Module M4103C - *SportRegister*

—

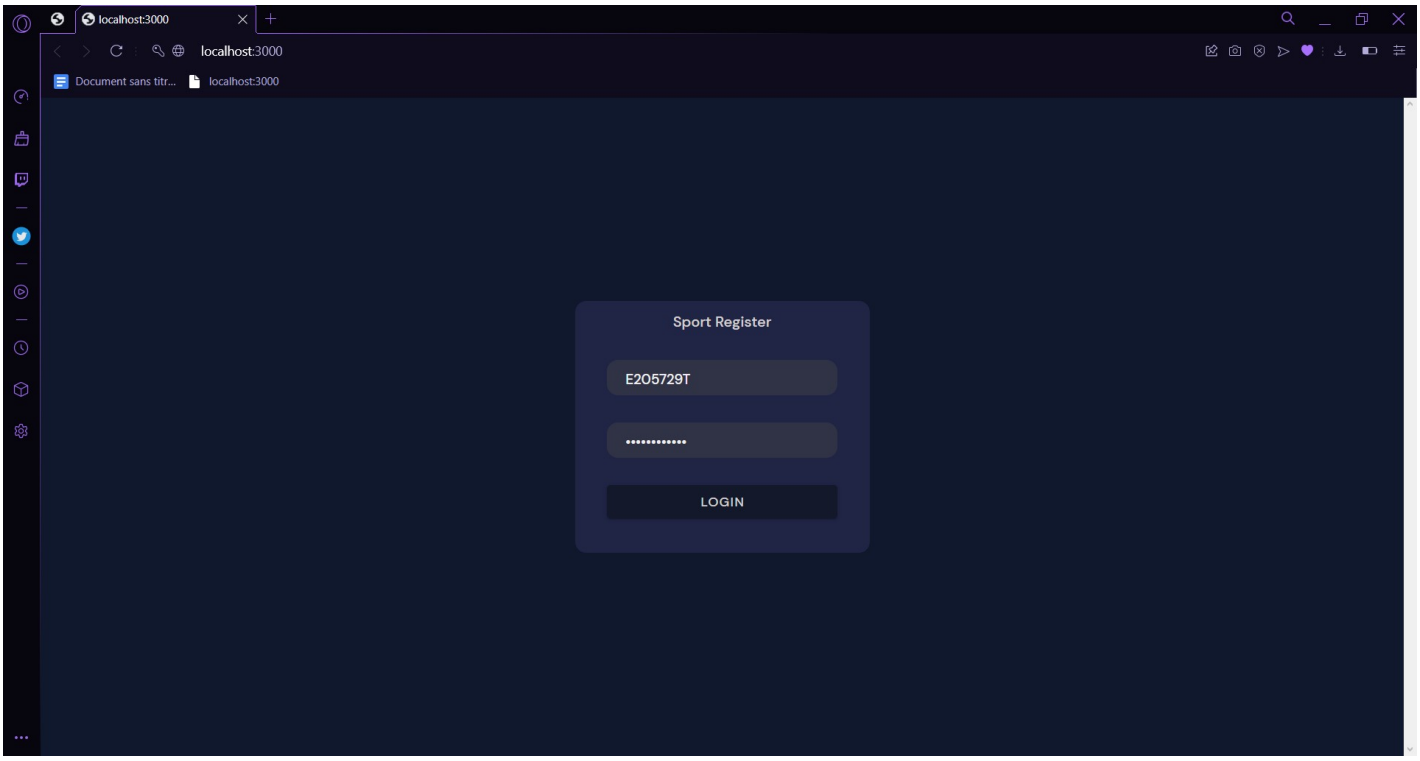
Rochard Léo & Naquin Elouan
Groupe 4

Introduction

Le site SUAPS de l'université de Nantes permet de s'inscrire à des sports en choisissant une plage horaire. Mais une contrainte nous a bloqué dans cette démarche : quand un sport est complet, il faut attendre que quelqu'un désiste sa place. Le site étant lent, et nécessite une reconnexion à chaque réactualisation pour voir si le sport n'est plus complet.
SportRegister est une application web qui permet de faciliter l'inscription au sport à l'université de Nantes.

