TP Front-End : React Native

L. Daverio et O. Hermant

6 janvier 2023

1 Environnement de travail

Ce TP utilise la machine virtuelle que nous vous avons fournie ou une installation manuelle (Mac M1). Si tout fonctionne, allez directement à la partie suivante. Sinon, des ordinateurs de prêt disponibles à l'audio-visuel (salle L121), que vous pouvez emprunter jusqu'à la fin du PI MobApp. Voici alors les étapes à suivre :

1. Se logger et installer l'image `.ova` dans son répertoire personnel, en sélectionnant ledit répertoire dans l'option d'import "Machine Base Folder".
2. Après l'import, dans la configuration de la VM (VM éteinte), changer si besoin l'option USB à "contrôleur 3.0" et allouer 4Go de RAM à cette machine virtuelle.
3. Il faut alors refaire une partie du TP de prise en main de la tablette, pour connecter celle-ci et la VM. Voir ce TP pour les détails. **Mettre un mot de passe robuste à la VM.**[1]
4. Surtout, **il faut add/commit/push vos fichiers systématiquement** car la VM est liée à un ordinateur, et rien ne garantit disponibilité du même ordinateur la fois suivante. À chaque changement d'ordinateur, il faut alors refaire cette procédure, en plus de devoir clone à nouveau le dépot git si besoin.

2 VM MobApp

Si vous avez la VM virtualbox MobApp, il est temps de la démarrer. La configuration permanente du clavier ne fonctionnait pas très bien, voici les trois commandes qu'il faut copier dans un terminal pour la faire fonctionner

```
f=$(basename $HOME/scripts/Ke*)
sed -i "s/vm/mobapp/g" $HOME/.config/autostart/$f
sed -i "s/vm/mobapp/g" $HOME/scripts/$f
```

Ensuite, redémarrer la VM et vérifier que votre clavier est correctement configuré.

[1] Cette sécurité reste facilement contournable. Si vos données sont sensibles, lorsque vous rendrez l'ordinateur, il peut être intéressant, soit d'effacer intégralement la VM, soit d'effacer de la VM la copie locale de votre dépôt git, après add/commit et **push**, quitte à devoir re-cloner le dépôt la fois suivante.

3 GitLab : configuration, fork et clonage d'un projet

Si ce n'est pas déjà fait, finalisez la configuration de votre compte gitlab CRI : `https://gitlab.cri.minesparis.psl.eu/` en vous rendant sur ce serveur, puis en récupérant votre mot de passe par la procédure "mot de passe oublié".

Suivez la procédure décrite ici :
`https://gitlab.cri.ensmp.fr/mobapp-2022/eleves/kivappa`
Le projet initial de ce TP reprend les exemples de cours.

4 Projet existant: installation

Dans la VM, une fois le clonage du projet git effectué :

1. **Dans un terminal**, allez dans le répertoire `kivappa`, qui correspond au repository git cloné précédemment. Pour ce faire, utilisez les commandes `cd, ls`.

   Toutes les actions et commandes de ce TP qui utilisent un terminal seront à faire en étant dans ce répertoire `kivappa`. En cas de doute, vérifiez que vous êtes bien dans celui-ci, par exemple avec la commande `pwd`.

2. Installez les modules node requis (les dépendances de votre projet) par:

   ```
   mobapp@vm-lmde:~/.../kivappa/$ npm install
   ```

   Cette étape n'a besoin d'être faite qu'à certains moments précis: juste après le clone/checkout ou lorsqu'un de vos camarades a ajouté (et commit/push) l'utilisation d'une nouvelle librairie[2]

3. Vérifiez que la tablette/un téléphone android est bien connecté, avec `adb devices` (cf. tp sur la tablette).

4. Augmentez les ressources systèmes allouées à watchman (ou bien relancer le point 5 à chaque fois qu'il plante), en exécutant dans un terminal le script

   ```
   mobapp@vm-lmde:~/.../kivappa/$ watchman-fix.sh
   ```

5. **lancer le packager d'application** de react-native (rappel : voir point 1), avec la commande :

   ```
   mobapp@vm-lmde:~/.../kivappa/$ npx react-native start
   ```

6. Enfin, **dans un deuxième terminal séparé**[3], lancer la compilation puis l'installation de KivAppA sur la tablette/le téléphone avec la commande :

[2] NB : lorsque l'on se met à utiliser une nouvelle librairie, mis à part son inclusion dans les fichiers javascript/react-native, il faut aussi faire l'installation avec `npm install ma-super-librairie`. De même, pensez à `npm remove` les libraries dont vous ne vous servez plus.

[3] conseil : mettre ce terminal dans un deuxième onglet avec `Fichier -> Nouvel Onglet` ou le raccourci clavier `Ctrl + Shift + T`

```
mobapp@vm-lmde:~/.../kivappa/$ npx react-native run-android
```

Cela prend un certain temps, surtout la première fois, c'est normal. Ensuite, à chaque nouvelle sauvegarde, l'application mobile devrait se recharger automatiquement. Si ce n'est pas le cas, il faut relancer la commande ci-dessus.

