Paul Bernard
François Domecq

Projet de Développement Mobile

Application *Ensciens*

ENSC 2A - Année 2021-2022

Table des matières

Introduction

Énoncé

L'objectif de ce projet est de réaliser une application mobile utilisant l'API que nous avons établi au cours du projet de Génie Logiciel au semestre 7. L'application doit être réalisée en React Native.

Code	Description
ET_01	L'application est développée à l'aide d'Expo et basée sur le langage TypeScript.
ET_02	Tous les fichiers sources sont formatés à l'aide de l'outil Prettier.
ET_03	La structure de l'application distingue : - les composants élémentaires (/components) - les fichiers liés à la navigation (/navigation) - la ou les vues principales (/screens) - les classes métier et techniques pour les accès externes (/services)
ET_04	Les props et l'état de tous les composants qui en font usage sont explicitement définis via des interfaces.
ET_05	La granularité des composants est la plus fine possible (autrement dit : un composant trop "gros" doit être découpé en sous-composants plus élémentaires).
ET_06	Les portions de code délicates et/ou importantes sont commentées.

Thématique de l'application

Pour rappel, l'application que nous avons décidé de développer est un réseau social propre à l'ENSC. Chaque bureau, club et élèves peuvent créer des publications, affichées dans le fil d'actualité dans l'ordre chronologique. Les bureaux et clubs de l'école peuvent créer des évènements (week-end, soirée, ...) et communiquer sur ceux-ci. Un système de vue de profils permet aux étudiants d'avoir des informations sur les autres étudiants et sur les différents clubs et bureaux de l'école.

L'application permet aussi la gestion de la coupe des familles : les responsables du bureau des familles peuvent ajouter des points aux 5 familles de l'école, et les étudiants peuvent suivre le classement de la coupe des familles.

Le projet ASP.NET Core MVC

L'API qui a servi de back-end à notre application est composée de 7 modèles principaux : les familles, les élèves, les publications, les clubs, les bureaux, les commentaires (de publication) et les évènements. L'API permet au client de créer, supprimer, modifier ou de consulter une instance de chacun des modèles (sauf les familles qui ne sont ni créables, ni supprimables). Certaines actions sont restreintes par un système d'authentification : certaines requêtes requièrent d'avoir en header le mot de passe et l'adresse mail d'un utilisateur habilité.

Certains *endpoints* spécifiques ont été ajoutés pour répondre aux besoins de l'application : par exemple, il y a un endpoint spécifique à l'ajout de points de famille.

Cette API a dû être complétée au cours du projet de développement mobile pour répondre à certains besoins spécifiques. L'ensemble de ces ajouts sont détaillés plus loin.

Cette API a été mise en ligne à l'adresse suivante :

https://ensc-ensciens.azurewebsites.net/

Présentation de l'application

Une vidéo de présentation de l'application est disponible à l'adresse suivante :