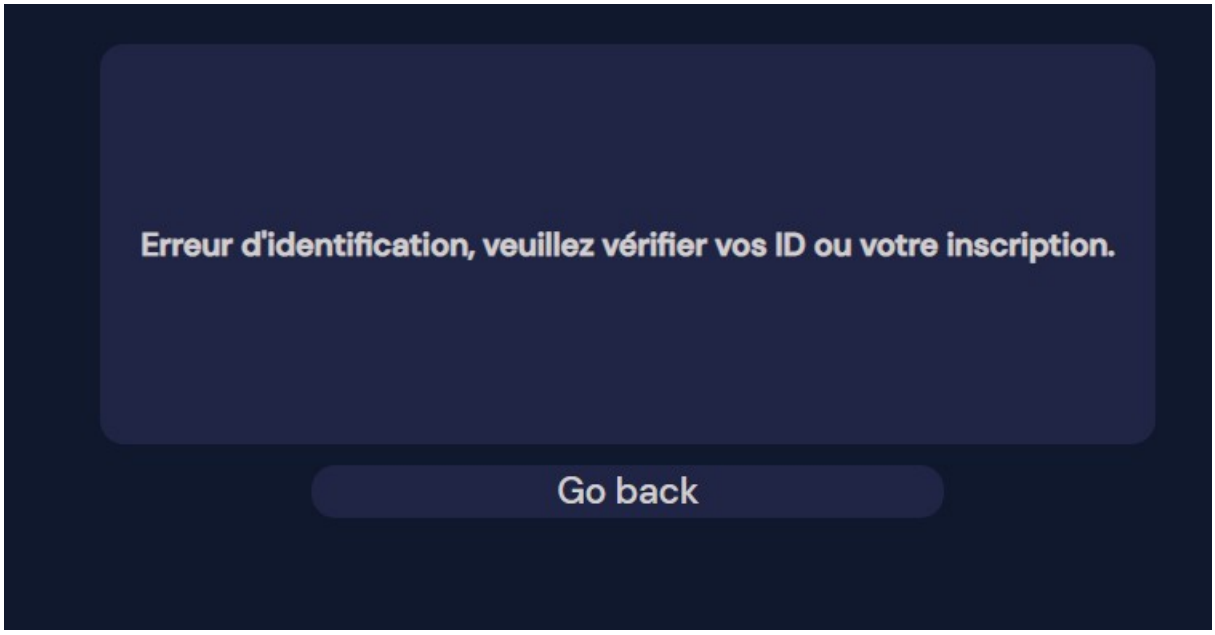
On utilise l'API de l'université de Nantes afin d'obtenir une interface qui ressemble à celle du site de base tout en ajoutant notre fonctionnalité. En effet, il est possible grâce à notre application de se mettre en attente pour obtenir une place lorsque la séance sélectionnée est « complète ». De ce fait, on aura une place automatiquement.
Pour utiliser notre application, il suffit de renseigner ses identifiants de l'université de Nantes puis il faut sélectionner la ou les séances que l'on souhaite réserver.

