

AtlasFreelance – Guide Complet du Développement Admin

Version: 1.0

Date: Décembre 2025

Projet: Panel d'Administration React Native

Rôle: Admin + Plateforme (Qualité, Sécurité, Livraison)

Table des matières

1. [Contexte du projet](#)
 2. [Architecture globale](#)
 3. [Fichiers créés étape par étape](#)
 4. [Explication détaillée du code](#)
 5. [Prochaines étapes](#)
 6. [Conventions de code](#)
-

1. Contexte du Projet {#contexte}

Vue d'ensemble

AtlasFreelance est une plateforme de freelancing pour le Maroc et l'Afrique du Nord, visant à connecter freelances et clients de manière sécurisée et transparente. Le rôle « Admin + Plateforme » couvre :

- **Outils d'administration** (Dashboard, Users, Projects, Disputes, Reports)
- **Observabilité** (Logs, monitoring, erreurs)
- **Qualité** (CI/CD, tests, conventions)
- **Sécurité** (Validation, sanitization, rate limiting)

Objectifs du MVP

1. Dashboard admin avec KPI (métriques clés)
2. Gestion des utilisateurs avec recherche
3. Gestion des litiges et résolutions
4. Reports/analytique simple
5. Logs centralisés et error boundaries

Stack technologique

- **Frontend mobile:** React Native (Expo)
 - **Frontend web:** React
 - **Backend:** Node.js + PostgreSQL
 - **Paiement:** Stripe (intégré)
 - **Navigaton:** React Navigation
-

2. Architecture Globale {#architecture}

Structure des dossiers

```
src/
├── screens/
│   ├── admin/
│   │   ├── AdminDashboardScreen.js
│   │   ├── AdminUsersScreen.js
│   │   ├── AdminProjectsScreen.js
│   │   ├── AdminDisputesScreen.js
│   │   └── AdminReportsScreen.js
│   ├── auth/
│   │   ├── AccueilScreen.js
│   │   └── InscriptionScreen.js
│   └── navigation/
│       ├── AdminStack.js
│       └── MainStack.js
├── components/
│   ├── admin/
│   │   ├── KPICard.js
│   │   ├── UserListItem.js
│   │   ├── DisputeCard.js
│   │   └── ReportChart.js
│   └── ui/
│       ├── CustomButton.js
│       └── SearchBar.js
├── services/
│   ├── adminApi.js
│   ├── loggingService.js
│   ├── notificationsService.js
│   └── authGuard.js
├── utils/
├── validation.js
├── formatters.js
└── security.js
```

App.js (point d'entrée)

3. Fichiers Créés Étape par Étape {#fichiers}

Étape 1 : Structure de base (App.js)

Fichier: App.js

Rôle: Point d'entrée, navigation principale entre Accueil, Inscription et Admin

```
// Ce fichier déclare le Stack Navigator avec 3 écrans :
// - Accueil : page d'accueil avec boutons
// - Inscription : formulaire d'inscription
// - Admin Dashboard : porte d'entrée au panel admin
```

Contenu clé:

- Import des dépendances (expo-status-bar, react-native, navigation)
- Stack Navigator global
- 3 composants de page (Accueil, Inscription)
- Import de AdminStack pour le panel admin
- Styles centralisés (container, title, subtitle)

Étape 2 : Dashboard Admin (AdminDashboardScreen.js)

Fichier: src/screens/admin/AdminDashboardScreen.js

Rôle: Affiche 3 KPI en cartes colorées pour vue d'ensemble rapide

Qu'est-ce que ça fait?

1. État local avec useState : stocke tableau des KPI
2. useEffect : charge les KPI au montage (mockés pour l'instant)
3. FlatList : affiche les 3 cartes KPI verticalement

KPI affichés:

- Projets actifs : 127 (vert #4CAF50)
- Litiges ouverts : 3 (orange #FF9800)
- Revenus mois : 45,200 MAD (bleu #7cb1ddff)

Rendu visuel:

127 (Vert)
Projets actifs

3 (Orange)
Litiges ouverts

45,200 MAD (Bleu)
Revenus mois

Étape 3 : Navigation Admin (AdminStack.js)

Fichier: src/navigation/AdminStack.js

Rôle: Stack sépaée pour tous les écrans d'admin

// Structure proposée (à créer) :

```
import { createStackNavigator } from '@react-navigation/stack';
```

```
const Stack = createStackNavigator();
```

```
export default function AdminStack() {
```

```
  return (
```

```
    <Stack.Navigator screenOptions={{ headerShown: true }}>
```

```
    <Stack.Screen name="AdminDashboard" component={AdminDashboardScreen} />
```

```
    <Stack.Screen name="AdminUsers" component={AdminUsersScreen} />
```

```

<Stack.Screen name="AdminProjects" component={AdminProjectsScreen} />
<Stack.Screen name="AdminDisputes" component={AdminDisputesScreen} />
</Stack.Navigator>
);
}

