Offre : Programmation entre 15 et 17 ans

Activité : Introduction à Python
Enfant : Marwan mobader
L'horaire : dimanche,10:00:00,12:30:00

Offre : Programmation entre 15 et 17 ans

Activité : Introduction à Python
Enfant : amir kilbi
L'horaire : dimanche,12:30:00,15:00:00

Offre : Programmation entre 15 et 17 ans

Activité : Introduction à Python
Enfant : rawan Vernas

<http://localhost:8081/submitrequest>

Offre : Programmation entre 15 et 17 ans

Activité : Introduction à Python
Enfant : rawan Vernas
L'horaire : mardi,20:00:00,22:30:00

Offre : Programmation entre 15 et 17 ans

Activité : Introduction à Python
Enfant : Marwan mobader
L'horaire : dimanche,10:00:00,12:30:00

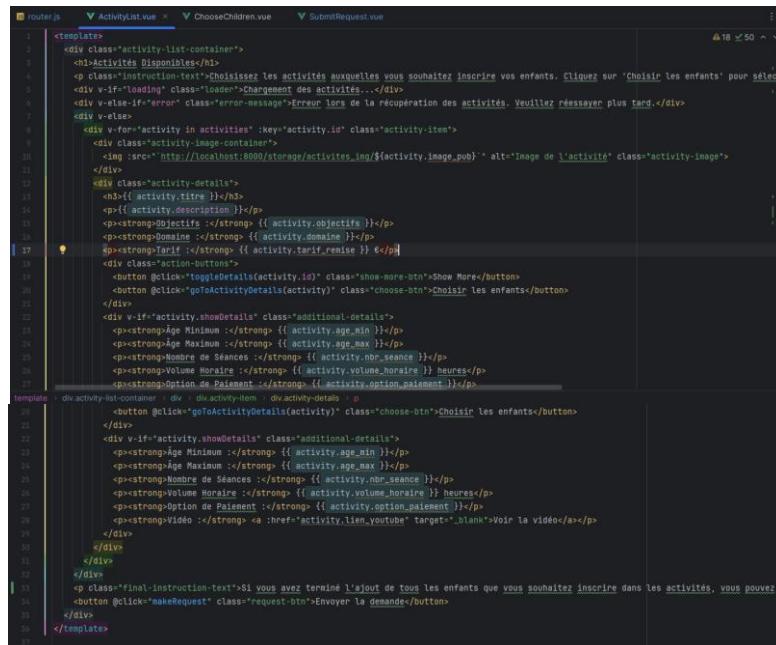
Choisissez un Pack (Optionnel):

Pack Enfant Pack Atelier

Si vous avez terminé l'ajout de tous les enfants que vous voulez dans les activités, vous pouvez envoyer la demande.