Démonstration vidéo

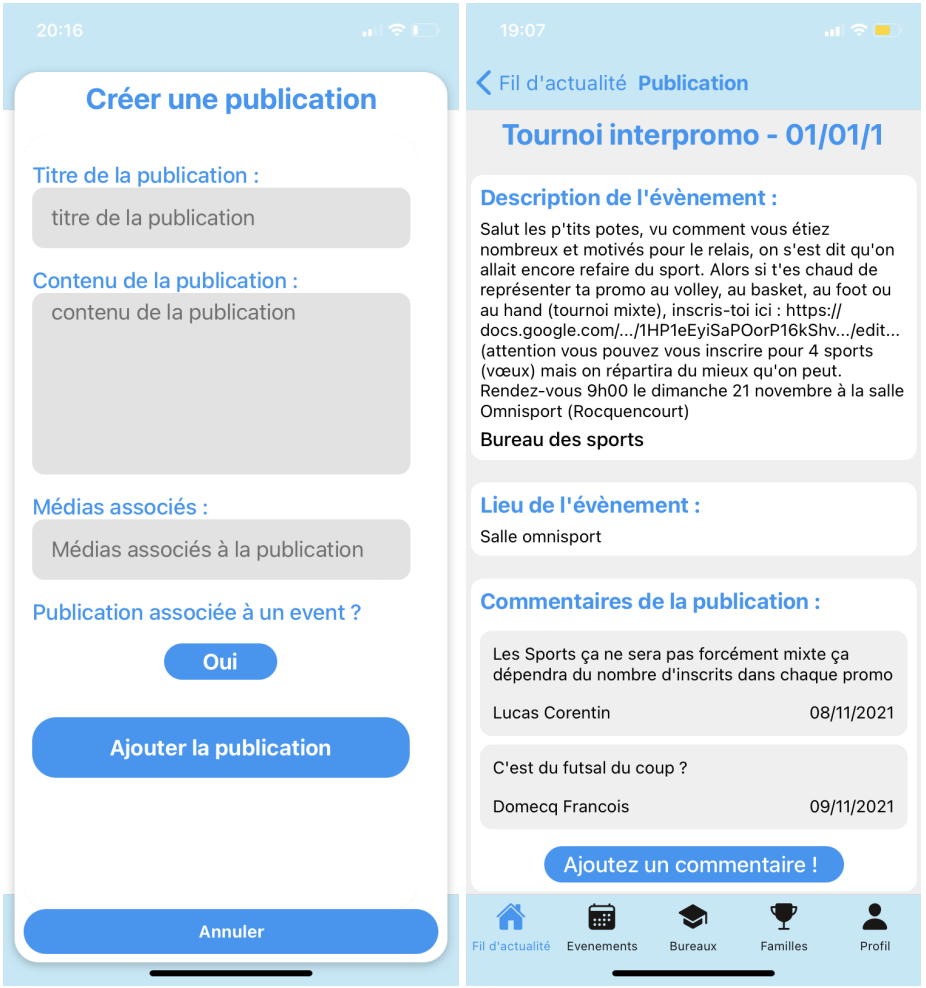
Cette vidéo permet d'avoir une vue d'ensemble de l'application et des fonctionnalités disponibles pour l'utilisateur. L'utilisateur identifié ici est un membre du bureau des familles, il peut en conséquence gérer la coupe des familles, ce que ne peuvent pas faire les autres étudiants.

Les vues principales et les fonctionnalités associées

Le fil d'actualité

figure 1 : Ecran du fil d'actualité

Sur cet écran, l'utilisateur peut visualiser les différentes publications des plus récentes aux plus anciennes. La vue affichée est une ScrollView, l'utilisateur peut donc scroller vers le bas pour afficher toutes les publications. Il peut également effectuer deux autres actions :



figures 2 & 3 : Modal de création de publications & Ecran d'une publication

- La première action que l'utilisateur peut effectuer est d'ajouter une publication. Si il clique sur le bouton '+' sur la page d'accueil, une fenêtre modale s'ouvre et l'utilisateur doit alors remplir un formulaire pour créer une publication. Il peut également décider d'associer un événement à la publication.

- La deuxième action que l'utilisateur peut effectuer est de cliquer sur une publication. Lorsqu'il clique dessus, il est redirigé vers une page où le contenu de la publication est affiché entièrement, ainsi que les commentaires et informations complémentaires. La fonctionnalité pour ajouter un commentaire n'est pas encore fonctionnelle.