```

4. Explication Détaillée du Code {#explication}

4.1 App.js - Point d'entrée

Imports:

```

import { StatusBar } from 'expo-status-bar'; // Barre d'état
import { StyleSheet, Text, View, Button } from 'react-native'; // Composants
import React from 'react';
import { NavigationContainer } from '@react-navigation/native'; // Conteneur navigation
import { createNativeStackNavigator } from '@react-navigation/native-stack'; // Stack natif
import AdminStack from './src/navigation/AdminStack'; // Panel admin complet

```

Composant Accueil:

```

function Accueil({ navigation }) {
  // navigation vient automatiquement du Stack Navigator
  return (
    <View style={styles.container}>
      <Text>Atlas Freelance</Text>
      <Button title="Espace Admin"
        onPress={() => navigation.navigate('Admin Dashboard')} />
    </View>
  );
}

```

- { navigation } : destructuring de la prop injectée par React Navigation
- navigation.navigate('Admin Dashboard') : change d'écran vers le panel admin

Stack.Navigator:

```

<Stack.Navigator>
  <Stack.Screen name="Accueil" component={Accueil} />
  <Stack.Screen name="Inscription" component={Inscription} />
  <Stack.Screen name="Admin Dashboard" component={AdminStack} />
</Stack.Navigator>

```

- 3 écrans au niveau principal
- "Admin Dashboard" charge tout le AdminStack (qui contient Dashboard, Users, etc.)

4.2 AdminDashboardScreen.js - Dashboard avec KPI

État et données:

```

const [kpis, setKpis] = useState([]); // Tableau vide au départ

useEffect(() => {
  setKpis([
    { title: 'Projets actifs', value: 127, color: '#4CAF50' },
    { title: 'Litiges ouverts', value: 3, color: '#FF9800' },
  ]);
}, []);

```

```
{ title: 'Revenus mois', value: '45,200 MAD', color: '#7cb1ddff' },  
]);  
}, []); // [] = s'exécute UNE FOIS au montage
```

Breakdown:

- `useState([])` : crée un state pour les KPI, initialement vide
- `useEffect(..., [])` : hook React qui s'exécute après le rendu (une seule fois avec [])
- Dans `useEffect`, on remplit les KPI avec des données mockées (à remplacer par API plus tard)

FlatList - Affichage en liste:

```
<FlatList  
data={kpis} // Les 3 objets KPI  
renderItem={({ item }) => ( // Pour CHAQUE KPI, affiche une carte  
  
  <Text style={{ fontSize: 36, color: item.color }}>  
    {item.value} // Affiche 127, 3 ou "45,200 MAD"  
  </Text> {item.title} // Affiche le titre  
  
  )}  
keyExtractor={...} // Clé unique pour React  
</>
```

Styles appliqués:

```
backgroundColor: 'white', // Carte blanche  
padding: 16, // Espacement interne  
marginBottom: 12, // Espace entre cartes  
borderRadius: 8, // Coins arrondis  
shadowColor: '#000', // Ombre  
shadowOpacity: 0.1,  
shadowRadius: 4,
```

4.3 Navigation avec `useNavigation` (futur)

Pour les boutons DANS `AdminDashboardScreen`, utiliser le hook:

```
import { useNavigation } from '@react-navigation/native';  
  
export default function AdminDashboardScreen() {  
  const navigation = useNavigation(); // ← Hook pour accéder à navigation  
  
  return (  
    <Button  
      title="Aller aux disputes"  
      onPress={() => navigation.navigate('AdminDisputes')}  
    </>  
  );  
}
```

Différence clé:

- Dans `App.js` (écrans du Stack): `navigation` vient comme prop

- Dans AdminDashboardScreen (écran importé): utilise useNavigation() hook
-

5. Prochaines Étapes {#prochaines}

Phase 1 (Semaine 1) : Fondations

- ☒ Dashboard avec KPI mockées
- ☐ AdminStack complet (navigation entre écrans admin)
- ☐ AdminUsersScreen avec recherche
- ☐ Logger + Error Boundary

Phase 2 (Semaine 2) : Données réelles

- ☐ Brancher les KPI à une API /admin/kpis
- ☐ AdminProjectsScreen avec filtres
- ☐ AdminDisputesScreen avec résolution
- ☐ Service adminApi.js

Phase 3 (Semaine 3) : Qualité

- ☐ Reports avec graphiques (recharts ou victory-native)
- ☐ Settings (feature flags, paramètres plateforme)
- ☐ Notifications admin
- ☐ ESLint + Prettier
- ☐ E2E tests (Maestro/Detox)

Phase 4 (Semaine 4) : Déploiement

- ☐ CI/CD EAS (Expo Application Services)
 - ☐ Sentry pour crash reporting
 - ☐ Build preview et release channels
 - ☐ Documentation de déploiement
-