Soumettre la Demande

- **Le code de composant activiter offre :**



```

<template>
  <div class="activity-list-container">
    <h1>Activités Disponibles</h1>
    <p>Choisissez les activités auxquelles vous souhaitez inscrire vos enfants. Cliquez sur 'Choisir les enfants' pour sélectionner les enfants que vous souhaitez inscrire dans cette offre. Si vous avez terminé l'ajout de tous les enfants que vous voulez dans les activités, vous pouvez envoyer la demande.</p>
    <div v-if="loading" class="loader">Chargement des activités...</div>
    <div v-else-if="error" class="error-message">Erreur lors de la récupération des activités. Veuillez réessayer plus tard.</div>
    <div v-else>
      <div v-for="activity in activities" key="activity.id" class="activity-item">
        <div class="activity-image-container">
          
        </div>
        <div class="activity-details">
          <h3>${activity.title}</h3>
          <p>${activity.description}</p>
          <p><strong>Objectifs :</strong> ${activity.objectifs}</p>
          <p><strong>Domaine :</strong> ${activity.domaine}</p>
          <p><strong>Tarif :</strong> ${activity.tarif_remise}</p>
        </div>
        <div class="action-buttons">
          <button @click="toggleDetails(activity.id)">Show More</button>
          <button @click="goToActivityDetails(activity)">Choisir les enfants</button>
        </div>
        <div v-if="activity.showDetails" class="additional-details">
          <p><strong>Age Minimum :</strong> ${activity.age_min}</p>
          <p><strong>Age Maximum :</strong> ${activity.age_max}</p>
          <p><strong>Nombre de Séances :</strong> ${activity.nbr_seance}</p>
          <p><strong>Volume Horaire :</strong> ${activity.volume_horaire} heures</p>
          <p><strong>Option de Paiement :</strong> ${activity.option_paiement}</p>
        </div>
      </div>
    </div>
    <div v-if="activity.showDetails" class="choose-children">
      <div>
        <button @click="goToActivityDetails(activity)">Choisir les enfants</button>
      </div>
      <div v-if="activity.showDetails" class="additional-details">
        <p><strong>Age Minimum :</strong> ${activity.age_min}</p>
        <p><strong>Age Maximum :</strong> ${activity.age_max}</p>
        <p><strong>Nombre de Séances :</strong> ${activity.nbr_seance}</p>
        <p><strong>Volume Horaire :</strong> ${activity.volume_horaire} heures</p>
        <p><strong>Option de Paiement :</strong> ${activity.option_paiement}</p>
        <p><strong>Vidéo :</strong> <a href="${activity.lien_youtube}" target="_blank">Voir la vidéo</a></p>
      </div>
    </div>
  </div>
  <p>Si vous avez terminé l'ajout de tous les enfants que vous souhaitez inscrire dans les activités, vous pouvez envoyer la demande.</p>
  <button @click="makeRequest" class="request-btn">Envoyer la demande</button>
</div>

```

- **La partie script :**

```

  </script>
  <script>
    methods: {
      async fetchActivities() {
        const offerId = this.$route.params.offerId;
        const offerFilter = this.$route.query.offerFilter;
        try {
          alert(offerFilter);
          const response = await axios.get(`http://localhost:8000/api/showOffer/activities/all/${offerId}`);
          this.activities = response.data.map(activity => ({ ...activity, showDetails: false }));
          this.loading = false;
        } catch (error) {
          console.error('Erreur lors de la récupération des activités:', error);
          this.loading = false;
          this.error = true;
        }
      },
      toggleDetails(activityId) {
        const activity = this.activities.find(act => act.id === activityId);
        if (activity) {
          activity.showDetails = !activity.showDetails;
        }
      },
    },
  </script>

```

- le code de composant select horaire :

```

1 <template>
2   <div class="select-schedule-container">
3     <h1>Les horaires disponibles</h1>
4     <p class="instruction-text">Choisissez une seule horaire pour : <span class="child-name">{{ childName }}</span></p>
5     <div v-if="loading">Chargement des horaires...</div>
6     <div v-else-if="error" class="error-message">Erreur lors de la récupération des horaires. Veuillez réessayer plus tard.</div>
7     <div v-else>
8       <div v-for="schedule in schedules" :key="schedule.id" class="schedule-card">
9         <label :for="schedule-${schedule.id}" class="schedule-label">
10          <input type="checkbox" :id="`schedule-${schedule.id}`" v-model="selectedSchedules" :value="`${{schedule.jour}},${{schedule.heure_debut}},${{schedule.heure_fin}}`" />
11          <div class="schedule-details">
12            <span class="schedule-day">{{ schedule.jour }}</span>
13            <span class="schedule-time">{{ schedule.heure_debut }} - {{ schedule.heure_fin }}</span>
14            <div class="schedule-info">
15              <span><strong>Eff. Min:</strong> {{ schedule.eff_min }}</span>
16              <span><strong>Eff. Max:</strong> {{ schedule.eff_max }}</span>
17              <span><strong>Places Restantes:</strong> {{ schedule.nbr_place_restant }}</span>
18            </div>
19          </div>
20        </label>
21      </div>
22    </div>
23    <button @click="submitSchedules" class="submit-btn">Terminer</button>
24  </div>
25 </template>