L’écran d’authentification

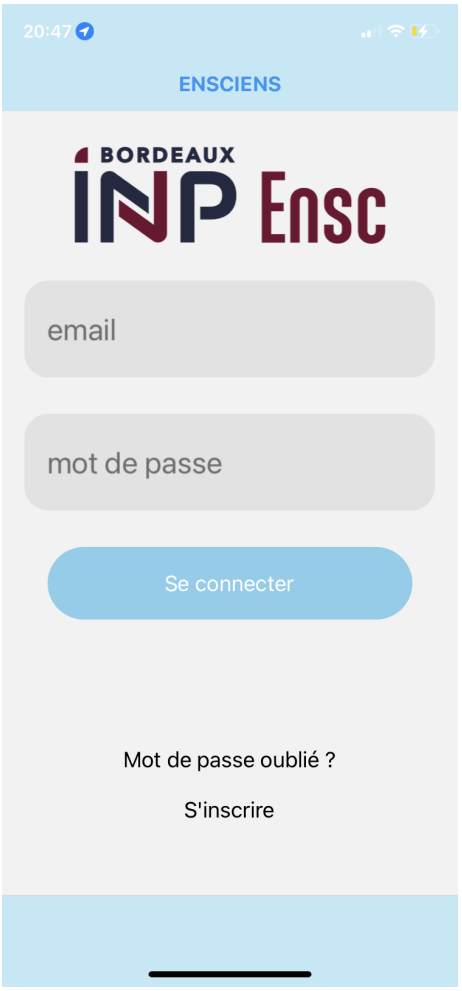


figure 4 : Écran d’authentification

Sur cet écran, l’utilisateur peut se connecter à son compte. L’authentification nous permet d’afficher certaines informations à l’utilisateur en fonction de ses appartenances à certains bureaux ou autres. Cela nous est également très utile pour gérer toute la partie d’autorisation de publication de commentaires, de posts.. que nous avons établi dans l’API.

Les bureaux

figure 5 : Ecran des bureaux

Sur cette page, nous avons affiché tous les bureaux de l'école avec un texte descriptif de ceux-ci. Lorsque l'utilisateur clique sur un bureau, il est redirigé vers la page associée à ce bureau :

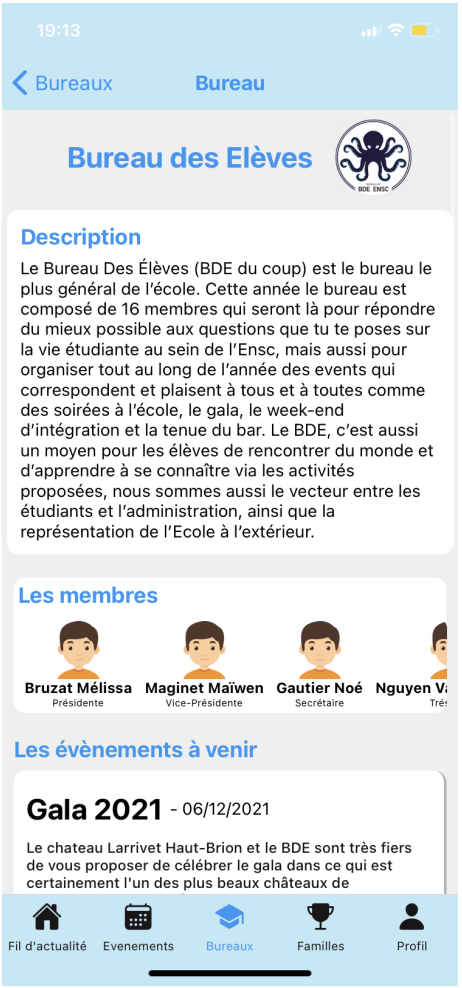


figure 6 : Écran d'un bureau

Sur cet écran, l'utilisateur a accès à la description entière du bureau. Il peut également visualiser les membres qui le composent et leurs rôles dans une ScrollView horizontale. Enfin, il a également accès aux prochains évènements organisés par le bureau ainsi qu'aux clubs qui composent le bureau. Si l'utilisateur clique sur un étudiant dans la ScrollView, il est alors redirigé vers une page 'Eleve' avec des informations sur l'élève sélectionné :

figure 7 : Écran Élève

Les informations affichées sont ici la promotion, la famille d'appartenance ainsi que l'adresse mail. Enfin, il est possible de visualiser si cet élève est membre d'un bureau ou d'un club.

