React Native / Expo Go sur VPS

Contexte

Mise en place d'un environnement de développement React Native avec test en temps réel sur téléphone physique, hébergé sur un VPS Linux (AlmaLinux), sans recours à un service cloud payant.

Stack retenue

- **React Native** via **Expo** (template `blank`)
- **Expo Go** sur le téléphone pour le test en temps réel
- **screen** pour maintenir le serveur de dev en arrière-plan
- **nftables** pour la sécurisation du port de dev

Pourquoi React Native plutôt que PWA

- Accès aux APIs natives (Bluetooth, NFC, notifications push réelles, caméra)
- Rendu natif iOS/Android — performances supérieures sur animations et listes
- Distribution possible via App Store / Play Store
- La PWA reste limitée sur iOS (notifications, Bluetooth, comportement installé)

Pourquoi Expo Go plutôt que EAS Build ou RN CLI

	Expo Go	EAS Build	RN CLI
Test rapide	✅ scan QR	❌ ~10min build	❌ setup lourd
iOS sans Mac	✅	❌ (payant)	✅
Open source	✅	⚠ service proprio	✅
Gratuit	✅	⚠ 30 builds/mois	✅

Expo Go est suffisant pour la phase de développement. EAS Build sera utile pour générer un vrai APK/IPA en production.

Installation

Node.js sur AlmaLinux

```
curl -fsSL https://rpm.nodesource.com/setup_20.x | sudo bash -
sudo dnf install -y nodejs
```

Créer le projet Expo

```
npx create-expo-app@latest mon-projet --template blank
cd mon-projet
```

Le template `blank` évite les conflits de peer dependencies liés à `expo-router`. Point d'entrée unique : `App.js`.

Lancer le serveur de dev

Problème tunnel ngrok

`--tunnel` nécessite `@expo/ngrok` installé globalement. Si npm global est sans sudo, configurer d'abord :

```
mkdir -p ~/.npm-global
npm config set prefix '~/.npm-global'
echo 'export PATH=~/.npm-global/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

Alternative retenue : host LAN avec IP publique VPS

Le tunnel n'est pas nécessaire si le port est ouvert et filtré par IP. On force Expo à utiliser l'IP publique du VPS :

```
REACT_NATIVE_PACKAGER_HOSTNAME=IP_DU_VPS npx expo start
```

Metro écoute alors sur `0.0.0.0:8081`, accessible depuis le téléphone via `exp://IP_DU_VPS:8081`.

Vérifier que Metro écoute bien :

```
ss -tlnp | grep 8081
# Attendu : *:8081
```

Sécurisation du port 8081

Contexte firewall

Le VPS utilise **nftables** directement (firewalld masqué). Les règles sont dans `/etc/nftables.conf`, chargé automatiquement via `systemctl enable nftables`.

Principe

Autoriser le port 8081 uniquement depuis l'IP publique de la connexion de développement (box FAI). PC et téléphone sur la même box partagent la même IP publique sortante.

```
Le PC (177.x.x.x)
    ↓
La box (LAN: 177.x.x.x / WAN: 80.x.x.x)
    ↓
Internet (voit 80.x.x.x)
```

Récupérer son IP publique :

```
# Depuis le poste de dev
curl -4 ifconfig.me
```

Règle nftables à ajouter dans `chain input`

```
ip saddr IP_PUBLIQUE_FAI tcp dport 8081 accept comment "Expo dev"
```

Ajouter à chaud sans reboot :

```
sudo nft add rule inet filter input ip saddr IP_PUBLIQUE_FAI tcp dport 8081 accept comment '"Expo dev"'
```

Puis persister dans `/etc/nftables.conf`.

IP dynamique

Les box FAI ont souvent une IP dynamique. Si la connexion est refusée, vérifier que l'IP n'a pas changé et mettre à jour la règle :

```
# Trouver le handle de l'ancienne règle
sudo nft -a list ruleset | grep 8081

# Supprimer
sudo nft delete rule inet filter input handle NUMERO

# Ajouter la nouvelle IP
sudo nft add rule inet filter input ip saddr NOUVELLE_IP tcp dport 8081 accept comment '"Expo dev"'
```

Note sur IPv6

Expo ne supporte pas bien IPv6 pour binder son serveur. La règle IPv4 seule suffit. Ne pas ajouter de règle `ip6 saddr` pour ce port.

Risque IP spoofing

Le spoofing TCP nécessite d'intercepter le handshake SYN/SYN-ACK, ce qui requiert d'être sur le même segment réseau ou un ISP corrompu. Les ISPs sérieux filtrent le spoofing sortant (BCP38). Risque réel négligeable pour un usage de développement.

Lancer Expo en arrière-plan avec screen

Pourquoi screen plutôt que `&` ou `nohup`

- `&` : process tué à la déconnexion SSH (signal SIGHUP)
- `nohup` : survit à la déconnexion mais pas d'interaction possible après
- `screen` : survit à la déconnexion ET on peut se rattacher pour voir les logs

Installation / fix permissions

```
sudo dnf reinstall screen
# reinstall recrée /run/screen avec les bonnes permissions (777)
```

Utilisation

```
# Créer une session nommée
screen -S expo

# Lancer Expo dedans
REACT_NATIVE_PACKAGER_HOSTNAME=IP_DU_VPS npx expo start

# Détacher sans tuer : Ctrl+A puis D

# Se rattacher plus tard
screen -r expo

# Tuer proprement le serveur Metro
kill $(lsof -ti:8081)
```

Tester sur le téléphone

1. Installer **Expo Go** (App Store ou Play Store)
2. Lancer `npx expo start` sur le VPS
3. Scanner le QR code depuis Expo Go
4. L'app se charge en temps réel avec hot reload

En cas d'erreur "Something went wrong" dans Expo Go : appuyer sur l'erreur pour afficher les logs détaillés.

Structure du projet (template blank)

```
mon-projet/
├── App.js          ← composant principal, tout le code ici
├── app.json        ← config Expo (nom, bundle ID)
├── babel.config.js
├── index.js        ← point d'entrée (ne pas modifier)
└── assets/
```

Prochaines étapes

- Brancher une base de données (Supabase self-hosted)
- Générer un APK Android via EAS Build
- Gérer la persistance des données offline

Revision #8
Created 2026-05-03 17:30:04 UTC by Sanjy
Updated 2026-05-08 21:59:24 UTC by Sanjy