```

le stockage de l'inscription d'un enfant dans le local storage :

```

61   submitSchedules() {
62     if (this.selectedSchedules.length !== 1) {
63       alert('Veuillez sélectionner exactement une horaire.');
64       return;
65     }
66
67     const selectedSchedule = this.selectedSchedules[0];
68
69     const selectedActivity = {
70       offerId: this.offerId,
71       offerTitre: this.offerTitre,
72       activite_offre_id: this.activityId,
73       activityTitre: this.activityTitre,
74       enfant_id: this.childId,
75       childName: this.childName,
76       horaire: selectedSchedule
77     };
78     console.log(selectedActivity);
79
80     // Stocker l'activité sélectionnée dans le localStorage
81     let activities = JSON.parse(localStorage.getItem('selectedActivities')) || [];
82     activities.push(selectedActivity);
83     localStorage.setItem('selectedActivities', JSON.stringify(activities));
84

```

- le code de la récapitulatif:

```

111  export default {
112    name: 'SubmitRequest',
113    data() {
114      return {
115        selectedActivities: JSON.parse(localStorage.getItem('selectedActivities')) || [],
116        selectedPack: JSON.parse(localStorage.getItem('selectedPack')) || [] // Ajout du stockage des packs
117      };
118    },
119    methods: {
120      async submitRequest() {
121        try {
122          const selectedPackString = this.selectedPack.join(',');
123          alert(selectedPackString);
124
125          const response = await axios.post(`http://localhost:8000/api/create/demande/${selectedPackString}`, {
126            activities: this.selectedActivities
127          });
128
129          alert(response.data.name);
130          localStorage.removeItem('selectedActivities'); // Nettoyer le localStorage après soumission
131          localStorage.removeItem('selectedPack'); // Nettoyer le stockage des packs après soumission
132          this.$router.push('/afficherpage');
133        } catch (error) {
134

```

2. Interface de l'administrateur :

Dans le cadre du développement d'une interface d'administration pour notre application web, j'ai été chargé de concevoir et d'implémenter plusieurs composants clés en utilisant Vue.js. Cette tâche a impliqué la création et la gestion d'interfaces utilisateur spécifiques à l'administration, permettant une interaction fluide et efficace avec la base de données de l'application.

The screenshot shows the Admin interface. On the left is a sidebar with a purple header labeled 'PROFIL' and a red 'Déconnexion' button. Below the header, there are four menu items: 'Mes Offres', 'Mes activités', 'Demandes', and 'Animateurs', each with a dropdown arrow. Under 'Mes Offres', there is a card for 'animante' with a photo of a girl wearing headphones, the title 'animante', the description 'chimie à travers des expériences amusantes....', and a 'détails' button. The other three menu items have similar cards below them. The main content area is titled 'Offres Top' and displays three cards for 'Robotique avancée', 'Robotique avancée pour les adolescents....', and 'Programmation entre 15 et 17 ans'. Each card has a photo of a girl wearing headphones, the title, a brief description, and a 'détails' button.

Les composants développés, comme illustrés dans la capture d'écran fournie, incluent des pages pour la gestion des activités, des offres, des animateurs, ainsi que des listes détaillées pour une administration facile. Chaque composant a été conçu avec un souci de clarté et de fonctionnalité, assurant ainsi que les administrateurs de l'application peuvent gérer les contenus de manière intuitive et efficace.

Ces développements s'inscrivent dans une démarche plus large visant à offrir une expérience utilisateur optimale pour les gestionnaires de l'application, en leur fournissant tous les outils nécessaires pour mener à bien leurs tâches administratives sans complications. Ce rapport détaillera les spécificités de chaque composant, les choix techniques réalisés, ainsi que les interactions entre les différents éléments de l'interface.