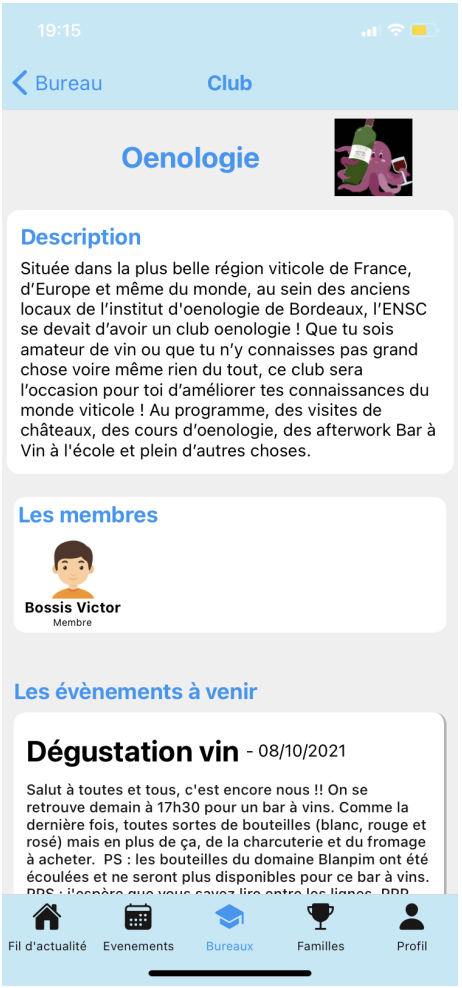
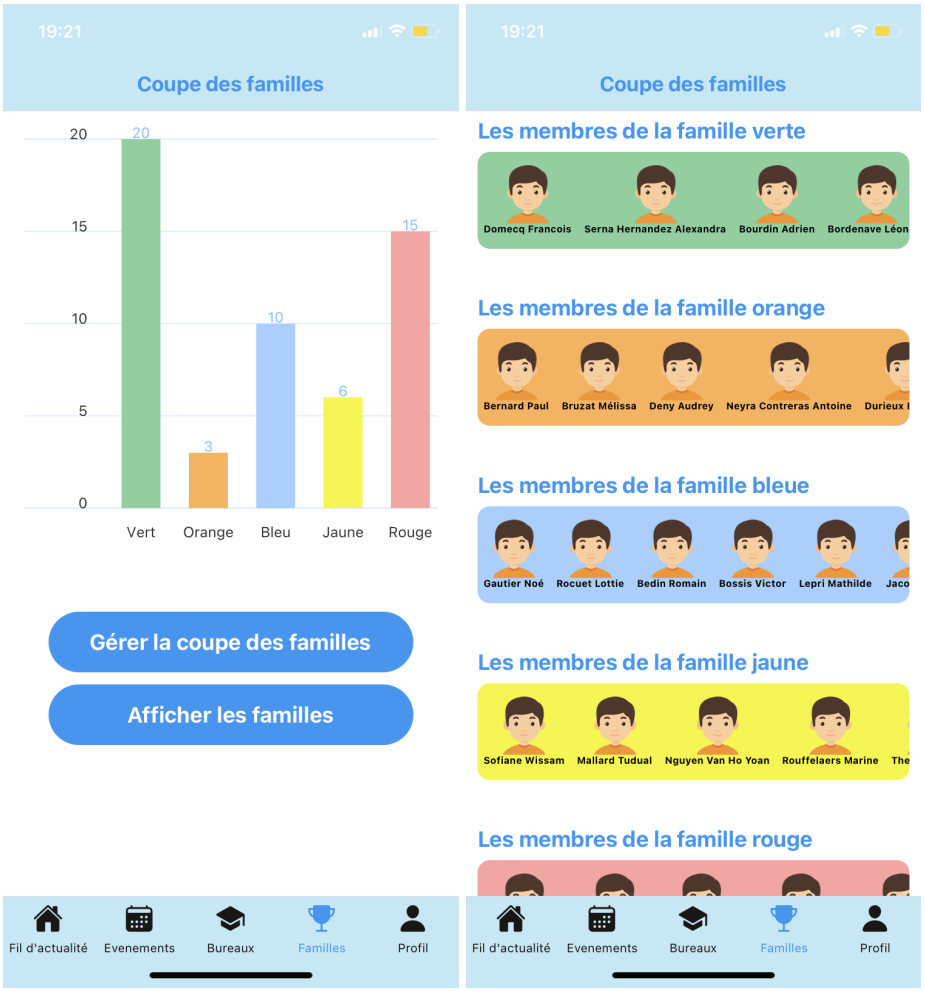


figure 8 : Ecran Club

Si l'utilisateur clique sur un club, il est redirigé vers l'écran 'Club' . Cet écran est très similaire à l'écran 'Bureau' puisqu'une description du bureau est affichée, la liste des membres ainsi que la liste des évènements à venir.

