

Installer des émulateurs

Table des matières

I. Contexte	3
II. Le simulateur sur iOS	3
III. Exercice : Appliquez la notion	6
IV. Le simulateur sur Android	7
V. Exercice : Appliquez la notion	15
VI. Essentiel	15
VII. Auto-évaluation	15
A. Exercice final	15
Solutions des exercices	17

I. Contexte

Durée : 1 h

Environnement de travail : Un carnet pour prendre des notes

Pré-requis : Les bases de React Native et de la mise en page

Contexte

Lorsque l'on développe une application mobile, il va de soi que l'on a besoin d'un téléphone mobile pour pouvoir tester le rendu final de l'application. Cependant, on peut posséder un iPhone sans avoir de téléphone Android et, quand bien même ce serait le cas, nous n'aurions pas tous les modèles existants à notre disposition.

Pour palier cela, il est possible d'installer des émulateurs mobiles, qui sont en fait des programmes dont la fonction est de simuler le fonctionnement d'un téléphone. De cette manière, on a accès à plusieurs types de téléphones avec différentes tailles d'écran et différentes versions du système d'exploitation.

II. Le simulateur sur iOS

Objectif

- Installer un simulateur sur iOS

Mise en situation

Sur le système d'exploitation d'Apple, il est possible d'émuler différents types de terminaux, de l'iPhone à l'iPad. Cependant, il n'est pas possible d'utiliser un simulateur iOS, ni même de développer pour iOS, si l'on ne possède pas un ordinateur ayant un système d'exploitation MacOS. Si l'on n'a pas d'ordinateur sous MacOS, on peut directement passer au chapitre 2, ou profiter du chapitre 1 pour sa culture générale.

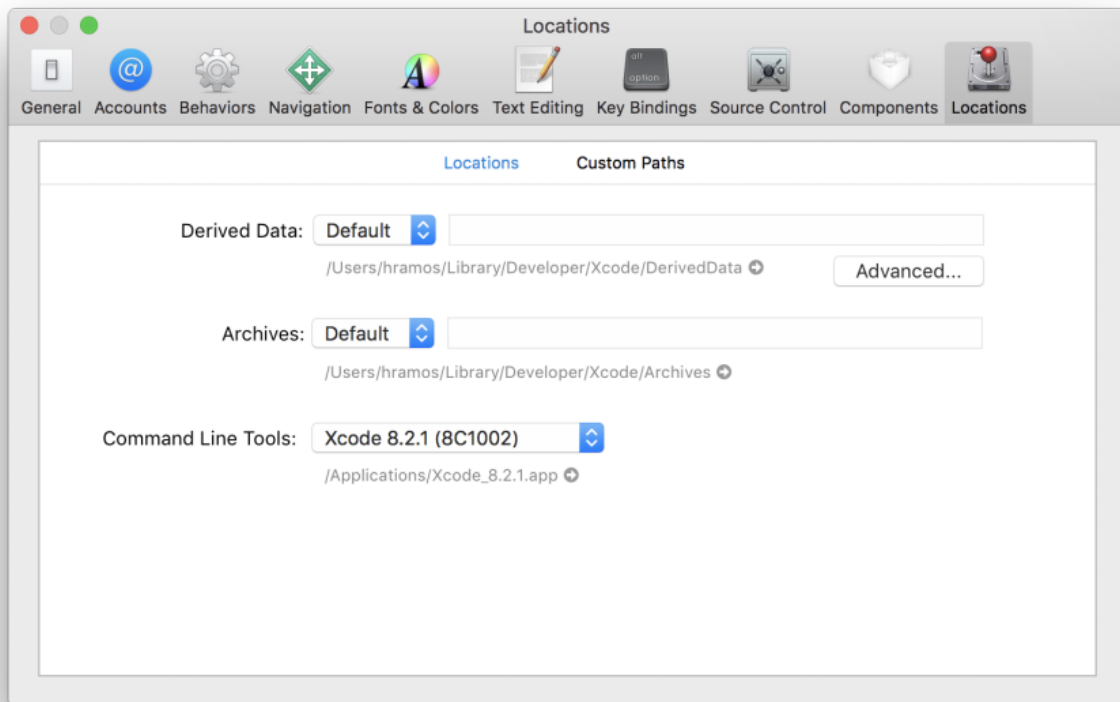
Installer un simulateur sur iOS

Pour installer un simulateur sur iOS, il faut télécharger l'application XCode, qui est un IDE spécialement conçu par Apple pour la création d'applications iOS. Même si nous n'utiliserons pas XCode pour nos futurs développements, il nous servira pour bon nombre de choses, telles que l'accès aux simulateurs ou, nous le verrons plus tard, le processus de build d'une application.

Pour ce faire, il faut télécharger l'application XCode en cliquant sur ce lien.¹ Si on a déjà installé XCode sur notre système, il faut s'assurer qu'il s'agit de la version 9.4 ou d'une version plus récente.

Nous avons également besoin d'installer les outils de ligne de commande d'XCode. Pour cela, il faut ouvrir XCode, puis choisir *Préférences...* dans le menu Xcode. Ensuite il faut aller dans le panneau *Locations* et installer les outils en sélectionnant la version la plus récente dans la liste déroulante *Command Line Tools*.

1 <https://apps.apple.com/us/app/xcode/id497799835?mt=12>



Pour installer un simulateur, il faut faire `Xcode > Préférences...` et sélectionner l'onglet **Composants**. Ensuite, il faut choisir un simulateur avec la version correspondante d'iOS que l'on souhaite utiliser.

Utiliser le simulateur d'iOS

Le plus simple est de lancer son application avec `expo start` puis de presser la touche `i` dans la console, ou cliquer sur `Run on iOS simulator` depuis l'interface web qui a été ouverte automatiquement par la CLI Expo.

De cette manière, Expo se chargera automatiquement d'installer l'application Expo sur l'émulateur et d'ouvrir la bonne URL de projet.

Il n'est pas possible d'avoir accès à l'AppStore sur un émulateur, la seule solution pour installer l'application Expo est donc via la CLI, comme nous venons de le voir.

Il sera également un peu plus difficile d'utiliser la version web d'Expo avec un émulateur. En effet, il n'est pas possible de flasher le QR code depuis l'émulateur : il faudra donc utiliser les autres manières de faire, notamment celle demandant de se connecter sur un compte Expo.

Syntaxe **À retenir**

- Le simulateur de iOS est un outil très pratique, mais disponible uniquement pour la plateforme d'Apple. En effet, si l'on a pas de Mac, on ne peut pas l'utiliser, ni même développer d'applications pour iOS.
- Son installation est relativement simple, car il est embarqué par défaut avec XCode, l'environnement de développement de base pour iOS et Apple.

Complément

- <https://reactnative.dev/docs/environment-setup>
- <https://docs.expo.io/workflow/expo-cli/>
- <https://docs.expo.io/workflow/ios-simulator/>

III. Exercice : Appliquez la notion

Question

[solution n°1 p.19]

Affichez une carte avec une ombre spécifiquement pour iOS et testez le rendu final sur votre nouvel émulateur.

Vous devrez utiliser les propriétés `shadowOpacity`, `shadowRadius`, et `shadowOffset`.

Le rendu doit ressembler à cela :

Carrier 

3:21 PM



Bienvenue

Sur cette magnifique carte de votre émulateur d'iPhone.