The screenshot shows the code structure of the AdminPage.vue component. It is a tree view of files under the 'ADMIN' folder. The files listed are: ActivitylistAdmin.vue, AdminPage.vue (which is highlighted in blue), AjouterActivite.vue, AjouterOffre.vue, Animateur.vue, ListeEnfantActivites.vue, OfferDetailsAdmin.vue, OffersListAdmin.vue, sidebar.vue, and TopOffers.vue.

La page AdminPage intègre une barre de navigation pour gérer la visibilité d'une barre latérale et la déconnexion de l'utilisateur. Le composant <Sidebar> est animé et réactif, contrôlé par isSidebarOpen, tandis que <TopOffers> met en avant les offres clés. Cette structure assure une navigation fluide et une gestion efficace des fonctionnalités administratives.

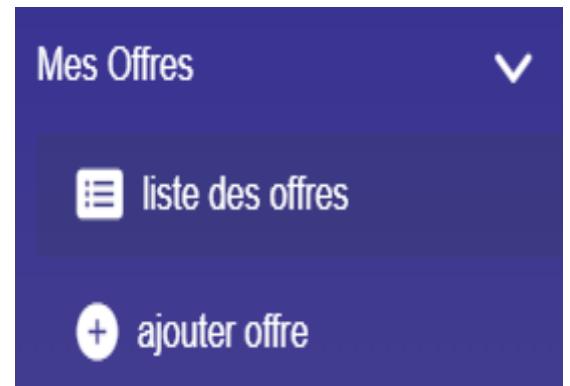
```

<template>
  <div class="main-container">
    <!-- En-tête -->
    <nav class="navbar">
      <div class="menu-icon">
        <button @click="toggleSidebar" class="btn-custom">
          <i class="fas fa-bars"></i> BIENVENUE
        </button>
      </div>
      <div class="logout-btn">
        <button @click="logout" class="btn-custom btn-danger">Déconnexion</button>
      </div>
    </nav>
    <transition name="slide">
      <sidebar v-if="isSidebarOpen"></sidebar>
    </transition>
    <div class="main-container" :class="{ 'sidebar-open': isSidebarOpen }">
      <TopOffers></TopOffers>
    </div>
  </div>
</template>

<script>
import Sidebar from '@/components/ADMIN/sidebar.vue';
import TopOffers from '@/components/ADMIN/TopOffers.vue';
export default {
  name: 'AdminPage',
  components: {
    Sidebar,
    TopOffers
  },
  data() {
    return {
      isSidebarOpen: false,
      showProfileMenu: false
    };
  },
  methods: {
    toggleSidebar() {
      this.isSidebarOpen = !this.isSidebarOpen;
    },
    logout() {
      this.$router.push('/signin');
    }
  }
};
</script>

```

La section "Mes Offres" de la barre latérale permet aux utilisateurs d'accéder à une liste des offres et de rajouter de nouvelles offres via des options clairement démarquées. Cette fonctionnalité améliore la gestion des offres en offrant un accès rapide et organisé



c

Toutes les Offres

Rechercher des offres...



Calcul Différentiel

Cours de calcul différentiel pour les lycéens....

[détails](#)



Sécurité Informatique

Introduction à la sécurité informatique et aux techniques de protection des données....

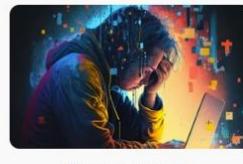
[détails](#)



Robotique avancée

Cours de robotique avancée pour les adolescents....

[détails](#)



Impression 3D

Apprentissage de l'impression 3D et du design....

[détails](#)



Les deux captures d'écran illustrent les fonctionnalités clés de gestion des offres éducatives dans une interface d'administration. La première image montre la page "Toutes les Offres", où les offres actuelles sont affichées de manière attrayante avec des options pour plus de détails, facilitant la navigation et la sélection par les utilisateurs. La seconde image dépeint le formulaire "Ajouter une Offre", permettant aux administrateurs d'insérer de nouvelles offres avec des détails tels que les dates et les remises. Ensemble, ces interfaces supportent une gestion efficace et organisée des offres éducatives, en améliorant l'expérience utilisateur et en simplifiant les tâches administratives.