La coupe des familles



figures 9 & 10 : Écran de la coupe des familles

Sur cette page, l'utilisateur a accès au classement de la coupe des familles. Il peut également afficher les familles et les membres qui les composent (cf vidéo démonstration) de la même manière qu'il regarde les membres d'un bureau. Enfin, le bouton 'Gérer la coupe des familles' n'est affiché que pour les membres du bureau des familles. Si un membre du bureau des familles clique sur ce bouton, la modale suivante s'ouvre :

figure 11 : Modal de gestion de la coupe des familles

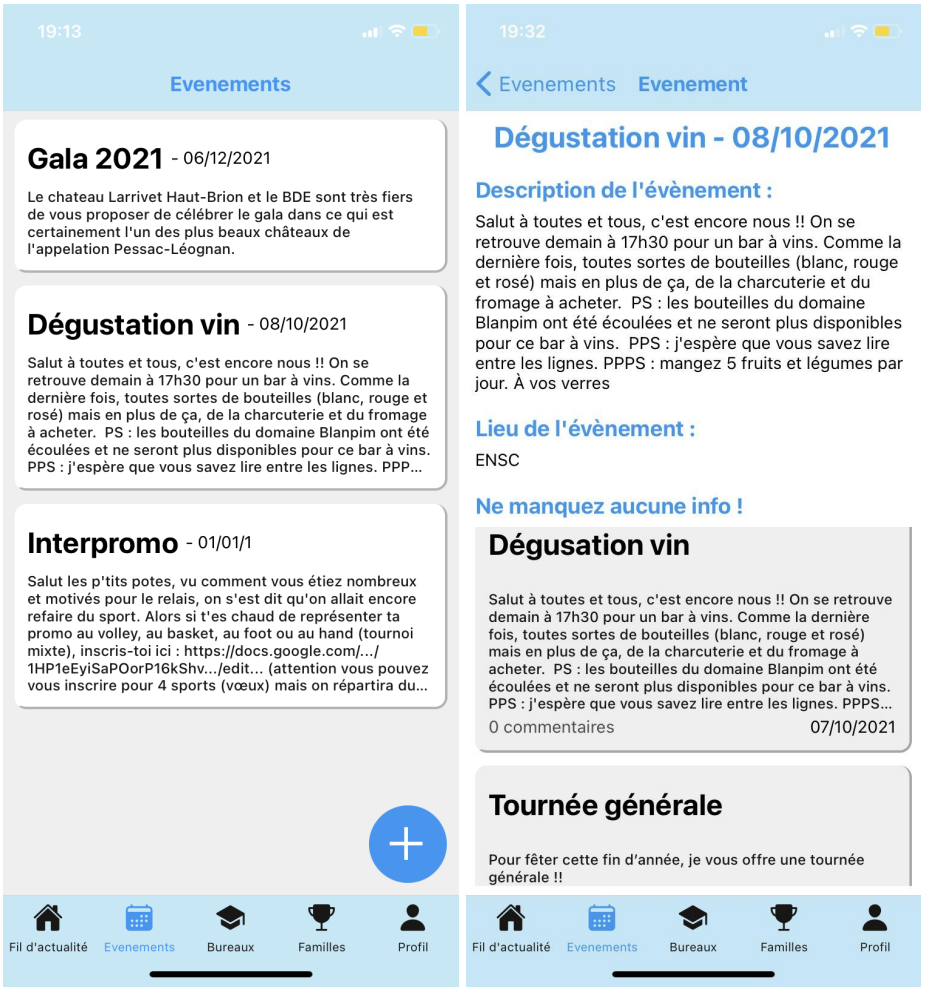
Pour gérer la coupe des familles, l'utilisateur peut sélectionner une famille et sélectionner le nombre de points à ajouter à la famille sélectionnée. Si il valide, les points sont alors ajoutés à la famille et le graphique est mis à jour.