Vision fonctionnelle



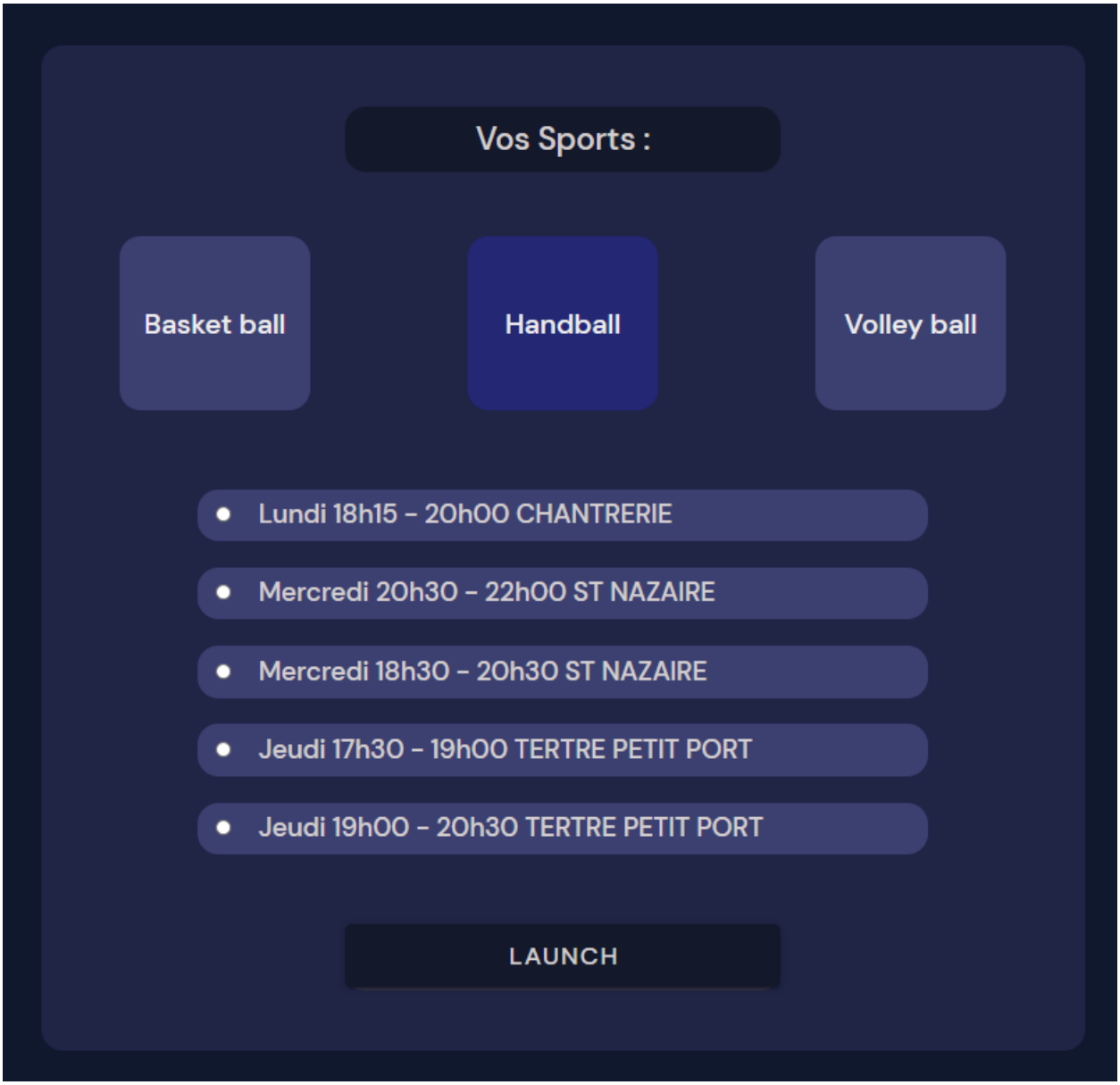
Sur la page d'accueil on trouve un formulaire nécessitant les identifiants de connexion de l'université,
On peut ensuite soumettre celui-ci avec le bouton « Login ».

Un mauvais identifiant ou mot de passe renvoie vers une page d’erreur :



En effet on précise ici de vérifier son inscription, car avec même si vos identifiants sont bons, l’API vous renverra un code erreur puisqu’il faut être inscrit au suaps.

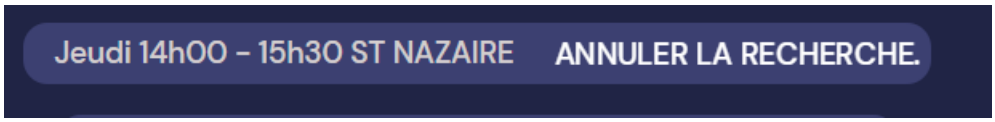
Ici, on retrouve la div qui apparaît lorsque nous sommes connectés. Avec les sports sélectionnés par l’utilisateur. Lorsque nous cliquons sur un des sports, on observe la liste des créneaux disponibles.



Il nous suffit ensuite de choisir le créneau que l’on veut, et d’appuyer sur « launch »,

En cas de réussite, nous obtenons un tag inscrit :

En cas de recherche car les places sont complètes, nous obtenons un message de recherche. Ce message indique que l'application s'occupe de chercher une place pour ce créneau toutes les X minutes :



Si on veut arrêter la recherche ou se désinscrire du créneau, il suffit de cliquer sur le tag:

Vision technique

Pour notre projet, nous avons choisi d'utiliser Nuxt.JS, il s'agit d'un framework basé sur VueJS et NodeJS. Nous avons fait ce choix car Nuxt 3 est nouveau (en version alpha actuellement) et nous désirions nous familiariser avec celui-ci. De plus, ayant étudié VueJS en cours, nous voulions utiliser cette technologie.

Nous avons également utilisé le framework CSS Vuetify pour pouvoir ajouter des spinners, des effets sur les boutons etc, ce qui nous as permis d'éviter trop de ligne de CSS.

NUXT 3

VUE JS

VUETIFY 3

Pour gérer notre projet, nous avons utilisé Gitlab pour pouvoir travailler efficacement. En ce qui concerne notre IDE, il s'agissait de WebStorm, l'outil proposé par JetBrains.

Pour développer notre application, nous avons observé les différentes requêtes réalisées par le navigateur lorsque nous nous inscrivions à un sport via l'outil proposé par l'IUT, car il n'y a pas de documentation pour l'api de l'université.