6. Conventions de Code {#conventions}

6.1 Nommage

```
// Composants : PascalCase
function AdminDashboardScreen() { }
function KPICard() { }

// Variables/fonctions : camelCase
const [kpis, setKpis] = useState([]);
const fetchKpis = () => { };
const filteredUsers = users.filter(...);
```

6.2 Structure de dossier

```
src/  
├── screens/ # Écrans complets  
├── components/ # Composants réutilisables  
├── navigation/ # Stacks et navigateurs  
├── services/ # API, logging, auth  
└── utils/ # Helpers, validation
```

6.3 Git workflow

Main : production

Develop : intégration

feat/admin-dashboard : branche feature

```
git checkout -b feat/admin-users
```

... modifications ...

```
git push origin feat/admin-users
```

**Créer PR avec description, screenshots,
checklist QA**

6.4 PR Checklist

- ☐ Tests passent (si applicable)
- ☐ ESLint + Prettier OK
- ☐ Screenshots des changements UI
- ☐ Pas de console.log en production
- ☐ Types/validations en place
- ☐ Documentation mise à jour

7. Fichiers Clés – Code Complet

App.js (complet)

```
import { StatusBar } from 'expo-status-bar';  
import { StyleSheet, Text, View, Button } from 'react-native';  
import React from 'react';  
import { NavigationContainer } from '@react-navigation/native';  
import { createNativeStackNavigator } from '@react-navigation/native-stack';  
import AdminStack from './src/navigation/AdminStack';
```

```

const Stack = createNativeStackNavigator();

function Accueil({ navigation }) {
  return (
    Atlas Freelance Trouve des missions, collabore, facture
    <Button title="S'inscrire" onPress={() => navigation.navigate('Inscription')} />
    <Button title="Espace Admin" onPress={() => navigation.navigate('Admin Dashboard')} />

    );
}

function Inscription() {
  return (
    Inscription Formulaire à venir..
  );
}

export default function App() {
  return (

    <Stack.Navigator>
    <Stack.Screen name="Accueil" component={Accueil} />
    <Stack.Screen name="Inscription" component={Inscription} />
    <Stack.Screen name="Admin Dashboard" component={AdminStack} />
    </Stack.Navigator>

  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#e4e8ecff',
  },
  title: {
    fontSize: 32,
    fontWeight: 'bold',
    color: '#1e293b',
  },
  subtitle: {
    fontSize: 16,
    color: '#64748b',
    marginTop: 8,
    textAlign: 'center',
  },
});

```


AdminDashboardScreen.js (complet)

```
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList } from 'react-native';

export default function AdminDashboardScreen() {
  const [kpis, setKpis] = useState([]);

  useEffect(() => {
    setKpis([
      { title: 'Projets actifs', value: 127, color: '#4CAF50' },
      { title: 'Litiges ouverts', value: 3, color: '#FF9800' },
      { title: 'Revenus mois', value: '45,200 MAD', color: '#7cb1d1' },
    ]);
  }, []);

  return (
    <View style={{ padding: 16 }}>
      <Text style={{ fontSize: 24, fontWeight: 'bold', marginBottom: 16 }}>
        Dashboard Admin
      </Text>
      <FlatList
        data={kpis}
        renderItem={({ item }) => (
          <View style={{
            backgroundColor: 'white',
            padding: 16,
            marginBottom: 12,
            borderRadius: 8,
            shadowColor: '#000',
            shadowOpacity: 0.1,
            shadowRadius: 4,
          }}>
            <Text style={{ fontSize: 36, fontWeight: 'bold', color: item.color }}>
              {typeof item.value === 'number' ? item.value : item.value}
            </Text> {item.title}
          </View>
        )}
        keyExtractor={(item, index) => index.toString()}
      />
    </View>
  );
}
```

8. Checklist – Prêt pour la prochaine étape?

- [x] App.js avec navigation vers Admin
- [x] AdminDashboardScreen avec 3 KPI
- [] AdminStack complet (navigation interne admin)
- [] AdminUsersScreen avec recherche
- [] Service adminApi (fetch KPI réels)

- [] Logger centralisé
- [] Error Boundary
- [] Tests basiques
- [] ESLint + Prettier
- [] Build EAS preview
- [] Sentry configuré

9. Ressources et Références

- **React Navigation:** <https://reactnavigation.org/docs/getting-started>
- **React Native Docs:** <https://reactnative.dev>
- **Expo:** <https://docs.expo.dev>
- **EAS Build:** <https://docs.expo.dev/build/introduction>
- **Sentry React Native:** <https://docs.sentry.io/platforms/react-native>

Guide créé: Décembre 2025

Version: 1.0

Auteur: Équipe AtlasFreelance Admin

Status: Draft – Prêt pour phase 2