Le profil

figure 12 : Ecran profil

Cet écran est le même que l'écran d'élève à la différence que, depuis cet écran, l'utilisateur peut se déconnecter. Il était également prévu initialement de donner la possibilité à l'utilisateur de modifier certaines informations comme l'appartenance à un bureau ou un club ainsi que son adresse. Par manque de temps, ces fonctionnalités n'ont pas été implémentées.

Les évènements



figures 13 & 14 : Écran des évènements

Sur cet écran, la liste des évènements organisés à venir est affichée. L'utilisateur peut cliquer sur un évènement pour être redirigé vers l'écran de l'évènement. Sur cette page sont affichées les différentes informations ainsi que la liste des publications en lien avec cet évènement.

Les fonctionnalités

Les fonctionnalités, autres que l'affichage des données de l'API, qui ont été implémentées sont :

La connexion

L'utilisateur peut se connecter et se déconnecter de l'application. S'il se connecte et qu'il ferme l'application, il n'aura pas besoin de s'identifier à nouveau sauf si le cache a été réinitialisé ou qu'il s'est déconnecté préalablement. L'authentification était une fonction primordiale de notre application puisque certaines requêtes vers l'API nécessitent d'ajouter des headers qui contiennent le mail et le mot de passe de l'utilisateur connecté pour vérifier que celui-ci dispose bien des droits pour effectuer l'action qu'il souhaite. Nous avons également utilisé cette fonctionnalité pour modifier l'affichage de certaines informations selon l'utilisateur connecté comme avec le bouton de gestion de coupe des familles pour les membres du BDF.

L'ajout d'une publication

Il est possible d'ajouter une publication au fil d'actualité. Pour cela, l'utilisateur doit ouvrir la modal et remplir les champs requis. S'il le souhaite, il peut également associer la publication à un événement qui a déjà été créé.

La gestion de la coupe des familles

Il est possible pour les membres du BDF de gérer la coupe des familles. Pour ce faire, ils doivent ouvrir la modal, sélectionner l'équipe à laquelle ils souhaitent ajouter des points puis saisir le montant de point à ajouter. La requête pour modifier le nombre de points d'une famille vérifie que l'utilisateur est bien membre du bureau des familles dans l'API afin de lui donner ou non l'autorisation d'effectuer l'action.

Suppression publication

Il est possible pour un élève de supprimer une publication qu'il a postée. Pour cela, il lui suffit d'aller sur l'écran de sa publication et de cliquer sur l'icône 'poubelle' dans le header.

Fonctionnalités manquantes

Par manque de temps, nous n'avons pas pu implémenter différentes fonctions qui nous semblent importantes pour un réseau social au sein de l'école. Pour le moment, il n'est pas possible de créer un évènement, ni d'ajouter un commentaire à une publication. De plus, les fonctionnalités liées à l'utilisateur sont très limitées puisqu'il ne peut pas modifier ses informations personnelles ainsi que son mot de passe.

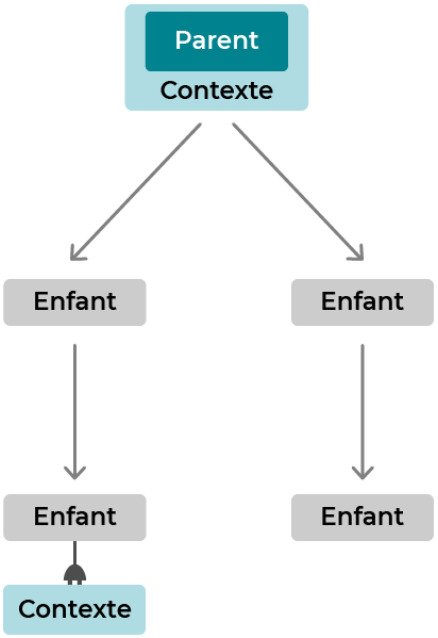
L'application manque aussi de la possibilité d'ajouter des images, que ce soit pour des photos de profil ou pour des images associées à des publications. Cette fonctionnalité a déjà été mis de côté en janvier du côté de l'API, puisqu'elle demandait un effort supplémentaire trop conséquent, notamment en matière de sécurité, de gestion de base de données, et d'appels API.