IV. Le simulateur sur Android

Objectif

- Installer un simulateur pour Android

Mise en situation

Contrairement à son homologue de chez Apple, le simulateur Android est supporté par tous les systèmes d'exploitation, allant de Windows à Apple et passant par Linux. Son installation est par contre un peu plus complexe, car il ne suffit pas d'installer un simple package applicatif comme sur Mac. Voyons ensemble comment procéder.

Installer un simulateur pour Android

Avant toute chose et selon le système d'exploitation utilisé, il faut un kit de développement Java :

- **Pour Mac**

Ouvrir un terminal puis taper la commande : `brew cask install adoptopenjdk/openjdk/adoptopenjdk8` ou, si l'on a déjà un JDK installé, il doit être en version 8 minimum.

- **Pour Windows**

Le JDK 8 est accessible ici <https://openjdk.java.net/projects/jdk8/> et il faudra Python 2, qui est disponible ici <https://www.python.org/downloads/windows/>.

- **Pour Linux**

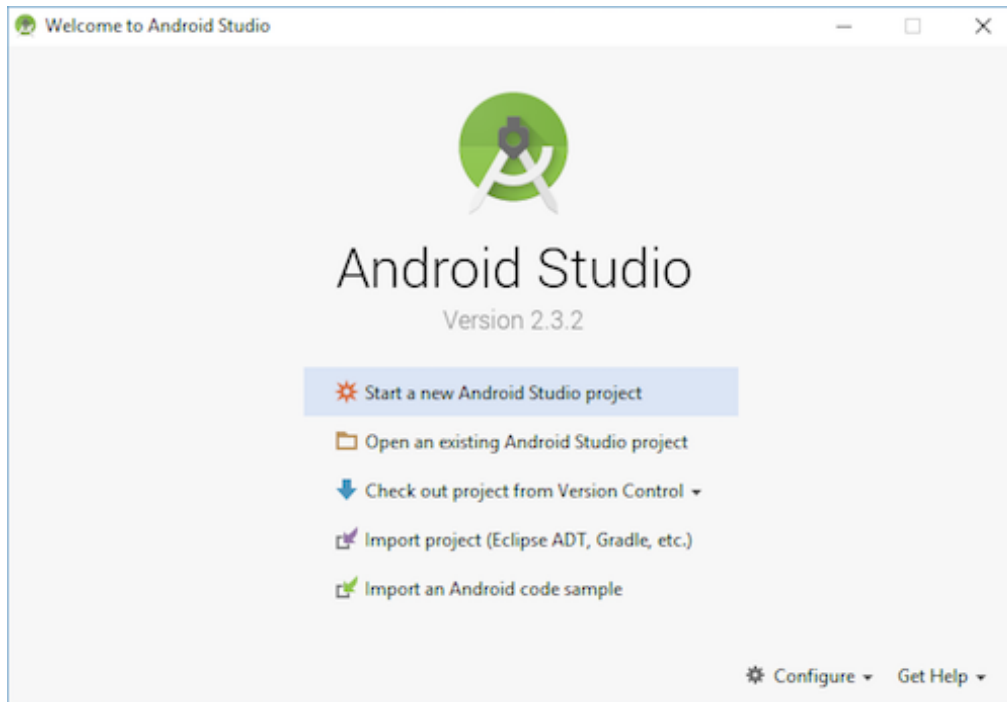
Il faudra idéalement un OpenJDK, soit ici <https://openjdk.java.net/>, soit grâce au gestionnaire de paquets, en fonction de votre distribution Linux.

Une fois le JDK prêt, la deuxième étape consiste à installer Android studio, que l'on peut récupérer à cette adresse : <https://developer.android.com/studio>. Il faut ensuite suivre les instructions de l'assistant d'installation d'Android Studio, et s'assurer que les cases situées à côté de tous les éléments suivants sont cochées :

- Android SDK
- Android SDK Platform
- Android Virtual Device
- Ainsi que Performance (Intel® HAXM) sur Windows

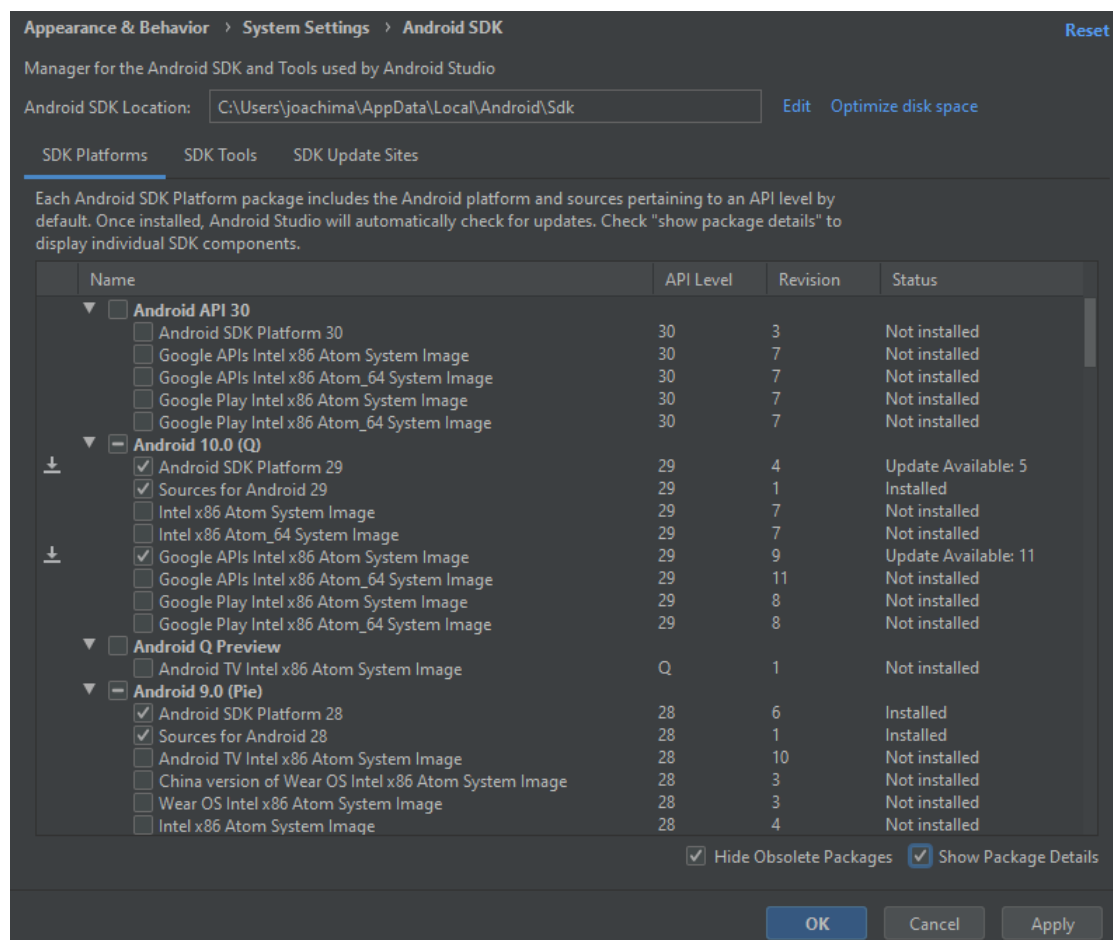
Il faut ensuite cliquer sur **Suivant** et continuer l'installation.

Une fois sur cet écran, il faut s'assurer que l'on possède une version spécifique d'Android SDK, et cliquer sur **Configurer**, puis **SDK Manager**. Sinon, on peut aussi cliquer sur **Préférences → Appearance & Behavior → System Settings → Android SDK**.