7. être patient et vérifier régulièrement l'état de la compilation du projet *et* de son installation sur la tablette par le serveur d'application dans l'autre terminal.

   Une fois que c'est terminé, il devrait y avoir sur votre tablette/téléphone une app, nommée `kivappa` avec un bouton et un compteur, comme vu en cours.

5 Extensions de l'app

Vérifiez régulièrement avec `git status` que tous les fichiers que vous avez modifiés sont bien inclus dans le commit à venir. Ne pas oublier de faire un **`git add`** de tous vos fichiers importants, puis un **`git commit`** et **`git push`**.

5.1 Bouton de remise à zéro

Dans ce premier exercice, ajoutez à votre application un bouton de remise à zéro du compteur.

Facultatif : Faire en sorte que les boutons soient alignés sur une seule ligne.

5.2 Bouton bonjour

Ajoutez un bouton permettant de saluer l'utilisateur lorsqu'il clique dessus. Faites en sorte que la phrase de bienvenue reste affichée à l'écran dans un composant **`<Text>`** séparé.

5.3 Bouton de changement de style

Ajoutez un autre bouton qui permette de changer la couleur de fond de votre application. Étapes suggérées :

- Apprenez à définir la couleur de fond statiquement dans le code, avec la propriété `style`.
- Utilisez une variable d'état (avec `useState`) qui permette de définir dynamiquement cette propriété.

Facultatif :

- récupérez la couleur de fond depuis un champ d'input donné par l'utilisateur (par exemple, une liste)
- ajoutez une option "aléatoire" à cette liste et tirez une couleur au sort dans tout l'espace de couleurs RGB.
- modifiez d'autres propriétés de style.

5.4 Convertisseur Fahrenheit-Celsius

En dessous de tous ces boutons, développez une fonctionalité de conversion Celsius–Fahrenheit. Quelques étapes possibles, de degré de difficulté croissant:

1. Développez et testez vos fonctions de conversion, en javascript pur. Rappel des formules :
$$\begin{aligned} X\,^\circ\mathrm{C} &= \tfrac{5}{9} * (Y\,^\circ\mathrm{F} - 32) \\ Y\,^\circ\mathrm{F} &= \tfrac{9}{5} * X\,^\circ\mathrm{C} + 32 \end{aligned}$$

2. Ajoutez deux boutons, un pour chaque type de conversion $F \leftrightarrow C$, un champ d'input dans lesquel vous lirez la valeur à convertir et un champ de texte où vous écrirez le résultat de votre conversion.

3. *Facultatif* : faites en sorte que la conversion se fasse *à la volée*, sans bouton. À chaque fois que votre utilisateur modifiera la température dans le champ d'input, la valeur du résultat devra se mettre à jour automatiquement.

4. *Facultatif* : rendez le comportement de la question précédente *symétrique*, comme ici, par exemple:

https://www.google.com/search?q=fahrenheit+celsius

5.5 Style

Ajoutez du style à votre application. Il pourra notamment être utile de vous documenter sur la méthode de disposition des "boîtes flexibles" (flexbox) https://reactnative.dev/docs/flexbox.

Attention : les `<Button>` devront être remplacés par des `<TouchableOpacity>` contenant un `<Text>` pour pouvoir être correctement pris en compte par les flexboxes.

5.6 Fichiers séparés

Séparez vos composantes, ainsi que le style, en différents fichiers. N'oubliez pas, pour chacun des fichiers que vous créez, de les ajouter à votre repository git avec **`git add`**. Et, bien entendu, de commit/push.

6 Requêtes au serveur KiVa

6.1 Bouton de changement d'écran

Faites en sorte de pouvoir basculer à souhait entre votre travail de la section précédente, qui devrait intégralement être contenu dans le composant `<Matin>` et le travail de cette section, qui devra être contenu dans le composant `<ApresMidi>`.

Ce composant est défini dans le fichier `ApresMidi.js`, il reprend ce qui a été vu en cours. Éditer ce fichier avec VSCode (`code ApresMidi.js`), lire et comprendre l'organisation du code.

6.2 Requêtes à un serveur distant

La fonction `sendRequest` envoie une requête `GET` au serveur distant `https://kiva.mobapp.mines-paristech.fr/api`, avec une authentification basique et le login/mot de passe `kiva/bien`. Ses deux paramètres sont :

1. `keyFilter`, qui permet de paramétrer la requête avec le paramètre `filter`. Essayez, par exemple, avec la valeur `%l%`
2. `updateFunction`, la fonction de callback qui sera appelée lorsque la requête aboutira. React native appellera alors `updateFunction` en lui fournissant **un argument**, à savoir le contenu JSON de la réponse du serveur (obtenu à la ligne 18).