Implémentation

L'authentification

L'authentification était primordiale pour notre application. En effet, il fallait afficher différentes informations et fonctionnalités en fonction de l'utilisateur connecté. Par exemple, seuls les élèves membres du BDF peuvent gérer la coupe des familles, ou les élèves membres d'un bureau peuvent publier au nom du bureau (pas implémenté). Nous avons alors réfléchi à différentes solutions pour partager les informations de l'utilisateur connecté entre les différentes pages de l'application.

La solution la plus adaptée, aurait été d'utiliser le React Context, un outil de state management. C'est une API de React qui permet de partager et de modifier le state globalement dans l'application sans faire passer des props d'un composant parent vers plusieurs composants enfants.



La figure ci-dessus représente l'utilisation de Contexte. Un composant parent va modifier le state d'une variable contenue dans le Contexte. Alors en se connectant au Contexte et sans passer par des props, un composant enfant est capable de récupérer ces données et de les modifier à son tour. Cependant, cet outil de state management est un 'hook', il n'est donc pas utilisable pour les composants classes. Nous avons donc fait l'impasse sur cette solution pour privilégier la solution suivante :

Les données de l'utilisateur de l'application sont sauvegardées en local en mémoire, à l'aide de la bibliothèque React Native Async Storage. Au chargement de l'application, l'application récupère l'utilisateur stocké en mémoire. Dans ce cas là, l'application s'ouvre sur le fil d'actualité et l'utilisateur est connecté. Sinon, l'application s'ouvre sur le formulaire de connexion.

À la racine de l'application (dans le fichier ./navigation/tab-navigation.tsx), qui permet de choisir la vue Fil d'Actualité ou le formulaire de connexion, il y a un state nommé user qui contient undefined si il n'y a pas d'utilisateur connecté ou la valeur de l'utilisateur si il est connecté. Cette state peut être modifiée par une fonction updateUser. Cette fonction est passé en tant que props à deux vues : la vue Profil et la vue du formulaire de connexion. Ainsi, lorsque l'utilisateur se connecte ou se déconnecte dans la vue Profil, la fonction updateUser est appellée, et ainsi, puisque le state est modifié, l'écran bascule vers le formulaire de connexion (si l'utilisateur se déconnecte) ou vers le fil d'actualité (si l'utilisateur se connecte).

Lorsqu'une vue a besoin d'avoir des informations sur l'utilisateur de l'application, l'espace mémoire associé à la variable user est lu.

Changements dans l'API

Bureau

GET /api/BureauApi/{id}/members

Cette commande renvoie la liste des membres du bureau sous forme d'objet composé :

- d'une chaîne de caractère correspondant au rôle de l'élève dans le bureau
- d'un objet Eleve qui contient les informations de l'élève

Club

GET /api/ClubApi/{id}/members

Cette commande renvoie la liste des membres du club sous forme d'objet composé :

- d'une chaîne de caractère correspondant au rôle de l'élève dans le club
- d'un objet Eleve qui contient les informations de l'élève

Publication

GET /api/PublicationApi/{id}/publicateur

Cette commande renvoie une chaîne de caractères composée du prénom (que pour les élèves) et du nom du publicateur de la publication.

GET /api/PublicationApi/{id}/comments

Cette commande renvoie la liste des commentaires d'une publication.

GET /api/PublicationApi/canPost

Renvoie la liste des identités sous lesquelles l'utilisateur peut publier une publication (soi-même, et les clubs ou les bureaux desquels il ou elle est président.e ou vice-président.e.