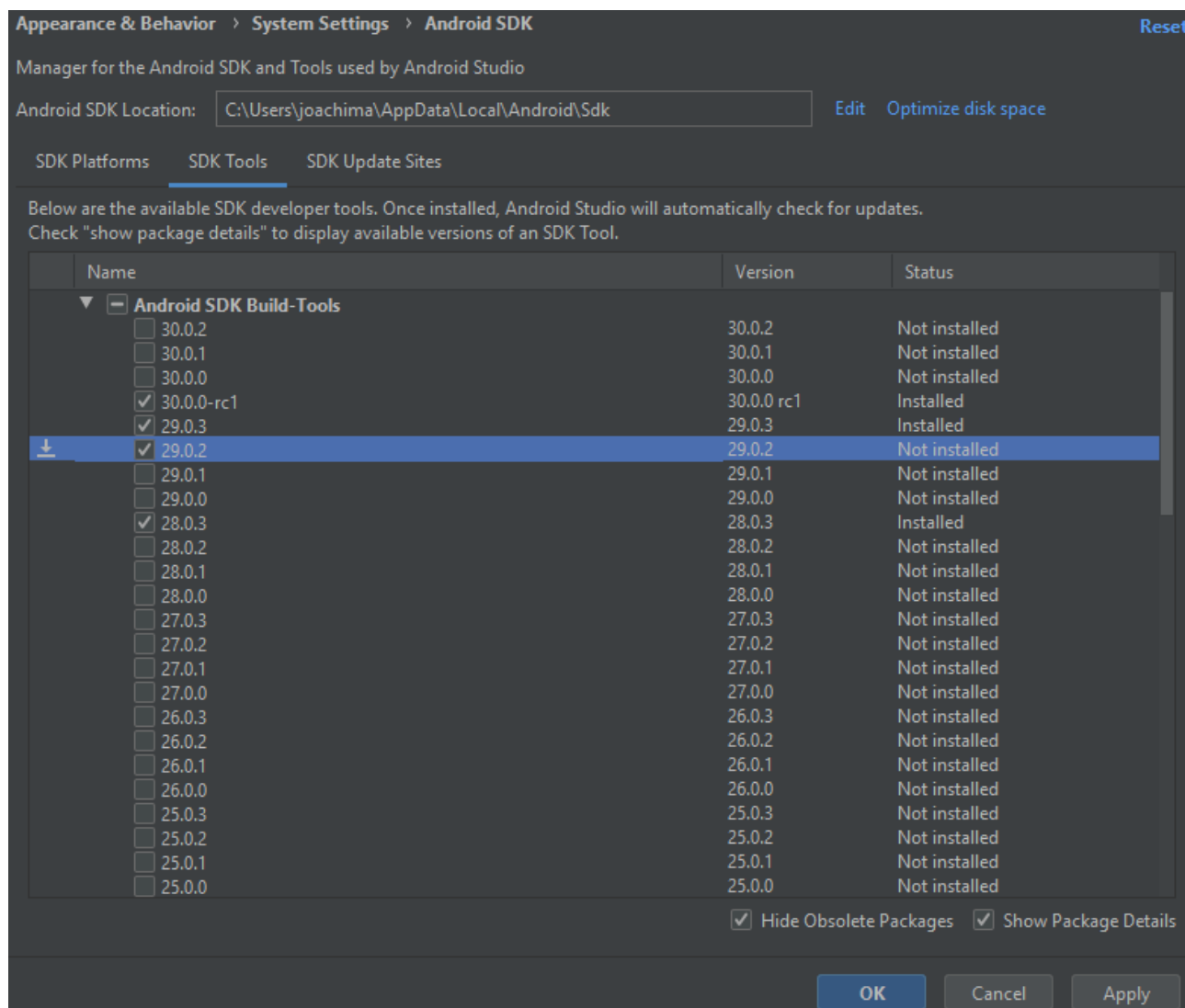
Il faut sélectionner l'onglet **Plateformes SDK** dans le gestionnaire SDK, puis cocher la case située à côté de **Show Package Details** dans le coin inférieur droit. Cherchez ensuite Android 10 (Q) et vérifiez que ces cases sont cochées :

- Android SDK Platform 29
- Intel x86 Atom_64 System Image ou Google APIs Intel x86 Atom System Image



Ensuite, il faut sélectionner l'onglet **SDK Tools** et cocher la case située à côté de **Show Package Details** ici aussi. Sur **Android SDK Build-Tools**, assurez-vous que 29.0.2 est sélectionné.

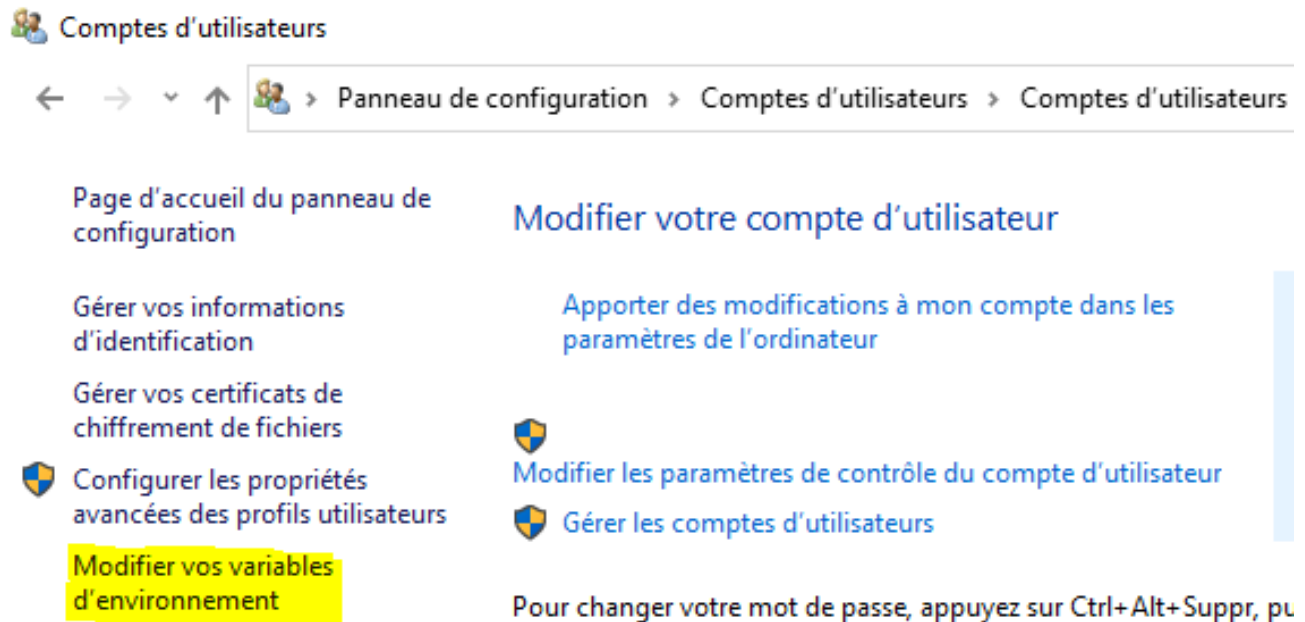
Il faut ensuite cliquer sur **Appliquer** pour télécharger les contenus, ce qui peut prendre un moment : il faudra alors patienter.



