UNIVERSITÉ DE NANTES

MIAGE RÉSEAU DES MIAGE DE FRANCE Nantes

Rapport de projet de TP

Développement d'applications mobiles

Amadou BAH & Abdelhamid NEJI

M2 MIAGE ISI 2020- 2021

responsable du module : Mr Alex Morel LAGARDE

Plan

I. Introduction

Dans le module de Développement d'Applications Mobiles, un projet de TP qui balaie toutes les notions abordées dans le module a été donné aux étudiants (en binôme).

Ce document vient restituer le travail qui a été effectué par notre binôme dans le cadre de ce TP.

Dans un premier temps, nous allons évoquer le sujet initial en précisant la raison de son changement. Ensuite, dans second temps, nous allons présenter le nouveau sujet en expliquant les différents choix d'implémentations.

II. Sujet initial

Le projet KAMEIN

En effet, au tout début nous avions prévu de réaliser un projet réel d'un des membres du binôme. Ce projet dénommé "KAMEIN" consiste à la mise en place d'un site internet pour sa préfecture d'origine. Les fonctionnalités prévues du site sont :

- Un tableau de bord permettant d'avoir des indicateurs sur la population de la localité : taux de natalité (global , par genre), taux de mortalité (global, par genre, par catégorie socioprofessionnelle), nombres de mariages par an, taux de divorce par an, …
- Une page d'actualités concernant la localité
- Une page dédiée aux événements socio-culturels et sportifs de la localité
- Une page pour présenter les sites touristiques de la localité et son histoire
- Une interface d'administration pour les gérants du site.

Le recensement de la population de la localité devra être stockée sur un Google sheet qui servira de base de données à l'application. Cette liste de la population qui sera tenue à jour, par l'ajout des naissances et des décès, permettra d'avoir des indicateurs fiables.

Les premières contraintes

Vu le délai court du TP et de celui des autres projets sans parler de la période d'entreprise, il était prévu de réaliser une version très light du projet KAMEIN.

Cependant, on s'est très rapidement rendu compte de l'existence d'un certain nombre de difficultés notamment la prise en main des API Google (Google Sheet, GDocs) que nous voulions utiliser pour stocker les données. En plus de cela, l'utilisation des API Google obligerait à celui qui va tester le projet de refaire toute la configuration du projet (déclarer un compte Gmail, activer des autorisations, obtenir des API KEYS sur Google cloud pour les mettre dans le code), rebuilder pour obtenir l'APK et créer son répertoire Google Drive afin d'y mettre le Google sheet qui servira de base de données. Nous avons trouvé cela

fastidieux pour celui qui va tester notre application. C'est ainsi que nous avons décidé de réadapter KAMEIN dans le cadre de ce TP.

La réadaptation du projet KAMEIN

Nous avions décidé de faire un projet similaire à KAMEIN, mais cette fois-ci pour la ville de Nantes et avec les données fournies par les API de Nantes Métropole.

Avec cette nouvelle réorientation nous n'avions plus de problème liés à la configuration des API Google et nous avions facilement trouvé les API dont nous avions besoins..

Comme le montrera l'historique des commits, nous étions bien partis dans la réalisation du tableau de bord qui contient les indicateurs sur la natalité et la mortalité au sein de la ville.

Une nouvelle contrainte

Après avoir réalisé les classes de services (dans le sens Vue JS) qui contiennent les appels aux API ainsi que certaines méthodes de traitement des données recueillies, nous étions de nouveau confrontés à un blocage : la librairie de chart "Chartist" que nous avions utilisé pour afficher les graphiques du tableau de bord.

En effet, nous avons constaté que les graphiques ne s'affichent pas lorsque l'on rafraîchit la page de l'application. Après avoir épluché pas mal d'articles et de forum de dev nous ne sommes pas parvenu à résoudre le souci alors que le délai approchait à grand pas. C'est ainsi à l'unanimité que nous avons de nouveau décidé de changer de sujet.

III. Le nouveau sujet

Une application de gestion d'événements

Cette fois-ci nous avons de décidé de réaliser une application de gestion d'événements qui proposera les fonctionnalités suivantes :

- Une liste des événements des la ville de Nantes.
- La possibilité de cliquer sur événement pour avoir les détails sur une modale.
- La possibilité d'ajouter un événement sur sa liste de favoris via le clic sur un bouton de cet événement.
- La possibilité de consulter sa liste d'événements favoris et ainsi que d'en supprimer des événements précis.
- La possibilité d'ajouter ses propres événements via un formulaire.
- La possibilité de consulter ses propres événements et ainsi que d'en supprimer certains.

