

Passer un projet Next.js en application mobile avec Capacitor

Sommaire

1. Présentation
2. Prérequis
3. Préparation du projet Next.js
4. Installation et initialisation de Capacitor
5. Configuration de Capacitor
6. Script de préparation pour les API TypeScript
7. Build et synchronisation
8. Tests et génération des apps
9. Ressources

Présentation

Ce tutoriel explique comment transformer un projet Next.js en application mobile native (iOS/Android) à l'aide de Capacitor.

Si votre Next.js intègre des API routes en TypeScript, Capacitor ne les supporte pas directement : vous devrez les exclure temporairement du build et les restaurer ensuite.

Prérequis

- Node.js ≥ 14 et npm ou yarn
- Next.js (dernier release)
- Capacitor CLI (`npm install @capacitor/cli @capacitor/core`)

- Android Studio et/ou Xcode pour émuler et builder

Préparation du projet Next.js

1. Vérifiez que votre projet build correctement en version web :

```
npm run build && npm run start
```

2. Si vous avez des API routes en TypeScript sous `app/api/`, notez leur emplacement :
Capacitor n'intègre que du contenu statique.

Installation et initialisation de Capacitor

1. Installez Capacitor dans votre projet :

```
npm install @capacitor/cli @capacitor/core
```

2. Initialisez Capacitor (répondez aux questions) :

```
npx cap init
```

- Nom de l'app (ex. MyNextApp)
- Identifiant (ex. com.mydomain.mynextapp)
- Chemin de sortie web : `out` (le dossier généré par `next export`)

Configuration de Capacitor

1. Dans `capacitor.config.ts` ou `capacitor.config.json`, vérifiez :

```
export default {  
  appId: 'com.mydomain.mynextapp',  
  appName: 'MyNextApp',  
}
```

```
webDir: 'out',  
bundledWebRuntime: false  
};
```

2. Ajoutez les plateformes souhaitées :

```
npx cap add android  
npx cap add ios
```

Script de préparation pour les API TypeScript

Créez un fichier `prepare-capacitor-build.sh` à la racine du projet :

```
#!/bin/bash  
# Script pour préparer le build pour Capacitor  
  
echo "📁 Préparation du build pour Capacitor..."  
  
# Créer un dossier temporaire pour sauvegarder les API routes  
echo "📁 Sauvegarde des API routes..."  
mkdir -p temp_api_backup  
mv app/api/* temp_api_backup/ 2>/dev/null || :  
  
# Exécuter le build  
echo "📁 Exécution du build..."  
npm run build  
  
# Restaurer les API routes  
echo "📁 Restauration des API routes..."  
mkdir -p app/api  
mv temp_api_backup/* app/api/ 2>/dev/null || :  
rmdir temp_api_backup  
  
echo "✅ Build terminé! Les fichiers statiques sont dans le dossier 'out'."  
echo "📁 Vous pouvez maintenant exécuter 'npx cap sync' pour synchroniser avec Capacitor."
```

Rendez-le exécutable :

```
chmod +x prepare-capacitor-build.sh
```

Build et synchronisation

1. Lancez le script :

```
./prepare-capacitor-build.sh
```

2. Synchronisez Capacitor avec les plateformes :

```
npx cap sync
```

Cela copie le contenu de `out/` dans `android/` et `ios/`.

Tests et génération des apps

- Android :

```
npx cap open android
```

Dans Android Studio, choisissez un émulateur ou un appareil, puis `Run`.

- iOS :

```
npx cap open ios
```

Dans Xcode, sélectionnez un simulateur ou un appareil, puis `Build and Run`.

Ressources

- Documentation Capacitor : <https://capacitorjs.com/docs>
 - GitHub Capacitor : <https://github.com/ionic-team/capacitor>
 - Next.js Export : <https://nextjs.org/docs/advanced-features/static-html-export>
 - Exemples et plugins : <https://github.com/ionic-team/capacitor/tree/main/example>
-

Revision #1

Created 11 May 2025 14:57:09 by qoyri

Updated 11 May 2025 15:14:36 by qoyri