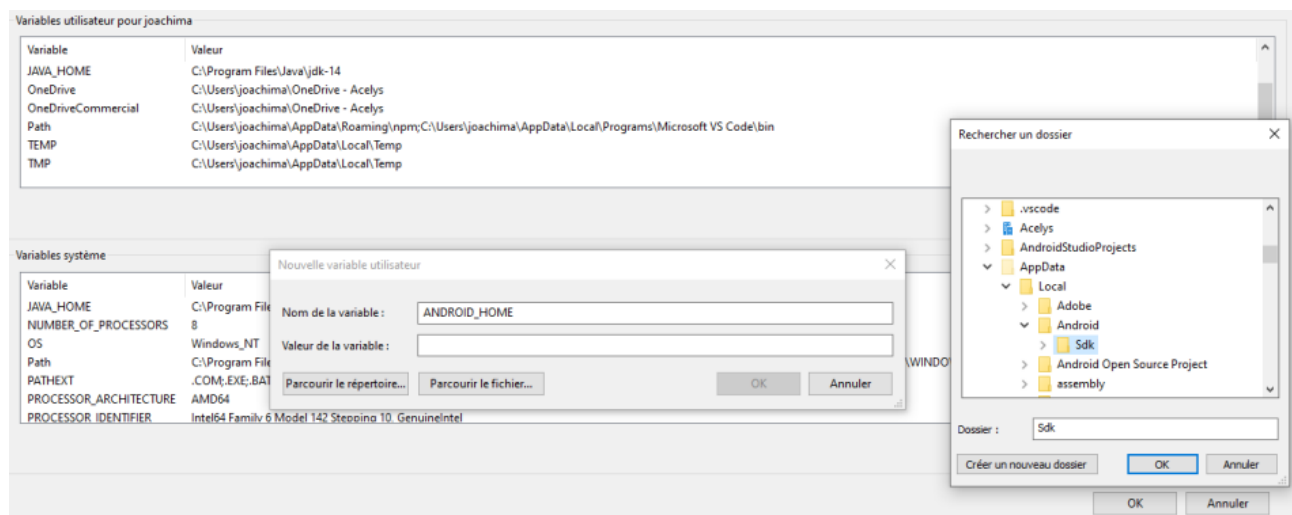
Prochaine étape, définir les variables d'environnement

- Sur Windows :

Il faut ouvrir le panneau de configuration puis **Compte utilisateurs** et de nouveau **Compte utilisateurs**. Enfin, il faut cliquer sur **Modifier vos variables d'environnement** :

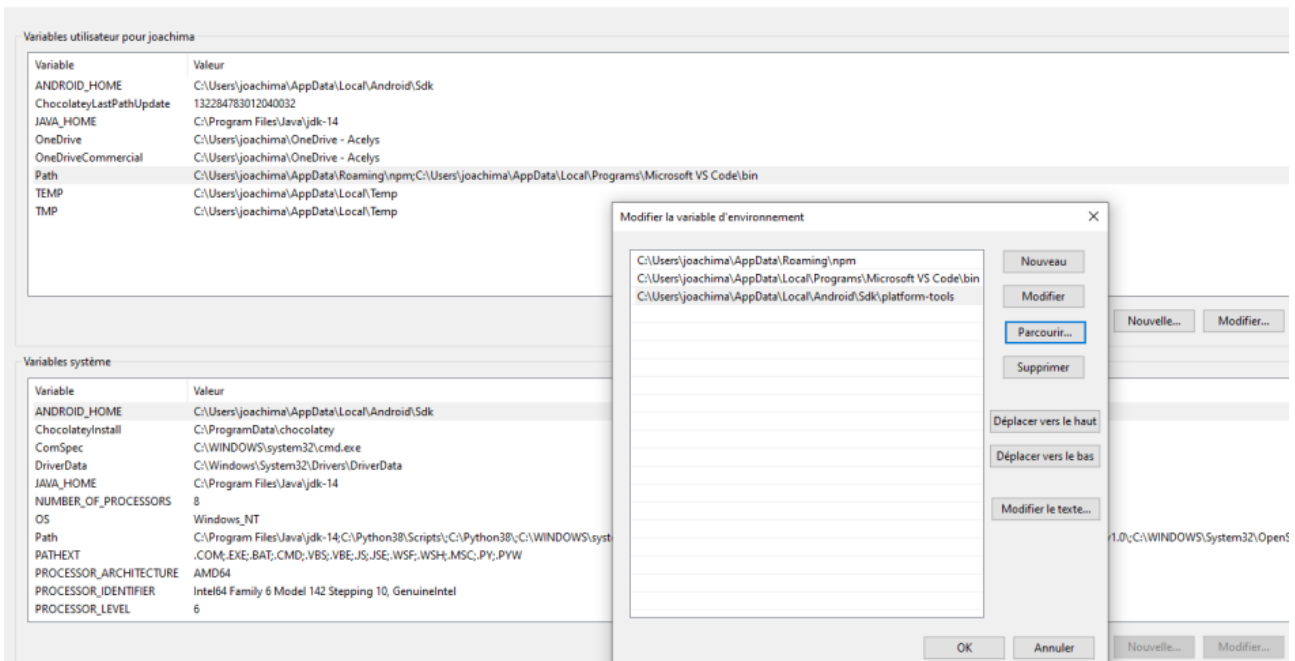


Vous devez ensuite créer une variable `ANDROID_HOME`, en cliquant sur **Nouveau**. Celle-ci doit pointer vers le dossier contenant le SDK Android précédemment téléchargé. Généralement, il se trouve dans `%LOCALAPPDATA%\Android\Sdk`.



Ensuite, vous devez éditer la variable `PATH`, en cliquant sur **Modifier**, et lui ajouter une entrée, avec le bouton nouveau, pointant vers les outils du SDK. Généralement, il se situe dans `%LOCALAPPDATA%\Android\Sdk\platform-tools`.