Cette capture d'écran présente une interface de la section "Activités Disponibles" où diverses offres éducatives sont listées pour l'administration. Chaque offre affichée inclut un titre et une description, permettant un aperçu rapide de l'activité proposée. Trois actions

Ajouter une Offre

programmation web

22/06/2024



07/07/2024



Remise (%)

Description

[Ajouter](#)

[Afficher les activités](#)

[Ajouter une activité](#)

[Supprimer l'offre](#)

ACTIVITÉS DISPONIBLES

Titre : Ab non in ab dicta consequuntur quam.

Description: Sit minus sit cum cumque. Cum ipsum sint deleniti consequatur cumque optio libero. Ut quia pariatur deleniti sequi nihil. Qui minus est alias autem vel accusantium dolore.

[Ajouter à l'offre](#)

Titre : Aspernatur accusamus quam delectus laudantium quo et.

Description: Nemo dolorum porro mollitia voluptatem odio qui. Consequuntur laboriosam sed impedit nulla a recusandae aut. Ratione aut non quas et rerum autem.

[Ajouter à l'offre](#)

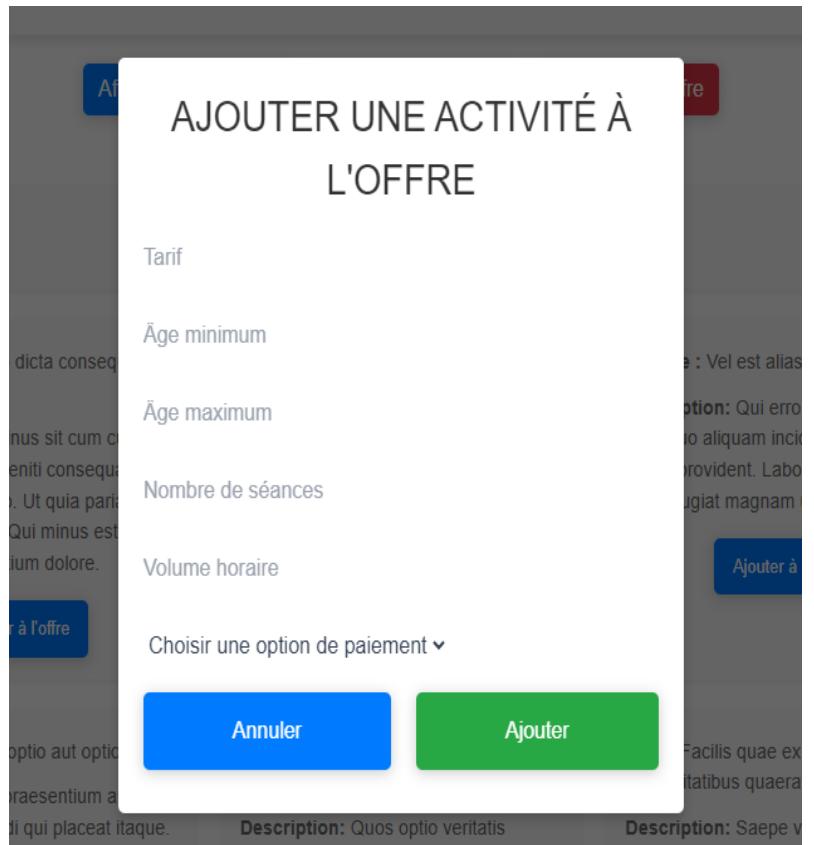
Titre : Vel est alias ipsum ratione.

Description: Qui error sunt maiores. Eum quo aliquam incidunt explicabo officia provident. Laborum adipisci atque fugiat magnam ut.