6.3 Analyse du code

Des composants font partie de cette app, ainsi que des fonctionalités. Quels sont-ils ? Les documenter **en ajoutant quelques commentaires dans le fichier App.js**.

Ne pas oublier de `git add ApresMidi.js`, puis `git commit` et `git push` après chaque question.

6.4 Requête sans paramètre

Modifier le corps de la fonction `sendRequest` pour enlever le paramètre `filter` de la requête au serveur.

6.5 Argument supplémentaire

Ajouter un troisième argument à la fonction `sendRequest`, afin que l'on puisse lui spécifier la route de l'API sur laquelle faire la requête.

6.6 Mise à jour continue de la liste clefs/valeurs

Faire en sorte que la liste des couples clefs/valeurs se mette à jour *à chaque fois que l'utilisateur tape un caractère* et ne sélectionne que les couples clefs/valeurs correspondant au filtre.

Le bouton `"Envoyer"`, lui, sera modifié pour sélectionner *tous* les enregistrements.

7 Connexion au serveur Flask local

1. Reprendre votre serveur flask `kiva-one`, et le lancer dans un troisième terminal. Pour rappel, dans le répertoire `back-end`, exécuter, successivement, les commandes `make venv ; source venv/bin/activate ; make run`. Vérifier que le serveur local fonctionne bien, par exemple avec la commande `curl -i 'http://0.0.0.0:5000/version'`.

2. il faut **explicitement transférer les requêtes HTTP** faites par l'app sur la tablette au serveur, qui, lui, n'est pas sur la tablette. Pour ceci, il faut demander à `adb` de faire du reverse port forwarding. Dans un terminal, taper :

   ```
   adb reverse tcp:5000 tcp:5000
   ```

   Cela aura pour effet de transférer les requêtes faites sur le port `5000` de la tablette, vers le port `5000` de la VM, justement là où votre serveur KiVa attend ces mêmes requêtes.

   Du point de vue de l'application KiVappA, elle fera donc ses requêtes sur `http://localhost:5000` (*localhost* étant, bien entendu, la tablette/le téléphone).

Si tout fonctionne correctement, alors votre application mobile est maintenant capable de faire les mêmes requêtes que sur le serveur distant, mais sur votre propre serveur local. Testez.

8 Développement full-stack

Il s'agit maintenant de développer conjointement votre API REST et votre application KivAppA. Si vous manquez de routes sur votre API, développez le serveur, et s'il manque des fonctionalités à votre application mobile pour utiliser toutes les routes de votre API, développez votre application.

8.1 Requêtes sur d'autres routes de l'API

Ajouter tous les boutons, les champs et les fonctions nécéssaires pour que l'application mobile implémente :

- l'insertion d'un nouveau couple clef/valeur,
- la suppression du couple associé à une clef de la table.

Voir le TP sur HTML/CSS/JS pour les routes et méthodes offertes par le serveur `kiva`. Vous pourrez vous inspirer du fichier `simplerequest.js` pour la manière de gérer les requêtes de type `POST` (et des autres types).

Encore une fois, ne pas oublier de add/commit/push.

8.2 Unification des fonctions de requêtes

Dans le cas où les questions précédentes ont amené à développer plusieurs fonctions "sendRequest", il est temps de les factoriser et de rendre `sendRequest` configurable.

Faire une unique fonction, qui prendra deux paramètres supplémentaires :

- l'URL sur laquelle faire la requête
- un **paramètre optionnel** fournissant (ou non) un login/mot de passe, selon que la connexion demande une authentification ou pas

8.3 Configuration dynamique de la connexion au serveur

Ajouter des champs à votre app, pour permettre à l'utilisateur de choisir l'adresse du serveur et les login/mot de passe. Remplir ces champs à une valeur par défaut de votre choix. Testez la connexion à votre serveur local ainsi qu'au serveur distant `https://kiva.mobapp.mines-paristech.fr/api`

Faire en sorte que cette configuration soit sur *un écran séparé de votre application*. Il faudra peut-être partager quelques informations entre l'écran de configuration et l'écran principal de l'application à l'aide des `props` de react native.

Puis add/commit/push.

8.4 *(Facultatif)* Nettoyage

Vos camarades et vous-mêmes aurez sûrement alimenté la table KiVa, avec bon goût... ou pas. Développez une fonctionnalité de nettoyage. Exemple : un bouton "supprimer" à côté de chacun des enregistrement sélectionnés par votre requête, un bouton "supprimer tout", ou bien un bouton "supprimer" qui ne nettoie que les enregistrements correspondant au filtre du champ filter.

Séparez vos fonctionalités et composants en différents fichiers.

8.5 Pour la fin

Une fois que vous êtes arrivés jusqu'ici :

- aidez les camarades de votre groupe à arriver au même point que vous ;
- terminez les questions facultatives.