COMPTE-RENDU D'ACTIVITE

Projet Gestion des vacataires - Groupe 1

FLORIO Michele

Table des matières

1. Le projet

Depuis la réforme du BUT, le nombre de vacataires employés par l'IUT a grandement augmenté. A tel point qu'il a été décidé de définir un responsable des vacataires qui aura pour rôle de les affecter et désaffecter. Pour cela, ce responsable a besoin d'une interface web lui permettant de facilement gérer les vacataires mais aussi les différents cours affectables. Cet outil, à l'heure actuelle inexistant, sera créé par des élèves de troisième année de la filière informatique de l'établissement dans le cadre des heures de SAE.

Deux groupes projets ont été formés afin de développer le site de gestion des vacataires mais seul le meilleur produit sera retenu. Ces groupes sont évolutifs ; en effet, dans un premier temps formés uniquement d'élèves en formation initiale, des équipes de « renforcement » composées d'alternants viendront leur prêter main-forte.

J'ai rejoint l'équipe 1 du projet « gestion des vacataires » composée d'Enzo MANCINI, Bastien BALMES et Christopher MARIE-ANGELIQUE avec mes camarades alternants Bryce FUERTES, Alex JOLAS, Marco VALLE et Victor THOMPSON. A notre arrivée dans le groupe, le site web était assez bien avancé : il y avait une page pour la gestion des vacataires avec la quasi-totalité des interactions de fonctionnelles (création, modification, suppression) et une page pour la gestion des cours en construction. L'interface était plutôt soignée et reprenait bien la charte graphique de l'IUT. Le site était séparé en deux parties, le frontend codé en JavaScript avec le framework Angular et une API programmée avec Node.JS.

Quelques tâches étaient prévues pour les alternants, notamment celle qui m'a été assignée : le filtrage des données affichées sur les différentes pages du projet. Je m'y suis tout de suite attaqué, enfin juste après m'être renseigné sur le framework Angular (ne l'ayant jamais utilisé).

2. Réalisations personnelles

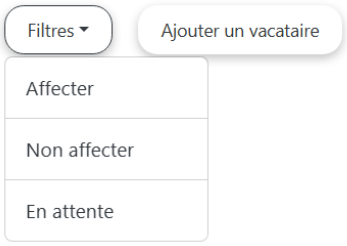
2.1 Filtres

Il s'agissait de ma tâche principale, celle qui m'a été fournie par les initiaux le jour de notre arrivée. A terme, les deux pages du site seront pleines d'informations (il n'y a qu'à voir la section « modules » de flopEDT pour se rendre compte plus ou moins du nombre de cours qu'il y aura), il faut donc pouvoir les filtrer afin de trouver facilement ce que l'on cherche. Au départ, je ne devais m'occuper que des filtres de la page des vacataires mais, au vu de la similarité entre les deux tâches, j'ai finalement aussi codé les filtres de la page des cours.

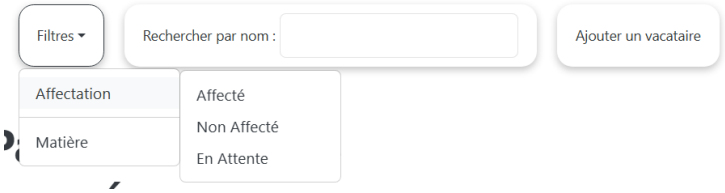
Issues : #19 / #20

Après avoir créé une branche git (« dev-filtres »), j'ai d'abord analysé ce qui était déjà fait pour déterminer, d'une part sur quelles propriétés il était intéressant d'appliquer les filtres, d'autre part la meilleure manière de mettre ces filtres en place.

Pour la partie vacataire, un bouton filtre était déjà mis en place avec trois autres boutons dans un menu déroulant (sans interaction pour le moment) ; la page cours elle en revanche ne contenait rien. J'ai alors décidé pour l'affichage de me baser sur ce composant déjà existant mais d'ajouter des sous-menus déroulants pour classer les différents filtres par catégories et une barre de recherche.



Page vacataire 1 – Prototype initial des filtres



Page vacataire 2 - Filtres définitif

Pour les vacataires, il a été décidé de mettre en place un filtrage par statut (affecté, non affecté, en attente) et par matière d'affectation ; la barre de recherche, elle, permet de récupérer un vacataire par nom et/ou prénom. Pour les cours, filtrage par matière et par département concerné et recherche par nom complet ou réduit du cours.

Les filtres à présent établis, il ne reste plus qu'à décider de quelle manière les faire fonctionner. Après réflexion, j'ai choisi d'appliquer le filtrage côté frontend plutôt que backend pour deux raisons principales : éviter de faire de multiples requêtes et le nombre de données qui ne sera jamais assez élevé pour causer des ralentissements à l'utilisateur. Pour cela, lors de la récupération de données à l'ouverture de la page, je récupère dans la liste de vacataires ou de cours les informations qui m'intéressent (liste des différents départements et matières) pour peupler les menus déroulants de filtre. Ensuite, il suffit de gérer les filtres actifs en les stockant dans les paramètres d'URL (ex : « /cours?matiere=developpement ») et de créer une fonction qui vient vérifier si le vacataire (ou le cours selon la page) rempli les filtres actifs.

```
<ng-container *ngFor="let cours of modules">
    <div *ngIf="isInFilter(cours)" class="cours-card">
```

Si isInFilter renvoi faux, la carte ne sera pas affichée

Le composant est alors mis en place et 100% fonctionnel, cependant quelques ajouts pour l'expérience utilisateur étaient indispensables ; ainsi, une pastille indique la valeur sélectionnée pour une catégorie de filtre donnée et il est possible de supprimer un filtre actif.