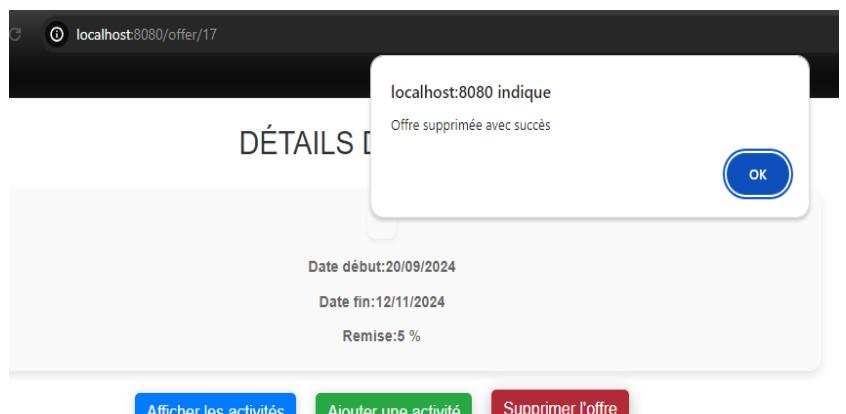
[Ajouter à l'offre](#)

principales sont disponibles en haut de la page : "Afficher les activités", "Ajouter une activité", et "Supprimer l'offre", facilitant ainsi la navigation et la gestion des activités. De plus, chaque offre a un bouton "Ajouter à l'offre", pour intégrer ces activités dans des programmes ou des promotions spécifiques. Cette interface est conçue pour aider les administrateurs à gérer efficacement les offres éducatives, en permettant une manipulation simple et directe des informations.

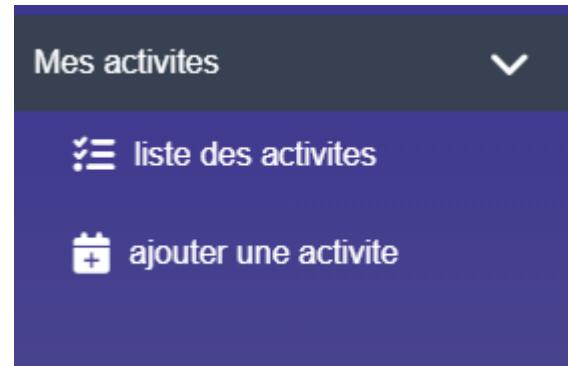
Le formulaire "AJOUTER UNE ACTIVITÉ À L'OFFRE" permet de spécifier les détails financiers et démographiques d'une activité, tels que le tarif, l'âge minimum et maximum, le nombre de séances, et le volume horaire. Il offre également une option pour sélectionner le mode de paiement, optimisant ainsi la personnalisation et la gestion des activités proposées.



la suppression d'une offre :



La section "Mes activités" offre des options pour afficher la liste des activités existantes ou pour en ajouter de nouvelles, facilitant la gestion des programmes.



Ce script Vue.js gère la soumission d'une nouvelle activité via un formulaire, utilisant Axios pour envoyer les données à une API. En cas de succès, il nettoie le formulaire et redirige vers AdminPage, tandis qu'en cas d'erreur, il affiche un message détaillé pour aider au débogage. Le code montre une utilisation efficace de la gestion des états et des réponses API pour améliorer l'expérience utilisateur.

```
<script>
export default {
  methods: {
    submitForm() {
      const formData = new FormData();
      formData.append('image_pub', this.image_pub);
    }

    axios.post('http://localhost:8000/api/create/activity/', formData)
      .then(response => {
        console.log('Activité ajoutée:', response.data);
        this.message = 'Activité ajoutée avec succès!';
        this.activite = {
          titre: '',
          description: '',
          objectifs: '',
          domaine: '',
          lien_youtube: ''
        };
        this.image_pub = null;
        this.$refs.image.value = '';
        this.$router.push('/AdminPage');
      })
      .catch(error => {
        console.error('Erreur lors de l\'ajout de l\'activité:', error.response.data);
        this.message = 'Erreur lors de l\'ajout de l\'activité : ' + error.response.data.detail;
      });
    }
  };
</script>
```

Le formulaire "Ajouter une Activité" permet aux utilisateurs de saisir des informations détaillées sur une nouvelle activité, y compris un titre, une description, des objectifs, un domaine, un fichier optionnel et un lien YouTube. Cette interface facilite la contribution et l'organisation des contenus éducatifs.