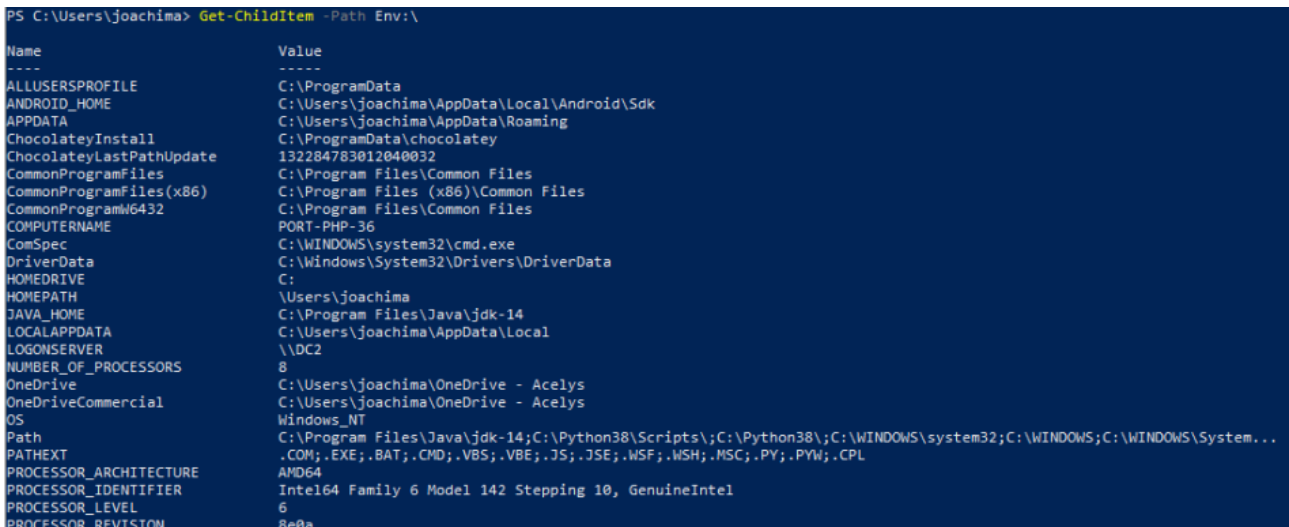
Variables d'environnement



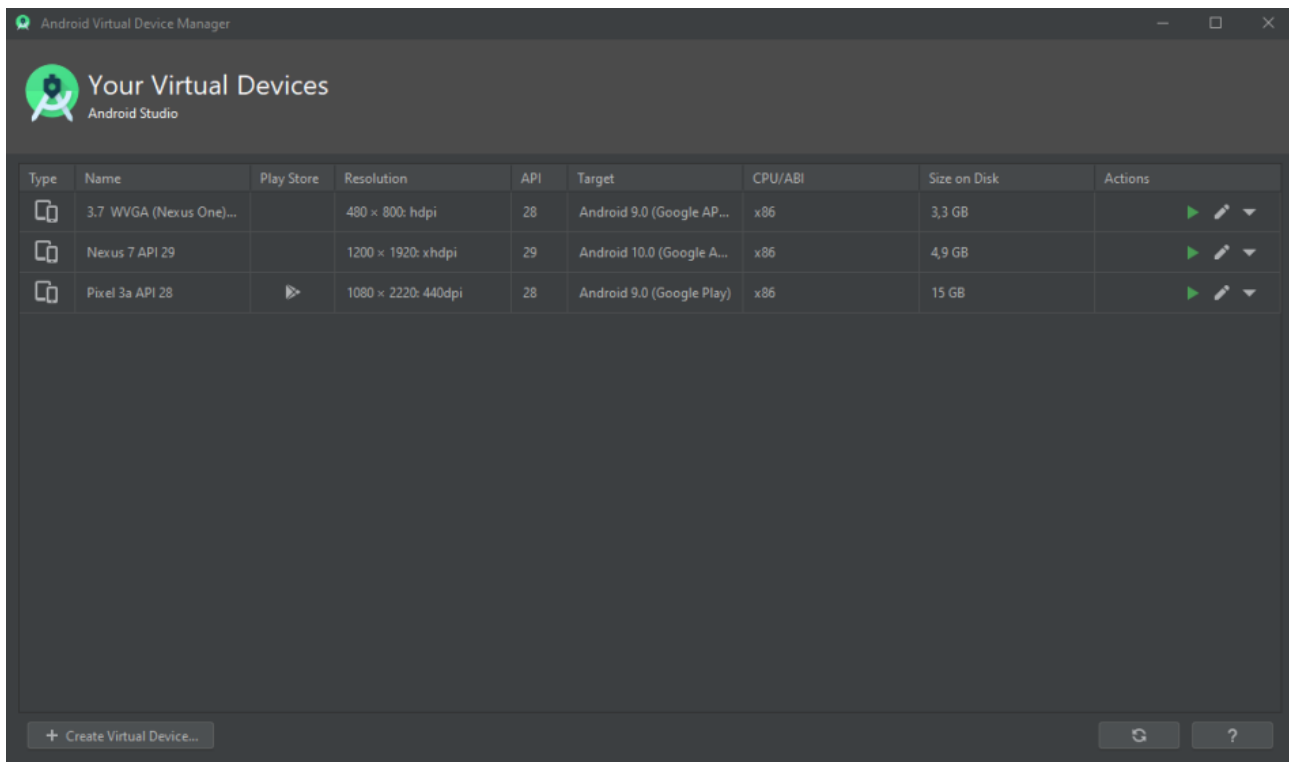
- Sur Mac et linux, il faudra modifier le `~/ .bashrc` ou le `~/ .bash_profile` ou son équivalent sur un autre terminal, comme ZSH, et ajouter les valeurs suivantes :

```
1 export ANDROID_HOME=$HOME/Library/Android/sdk
2 export PATH=$PATH:$ANDROID_HOME/emulator
3 export PATH=$PATH:$ANDROID_HOME/tools
4 export PATH=$PATH:$ANDROID_HOME/tools/bin
5 export PATH=$PATH:$ANDROID_HOME/platform-tools
```

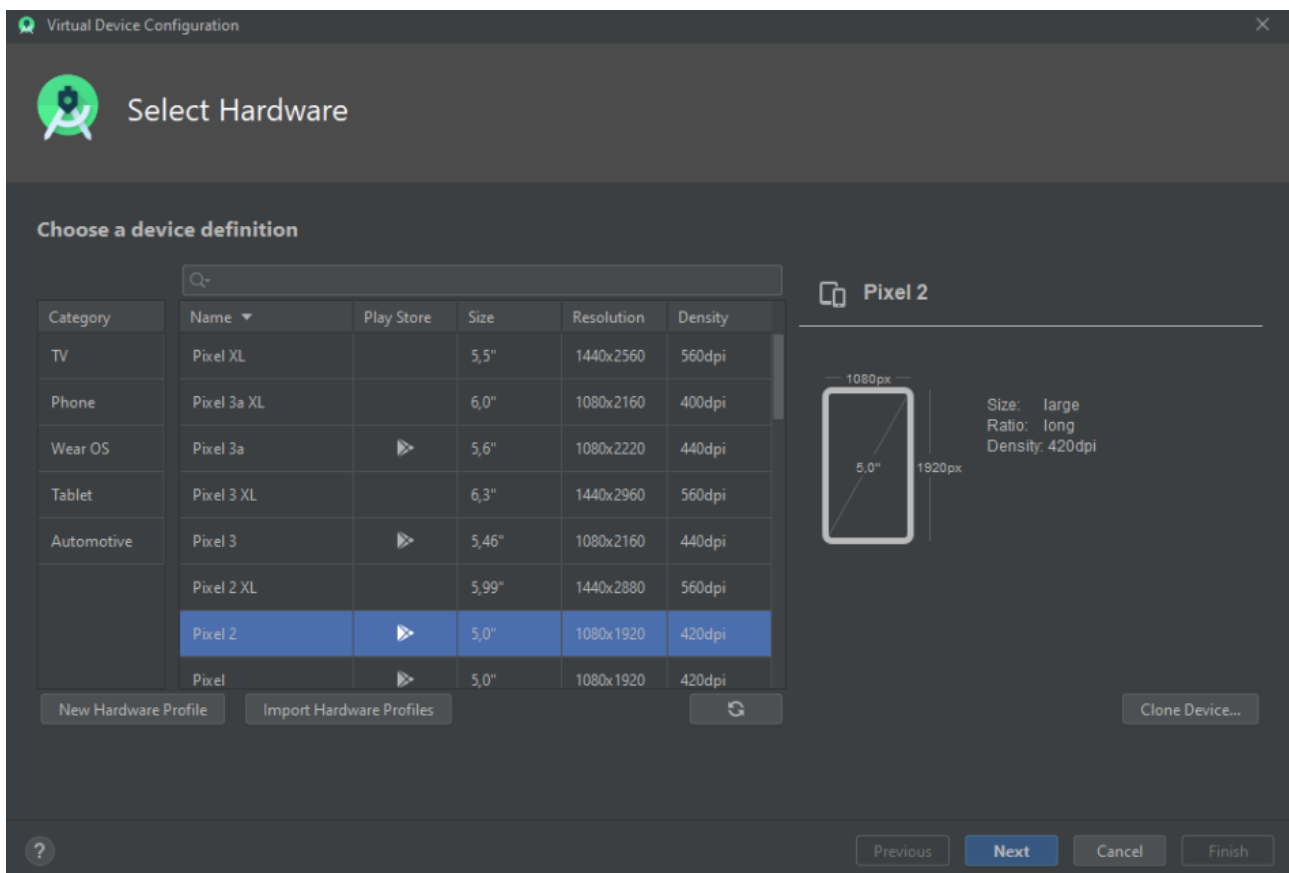
- Sur Windows, il faut lancer Power Shell pour vérifier que la variable d'environnement a bien été ajoutée. Cela se fait avec la commande : `Get-Childitem -Path Env:\`.