Sécurité : La requête à l'API doit avoir en entête les champs Mail et Password contenant respectivement l'adresse mail et le mot de passe d'un élève dans la base de données.

Commentaire

GET /api/CommentaireApi/{id}/Publicateur

Cette commande renvoie une chaîne de caractères composée du prénom (que pour les élèves) et du nom du publicateur du commentaire.

Evenement

GET /api/EvenementApi/canPost

Renvoie la liste des identités sous lesquelles l'utilisateur peut publier un évènement (pas lui même puisque les élèves ne peuvent pas organiser d'évènement, mais les clubs ou les bureaux desquels il ou elle est président.e ou vice-président.e.

Sécurité : La requête à l'API doit avoir en entête les champs Mail et Password contenant respectivement l'adresse mail et le mot de passe d'un élève dans la base de données.

Eleve

GET /api/EleveApi/auth

Permet d'authentifier l'utilisateur. La requête doit contenir une entête "Mail" et une entête "Password" qui doivent correspondre à un élève dans la base de données. Si les informations sont correctes, l'API renvoie un objet Eleve qui contient l'ensemble des données de l'élève visé (avec le mot de passe en clair, contrairement à une simple requête GET /api/EleveApi/{id}. Si les identifiants sont incorrects, l'API renvoie une erreur 401 Unauthorized.

Gestion de projet

Répartition des tâches

Ci-dessous se trouve notre tableau de répartition des tâches. Toutes les tâches propres au lancement du projet ont été effectuées ensemble. Le tableau ne présente que les grandes tendances de la répartition du développement : en pratique, nous avons tous les deux travaillé sur tous les fronts du développement, certaines fois pour corriger ou améliorer ce qui a été fait, ou en travaillant en équipe sur des points difficiles.

Pour commencer le développement, nous avons préféré commencer par créer des pages vides pour mettre en place notre navigation.

	Paul	François
I - Lancement du projet		
Choix de l'identité visuelle		
Maquettage des écrans		
Gestion de l'API sur Azure		
Création du dépôt Github		
II - Développement de l'application		
Développement des vues et des modals		
Écran fil d'actualité		
Écran Bureaux		
Écran Coupe des familles		
Écran Profil		
Écran Publication		
Écran Bureau		
Écran Eleve		
Écran Club		
Modal Gérer la coupe des familles		

Modal Ajouter Publication		
Fonctions d'interface avec l'API web		
Gestion de l'authentification		
Ajout d'endpoints à l'API ASP.NET Core		
III - Finalisation du projet		
Relecture du code et commentaires		
Rédaction du rapport		

Bilan et perspectives sur le projet

Tout d'abord, ce projet nous a permis de nous familiariser avec la technologie React Native mais également le TypeScript. Nous avons également dû au cours du premier semestre développer notre API avec la technologie ASP.NET Core qui était nouvelle pour nous. Créer un réseau social est un projet ambitieux, nous savions que la modélisation de l'API serait la partie la plus importante du projet. Nous y avons donc passé beaucoup de temps, mais malgré cela, nous avons tout de même dû apporter beaucoup de modifications à l'API pour récupérer des informations qui n'étaient pas accessibles avec notre API de base.

Concernant la partie frontend de l'application, nous sommes satisfaits du résultat. En effet, nous avons réussi à implémenter différentes fonctionnalités que nous jugions primordiales comme l'authentification ou l'ajout de publications. Tous les écrans que nous souhaitions développer ont été implémentés, et nous avons réussi à bien récupérer et afficher les informations de la base de données de l'application. Pour le design de l'application, nous nous sommes appuyés sur des maquettes que nous avions réalisées sur Figma au lancement du projet, et nous sommes également satisfaits de l'aspect général de l'application.

Comme dit précédemment, créer un réseau social est un projet ambitieux pour lequel nous aurions eu besoin de plus de temps pour le mener à terme. En effet, notre application n'est pas encore totalement complète puisqu'il manque certaines fonctionnalités importantes pour gérer la vie associative de l'école.