2.2 Optimisations

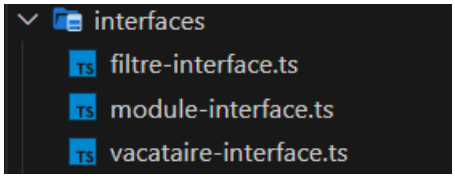
Ma tâche principale terminée, j'ai décidé de prendre du temps pour améliorer certaines parties de l'application afin de faciliter le développement du site de gestion de vacataires pour chacun.

2.2.1 Interfaces TypeScript

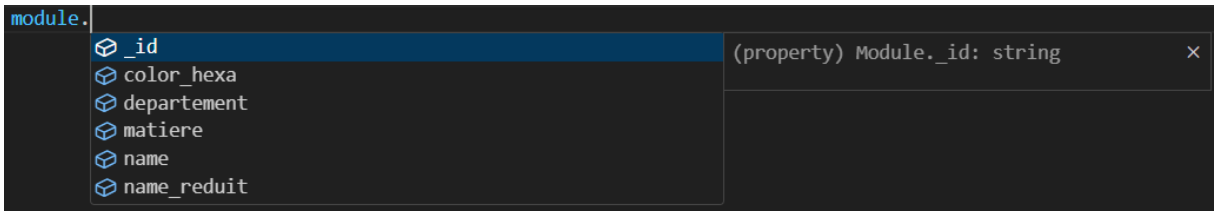
Issue : #52

Le frontend du projet est basé sur le framework Angular ce qui implique donc l'utilisation de TypeScript. J'ai cependant remarqué quelques mauvaises pratiques vis-à-vis de ce dernier dans certains endroits du code ; notamment un problème de typage pour les données renvoyées par l'API. En effet, l'avantage d'utiliser TypeScript plutôt que JavaScript est de pouvoir assurer un typage efficace des variables utilisées et ainsi faciliter le développement, cependant les variables contenant les données principales de chaque page (modules pour la page des cours et vacataires pour la page des vacataires) sont typées en « any » rendant donc TypeScript inutile à leur égard.

J'ai alors créé une interface pour chacune de ces variables et je les ai stockées dans un nouveau dossier à la racine de l'application : « interfaces ». A présent, TypeScript connaitra les différentes propriétés que possède chaque objet (vacataire ou cours) ce qui permettra de programmer plus efficacement.



Interfaces 1 - Nouveau dossier des interfaces

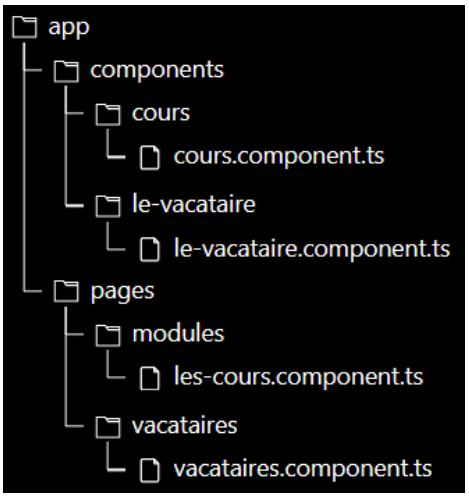


Interfaces 2 - Aide au développement

2.2.2 Arborescence projet

Issue : #53

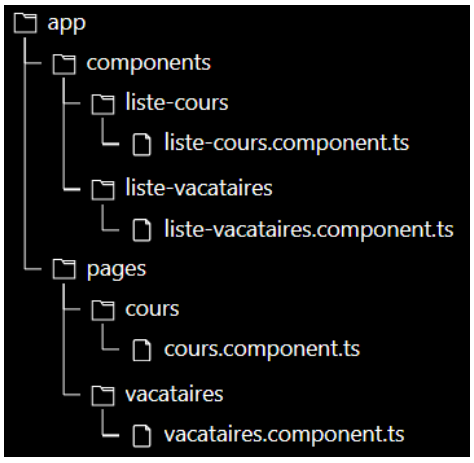
Une autre amélioration nécessaire était le nommage de certains fichiers. En effet, certains composants avaient un nom qui portait vraiment à confusion et il était alors difficile de se repérer efficacement ; en voulant apporter une modification, on se retrouvait facilement dans le mauvais fichier.



Structure projet 1 - Arborescence avant

A présent, la structure est bien plus claire :

- Les composants ont un nom significatif qui représente la partie de page qu'ils traitent (ex : « liste-cours » représentent la liste des cartes affichée sur la page cours)
- Les noms des pages ont été harmonisés (« vacataires » et « cours »)



Structure projet 2 - Arborescence après

3. Bilan

Les filtres (ma tâche assignée) sont complètement fonctionnels, les interfaces TypeScript m'ont aidé sur la fin de mon développement et depuis la réorganisation de la structure du projet je tombe à chaque fois sur le bon fichier à modifier. J'ai également pu apporter mon aide à mes autres

camarades quand ils avaient des problèmes avec leur code ; je suis alors entièrement satisfait de mon travail apporté sur ce projet. Aussi, c'était un plaisir de travailler avec cette équipe qui a produit de très bonnes choses (l'authentification de Bryce et Alex, l'environnement de travail de Marco, la gestion de la page des cours par Enzo…) ; je constate une bonne progression après les deux semaines de renfort des alternants et ai hâte de voir comment sera le produit avancé par les initiaux.