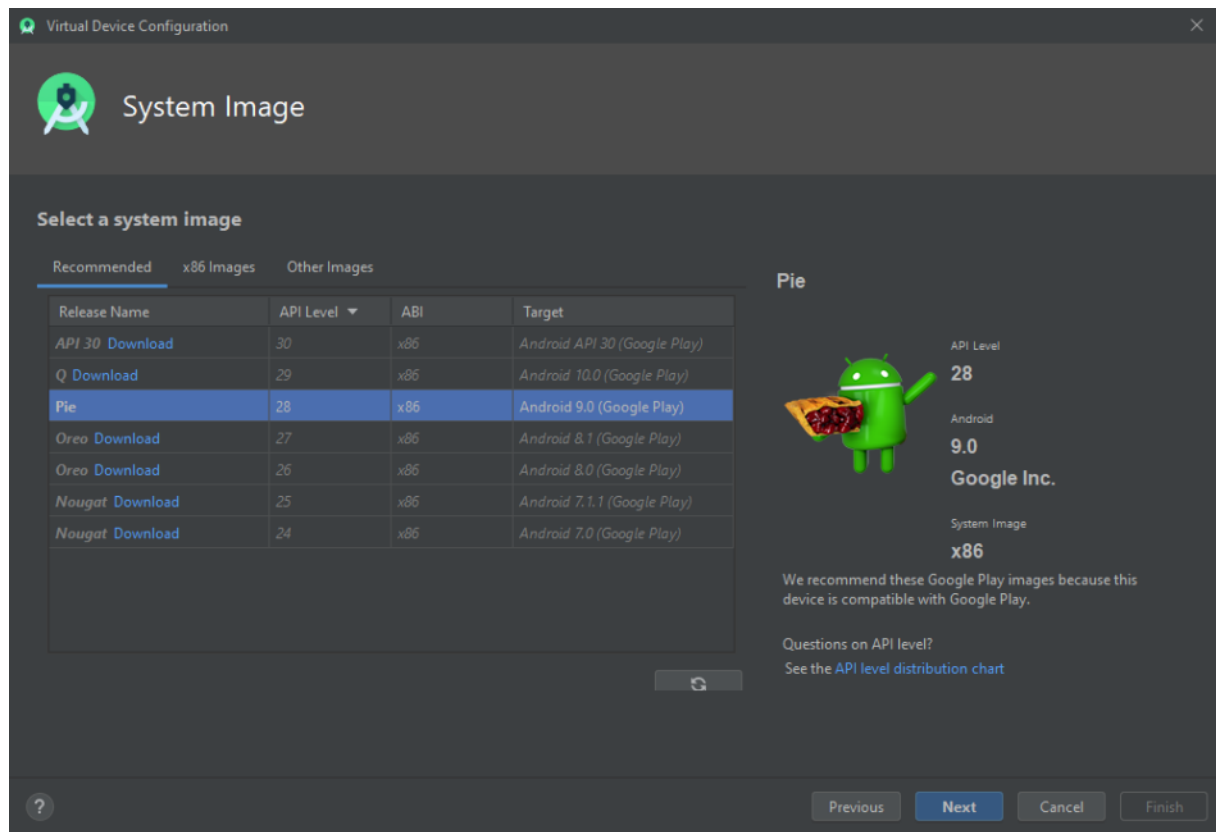
Une fois tout cela fait, à partir de l'écran principal d'Android Studio, allez dans **Outils -> Gestionnaire AVD**, puis appuyez sur le bouton **+ Créer un dispositif virtuel** (ou *Create virtual device* en anglais).



La première étape consiste à choisir le modèle d'appareil que l'on souhaite émuler, puis à cliquer sur **Next**.

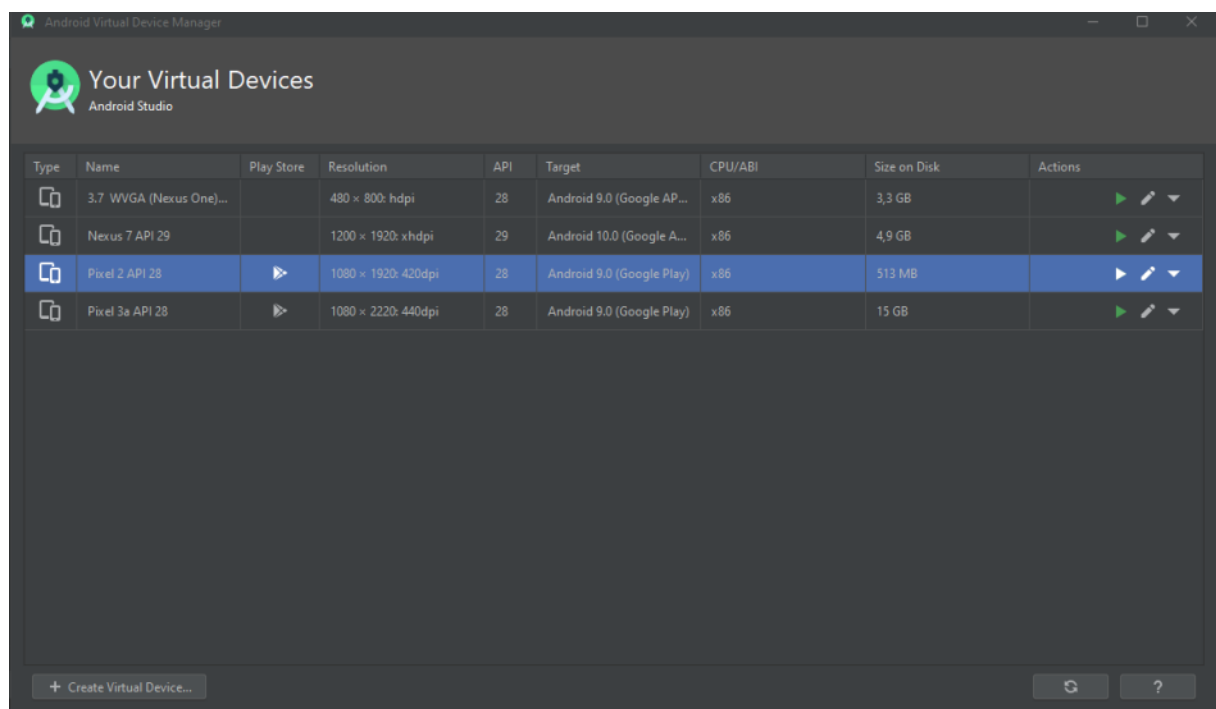


Il faut ensuite choisir la version d'Android que l'on veut utiliser et cliquer sur **Next**.