Ajouter une Activité

Titre de l'activité

Description de l'activité

Objectifs de l'activité

Domaine de l'activité

Aucun fichier choisi

Lien YouTube (optionnel)

Ajouter

Cette interface utilisateur présente un tableau listant des enseignants avec des colonnes pour le nom, l'email, le domaine d'expertise, et des boutons pour afficher les horaires occupés et libres. Elle est conçue pour permettre une gestion efficace des disponibilités du personnel éducatif, facilitant l'organisation des plannings et la communication.

Nom	Email	Domaine	Horaires	Horaires Occupés	Horaires Libres
tahiri karim	annabell32@example.net	informatique	<input type="button" value="Afficher Horaires"/>	<input type="button" value="Afficher Horaires Occupés"/>	<input type="button" value="Afficher Horaires disponible"/>
Benani mohammed	nmayert@example.org	physique	<input type="button" value="Afficher Horaires"/>	<input type="button" value="Afficher Horaires Occupés"/>	<input type="button" value="Afficher Horaires disponible"/>

Cette fenêtre modale affiche les horaires d'un enseignant, spécifiant les heures de début et de fin pour un jour donné. Le bouton "Fermer" permet de quitter la vue des horaires, rendant l'interface pratique pour consulter rapidement les disponibilités spécifiques.

Horaires

Jour	Heure de début	Heure de fin
Thursday	08:39:07	00:25:08

[Fermer](#)

[Afficher Horaires Occupés](#)

[Afficher Horaires Non Occupés](#)

Le formulaire "Ajouter Animateur" permet aux administrateurs d'enregistrer de nouveaux animateurs en fournissant des informations essentielles telles que le nom, l'email, le domaine d'expertise et un mot de passe. Les boutons "Enregistrer" et "Annuler" offrent des options simples pour confirmer ou abandonner l'ajout

[Ajouter Animateur](#)

[Enregistrer](#) [Annuler](#)

➤ Autre composant :

The screenshot shows a contact form titled "Contactez-nous". The form includes fields for "Nom" (Name), "Email", and "Message", each with a corresponding input box. A blue "Envoyer" (Send) button is located at the bottom right of the form area. The background features a purple header bar with a logo and navigation links: Accueil, À propos de nous, FAQ, Contact, and a user icon.

The screenshot shows a FAQ page titled "Foire aux Questions (FAQ)". It displays two collapsed sections: "Comment créer un compte ?" (How to create an account?) and "Comment réinitialiser mon mot de passe ?" (How to reset my password?). Each section has a small explanatory text below it. The browser address bar shows the URL "http://localhost:9081/FAQ".

← → ⌛ http://localhost:8081/AproposNous



Accueil À propos de nous FAQ Contact

À propos de nous

Bienvenue sur notre plateforme dédiée à l'épanouissement et au développement personnel des enfants.

Notre Mission

Notre mission est de fournir un environnement sûr et stimulant où les enfants peuvent apprendre, jouer et grandir. Nous croyons que chaque enfant a un potentiel unique et nous efforçons de les aider à le réaliser pleinement.

Nos Valeurs

- **Éducation:** Offrir des activités éducatives de haute qualité.
- **Sécurité:** Assurer un environnement sûr pour tous les enfants.
- **Inclusivité:** Accueillir tous les enfants, quelles que soient leurs origines ou leurs capacités.
- **Innovation:** Utiliser des méthodes et des outils innovants pour l'apprentissage.

Notre Équipe

Notre équipe est composée de professionnels passionnés par l'éducation et le