Voici d'ailleurs l'exemple d'une requête que nous renvoie l'API lors d'une recherche de sport :


```
{code: "E205729T", civility: "M.", name: "ROCHARD", firstname: "Léo",…}
  birthDate: "16/01/2002"
  civility: "M."
  code: "E205729T"
  composante: "IUT DE NANTES"
  departement: "DEPT. INFORMATIQUE"
  email: "leo.rochard@etu.univ-nantes.fr"
  estBoursier: false
  firstname: "Léo"
  montantPaiement: 25
  name: "ROCHARD"
  paiementEffectue: true
▼ sports: [{code: 23,…}, {code: 24,…}, {code: 27,…}]
  ▼ 0: {code: 23,…}
    ▶ categorie: {code: 32, nom: "Sports Collectifs", picto: "ca-sports-collectifs", image: "collectifs.jpg",…}
      code: 23
    ▶ creneaux: [{site: "TERTRE PETIT PORT", places: 60, code: 529, jour: "Mardi", encadrant: "Gigi MEYRAT",…},…]
      description: "Les jours et horaires sont susceptibles de changer en fonction des périodes de l'année ( voir \" inf
      nom: "Basket ball"
      registrations: []
  ▶ 1: {code: 24,…}
  ▶ 2: {code: 27,…}
  typePersonne: "GE"
```


On voit une information intéressante qui est celle des sports auxquels nous sommes abonnés.
La zone « créneaux » sera l'information qui nous intéresse. On remarque également que le « code » est l'identifiant du sport.

Nous avons donc compris comment s'inscrire et se désinscrire. Le plus compliqué selon nous était de savoir comment récupérer les sports auxquels nous étions déjà inscrits puisque notre fonctionnalité ajoutée nécessite que l'on garde une liste des sports inscrits.

Ceci s'est fait grâce aux informations récupérées lorsque l'on se connecte via l'API de l'université, on obtient un fichier JSON qui nous fournit les informations souhaitées.

Une fois celles-ci récupérées, nous avions accès aux données de chaque sports, leurs horaires, lieux etc..

Nous pouvions donc effectuer la requête nécessaire avec les informations que nous avions.

Description du code :

components/ : Contient tout nos composants vue :

-Loading, qui est le composants pour faire l'effet css « flip » de la recherche

-Login, qui est le composant pour entrer les identifiants de connexion

-Sport, qui est l'activité principale, s'inscrire, désinscrire,

L'utilisation du framework Vuetify est utilisé sur les boutons de lancement pour ajouter un effet, et également pour le spinner de chargement d'une page.

composables/tools.ts : Contient toute nos fonctions détaillés liées à l'API, C'est un « proxy » afin de faciliter le lien avec l'API, nous savons que ce n'est pas recommandé, cependant, le serveur lancé par NuxtJs nécessitait l'activation de certains paramètres pour pouvoir fetch des informations en front. C'est pourquoi on a choisi de créer notre propre api qui s'occupait de fetch.

server/api : Contient nos outils pour intéragir, dans chacun de ces fichiers on fait appel à tools.ts. Chaque fichier contenu dans api correspond à une route, on retrouve donc « update , login, register et unregister » qui sont nécessaires à notre projet.

- Login s'occupe de nous login et donc de nous renvoyer une session, cette session nous permettra d'effectuer toutes les requêtes que l'on souhaite faire.
- Update s'occupe de réaliser la requête permettant d'avoir les informations sur un utilisateur, celle-ci nous permet de mettre à jour la liste des sports choisis et des créneaux auxquels nous sommes inscris, elle est appelée toutes les 15secondes.
- Register s'occupe quant à elle de nous inscrire à un créneau, si l'inscription n'a pas fonctionné, le créneau rentre dans une liste d'attente et l'inscription est retentée toutes les 2minutes jusqu'à ce que la personne soit inscrite.
- Unregister permet à l'utilisateur de se désinscrire d'une séance qu'il avait réservé.

Conclusion

Nous sommes satisfaits du travail que nous avons réalisé, en effet, notre application correspond à nos attentes.

Grâce à ce projet, nous avons pu renforcer nos connaissances en développement web, tout en incluant ce que nous avions étudié lors de nos séances de TD.
Le fait de manipuler Nuxt.JS nous a aidé en ce qui concerne l'organisation du code.

Ce que nous avons apprécié c'était la liberté que nous avions pour ce projet. Le fait de devoir et pouvoir réaliser une application qui réponde à nos besoins est toujours plus motivant qu'un travail imposé.