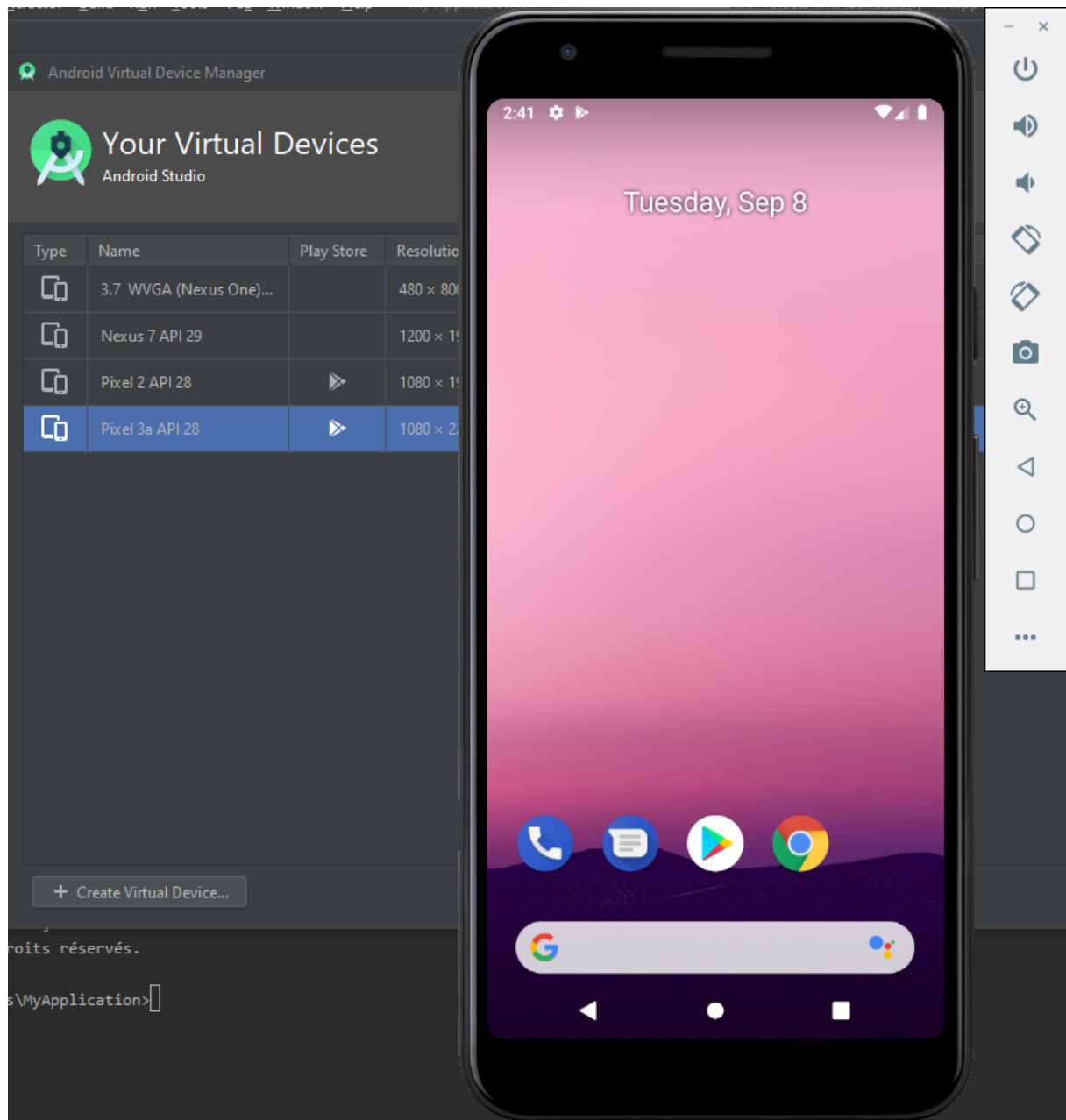


Un dernier écran résumant la configuration demandée permet de paramétrer quelques éléments supplémentaires si on le souhaite, mais généralement, la configuration par défaut est suffisante.

En cliquant sur **Finish**, on revient sur la liste des simulateurs disponibles.



Pour lancer le simulateur, il faut utiliser la flèche verte dans la colonne **Actions** du tableau.



Utiliser le simulateur d'Android

Le plus simple est de lancer son application avec `expo start` puis de presser la touche `a` dans la console, ou de cliquer sur `Run on Android device/emulator` depuis l'interface web qui a été ouverte automatiquement par la CLI Expo.

De cette manière, Expo se chargera automatiquement d'installer l'application Expo sur l'émulateur et d'ouvrir la bonne URL de projet.

Il n'est pas possible d'avoir accès au PlayStore (le magasin d'applications de Google) sur un émulateur : la seule solution pour installer l'appli Expo est donc via la CLI, comme nous venons de le voir.

Il sera également un peu plus difficile d'utiliser la version web d'Expo avec un émulateur. En effet, il n'est pas possible de flasher le QR code depuis l'émulateur : il faudra donc utiliser les autres manières de faire, notamment celle demandant de se connecter sur un compte Expo.

Syntaxe **À retenir**

- L'installation d'un émulateur Android est un peu plus périlleuse que sur iOS : n'étant pas propre à un système d'exploitation, les procédures sont un peu plus compliquées. Il faut notamment avoir Java et Python 2 installés sur son ordinateur et définir des variables d'environnement, telles que `ANDROID_HOME`.
- Une fois Android Studio installé, on peut paramétrer plusieurs types d'émulateurs pour répondre à nos besoins spécifiques.
- Si l'on rencontre le moindre soucis durant l'installation, c'est parce qu'il est possible que les procédures évoluent, ainsi que les versions. Il faudra alors se référer aux liens complémentaires ci-dessous, le temps que ce cours soit mis à jour. Il ne faut pas hésiter à signaler tout problème à ce sujet.

Complément

- <https://docs.expo.io/workflow/android-studio-emulator/>
- <https://reactnative.dev/docs/environment-setup>

V. Exercice : Appliquez la notion

Question

[solution n°2 p.19]

Lancez votre émulateur Android et, cette fois, expérimentez le composant `TouchableNativeFeedback` de Android, ainsi qu'une carte à la propriété `elevation`. Pour la carte, vous pouvez vous baser sur ce qui a été fait dans les modules sur les animations.

Le rendu doit ressembler à la vidéo ci-dessous :

[cf. emulator-4.mp4]

VI. Essentiel

Il n'est possible d'installer un émulateur iOS seulement sur un ordinateur tournant sous le système d'exploitation MacOS. Par contre, pour Android, même si c'est plus compliqué, on peut installer un émulateur sur toutes les plateformes.

Utiliser un émulateur permet de profiter de la puissance de calcul d'un ordinateur, qui dispose généralement d'un meilleur processeur et de plus de mémoire RAM. On peut également développer complètement hors-ligne, étant donné que tous les logiciels tournent sur l'ordinateur.

VII. Auto-évaluation

A. Exercice final**Exercice 1**

[solution n°3 p.21]

Exercice

Quelle variable d'environnement faut-il définir pour pouvoir installer l'émulateur d'Android ?

- ☐ EMULATOR_ANDROID
- ☐ ANDROID_HOME
- ☐ STUDIO_HOME

Exercice

XCode est...