- L'application propose également l'affichage de la position géographique de l'utilisateur après autorisation. Cette fonctionnalité se trouve en pied de page de l'application.

- Un lien vers le code source du projet est disponible en bas de page. C'est d'ailleurs cette fonctionnalité qui nous permettra de tester plus tard un des plugins Capacitor.

Choix technologiques

En ce qui concerne le design, nous sommes partis sur du **Material Design** pour le design de l'application au de Bulma ou Bootstrap par exemple. Nous avons voulu l'essayer pour avoir de technos dans notre palette vu que nous avions déjà testé les 2 autres citées.

Ainsi, les événements sont affichés via des Cards Material Design qui ressemblent beaucoup d'ailleurs à celles de Bulma.

Axios a été choisi pour effectuer les appels REST.

Les événements favoris de l'utilisateur sont stockés dans le **Local Storage** car nous n'avons pas voulu gérer de base de données ou encore d'utiliser des données statiques.

Structuration du code

Nous avons maintenu la structure initiale de KAMEIN même si là nous n'appelons qu'une seule API. Nous avons un répertoire ***services*** qui contient les codes d'appels des API.

La classe ***CommonService*** contient une méthode générique qui permet d'appeler n'importe quels API de Nantes métropoles via **Axios**. Il suffit juste de lui passer les bons arguments à la construction. La classe **AgendaService** qui permet de récupérer la liste des événements fait appel à ***CommonServcie***.

Un répertoire **pages** contient les pages du menu de l'application, à savoir la liste des événements de la ville de Nantes, les événements favoris, les événements personnels de l'utilisateur ainsi que l'ajout de ces derniers. L'affichage du détails d'un événement est fait via une modale pour plus de confort d'utilisation.

Un répertoire ***components*** regroupe les composants utilisés dans les pages.

Les routes se trouvent dans le répertoire ***routes*** .

La version mobile de l'application avec Capacitor

Comme précisé dans le sujet, il était demandé dans le sujet de proposer une version mobile de l'application via Capacitor et d'utiliser au moins deux plugins de ce dernier. En ce qui nous concerne, nous avons avons choisi utilisé 3 plugins Capacitor :

- **Storage** : nous avions utilisé le plugin storage de Capacitor afin de pouvoir stocker les événements favoris et ceux créés par l'utilisateur dans le local storage.

- **Browser** : nous avons voulu tester ce plugin qui permet de lancer un navigateur à l'intérieur de l'application, ainsi lorsque que l'on clique sur le lien pour voir le lien du code source de l'application qui se trouve en bas de page, on accède au repos **Github** du projet.

- **Géolocalisation** : nous avons utilisé le plugin de géolocalisation. Lorsque l'on clique sur le lien qui permet d'afficher la position géographique de l'utilisateur, qui se trouve en bas de page aussi, les coordonnées s'affichent après demande d'autorisation.

L'utilisation de ces plugins nous a permis d'avantage de comprendre Capacitor.

V. Conclusion

Ce projet nous aura permis de mieux comprendre les notions apprises en cours tout au long du module.

Le fait d'avoir rencontré pas mal de difficultés nous a permis de mieux comprendre certaines notions.
Par exemple, lorsque la gestion de l'asynchronisme des appels REST. Au début, nous pensions que le problème d'affichage des graphiques étaient liés aux appels asynchrones des API, donc nous avons assez creusé ce sujet et on s'est finalement rendu compte que le problème est tout autre.
Le binding d'événements entre un composant et son parent : nous avons quelques soucis lorsque nous étions en train de coder le bouton qui permet de fermer la modale qui affiche les détails d'un événement.

Nous avons découvert aussi de nouvelles choses : en plus des plugins Capacitor, nous avons découvert l'utilisation du **Local Storage** (lecture et écriture) à la fois sur le mobile et sur la version web.

Nous sortons de ce projet en ayant le sentiment d'avoir progressé et appris, durant ce module, des notions d'avenir (Capacitor, PWA) qui nous seront utiles pour la suite de notre carrière.