- ☐ Un simulateur de terminaux Apple
- ☐ Un IDE, développé par Apple, pour réaliser des applications iOS
- ☐ Une librairie pour React Native

Exercice

Il est possible d'installer un émulateur iOS sur Windows.

- ☐ Vrai
- ☐ Faux

Exercice

On peut installer un émulateur Android sur Mac.

- ☐ Vrai
- ☐ Faux

Exercice

Quel logiciel est requis pour installer un simulateur d'iPhone sur MacOS ?

- ☐ Visual Studio
- ☐ Eclipse
- ☐ XCode
- ☐ iOS Emulator Toolkit

Exercice

On peut avoir plusieurs émulateurs avec différents iPhones allumés en même temps sur un ordinateur.

- ☐ Vrai
- ☐ Faux

Exercice

Il est possible d'avoir plusieurs émulateurs avec différents Androids allumés en même temps sur un ordinateur.

- ☐ Vrai
- ☐ Faux

Exercice

Pour faire tourner une application React Native sur un simulateur Android, j'ai minimum besoin de...

- ☐ JDK7
- ☐ JDK8
- ☐ JDK9

Exercice

Android Studio, c'est...

- ☐ Un IDE pour développer des applications natives pour Android
- ☐ Un outil pour designer les maquettes de notre application
- ☐ La dernière version d'Android en 2020

Exercice

On a obligatoirement besoin d'un émulateur iOS ou Android et de builder une application pour développer avec React Native.

- ☐ Vrai
- ☐ Faux

Solutions des exercices

p. 6 Solution n°1

```
1 import React from "react";
2 import { View, Text, StyleSheet } from "react-native";
3
4 const App = () => {
5   return (
6     <View style={styles.container}>
7       <View style={styles.card}>
8         <Text style={styles.title}>Bienvenue</Text>
9         <Text style={styles.text}>
10           Sur cette magnifique carte de votre émulateur d'iPhone.
11         </Text>
12       </View>
13     </View>
14   );
15 };
16
17 const styles = StyleSheet.create({
18   container: {
19     flex: 1,
20     justifyContent: "center",
21     alignItems: "center",
22   },
23   card: {
24     backgroundColor: "white",
25     borderWidth: 1,
26     borderColor: "rgba(0, 0, 0, 0.1)",
27     padding: 20,
28     shadowOpacity: 0.5,
29     shadowRadius: 1,
30     shadowOffset: {
31       height: 2,
32     },
33   },
34   title: {
35     fontSize: 24,
36   },
37   text: {
38     marginTop: 10,
39   },
40 });
41
42 export default App;
```

p. 15 Solution n°2

```
1 import React from "react";
2 import {
3   View,
4   Text,
5   StyleSheet,
6   TouchableNativeFeedback,
7   Platform,
8 } from "react-native";
```

```

9
10 const App = () => {
11   return (
12     <View style={styles.container}>
13       // L'opérateur logique (&&) ici ne correspond pas tout à fait à la
14       même façon d'utilisation qu'en JavaScript classique. Ici, l'opérateur a
15       pour but de "raccourcir" le code. Voyons-le comme une condition
16       "if(Platform.OS === "android") {
17         <TouchableNativeFeedback>[...]</TouchableNativeFeedback>
18   } Si la condition est true, alors on affichera le code présent juste après
19   l'opérateur &&, sinon, le code ne sera pas interprété.
20
21     {Platform.OS === "android" && (
22       <TouchableNativeFeedback
23         onPress={() => {}}
24         background={TouchableNativeFeedback.Ripple("rgba(0, 0, 0, 0.2)")}}
25       >
26         <View style={styles.button}>
27           <Text style={styles.buttonText}>Bouton à retour natif</Text>
28         </View>
29       </TouchableNativeFeedback>
30     )}
31     <View style={styles.card}>
32       <Text style={styles.title}>Bienvenue</Text>
33       <Text style={styles.text}>
34         Sur cette magnifique carte de votre émulateur Android.
35       </Text>
36     </View>
37   </View>
38 );
39 };
40
41 const styles = StyleSheet.create({
42   container: {
43     flex: 1,
44     justifyContent: "center",
45     alignItems: "center",
46   },
47   button: {
48     padding: 20,
49     borderRadius: 5,
50     backgroundColor: "green",
51   },
52   buttonText: {
53     color: "white",
54   },
55   card: {
56     marginTop: 10,
57     backgroundColor: "white",
58     borderWidth: 1,
59     borderColor: "rgba(0, 0, 0, 0.1)",
60     padding: 20,
61     elevation: 3,
62   },
63   title: {
64     fontSize: 24,
65   },
66   text: {

```

```
67     marginTop: 10,  
68   },  
69 });  
70  
71 export default App;
```

Exercice p. 15 Solution n°3

Exercice

Quelle variable d'environnement faut-il définir pour pouvoir installer l'émulateur d'Android ?

- ☐ EMULATOR_ANDROID
- ☒ ANDROID_HOME
- ☐ STUDIO_HOME

 La bonne variable est `ANDROID_HOME`, elle indique le chemin du SDK Android.

Exercice

XCode est...


- ☐ Un simulateur de terminaux Apple
- ☒ Un IDE, développé par Apple, pour réaliser des applications iOS
- ☐ Une librairie pour React Native

 XCode est l'IDE pour réaliser des applications natives iOS en objective-C ou Swift.

Exercice

Il est possible d'installer un émulateur iOS sur Windows.


- ☐ Vrai
- ☒ Faux

 Malheureusement, les émulateurs iOS ne sont disponibles que sur Mac.

Exercice

On peut installer un émulateur Android sur Mac.

- ☒ Vrai
- ☐ Faux

 Les émulateurs Android tournent sur Java et sont donc disponibles sur toutes les plateformes.

Exercice


Quel logiciel est requis pour installer un simulateur d'iPhone sur MacOS ?

☐ Visual Studio

☐ Eclipse

☒ XCode

☐ iOS Emulator Toolkit


 Il faut installer XCode, qui contient tous les éléments pour installer un simulateur iOS.

Exercice

On peut avoir plusieurs émulateurs avec différents iPhones allumés en même temps sur un ordinateur.

☒ Vrai

☐ Faux


 À condition que l'on ait suffisamment de mémoire RAM pour que cela tourne, c'est effectivement possible, et cela permet de tester différentes résolutions et versions du système d'exploitation pour notre application.

Exercice

Il est possible d'avoir plusieurs émulateurs avec différents Androids allumés en même temps sur un ordinateur.

☒ Vrai

☐ Faux

 Pareil que pour iPhone, c'est également possible sur Android : on peut simuler les terminaux de plusieurs fabricants.

Exercice

Pour faire tourner une application React Native sur un simulateur Android, j'ai minimum besoin de...

☐ JDK7

☒ JDK8

☐ JDK9

 Il faut le Java Development Kit 8 (JDK8) minimum.


Exercice

Android Studio, c'est...

☒ Un IDE pour développer des applications natives pour Android

☐ Un outil pour designer les maquettes de notre application

☐ La dernière version d'Android en 2020

 C'est effectivement un IDE, tout comme XCode, pour réaliser des applications natives Android en Java ou Kotlin.

Exercice

On a obligatoirement besoin d'un émulateur iOS ou Android et de builder une application pour développer avec React Native.

☐ Vrai

☒ Faux

☐ Grâce à Expo, il est possible de développer sans installer d'émulateur avec son téléphone personnel, et également sans processus de build.