# Cahier de Charge Technique
# SpontyTrip

Sommaire



## 1. Vue d'ensemble du projet

### 1.1 Description générale

**SpontyTrip** est une application mobile collaborative de gestion de voyages développée avec React Native et Expo. Elle permet aux voyageurs de planifier, organiser et partager leurs aventures en temps réel avec une interface moderne et intuitive.

### 1.2 Objectifs principaux

- Collaboration : Permettre à plusieurs utilisateurs de co-organiser un voyage
- Temps réel : Synchronisation instantanée des données entre tous les participants
- Simplicité : Interface utilisateur intuitive et moderne
- Mobilité : Application mobile native cross-platform (iOS/Android)
- Fiabilité : Gestion offline et synchronisation automatique

## 1.3 Public cible

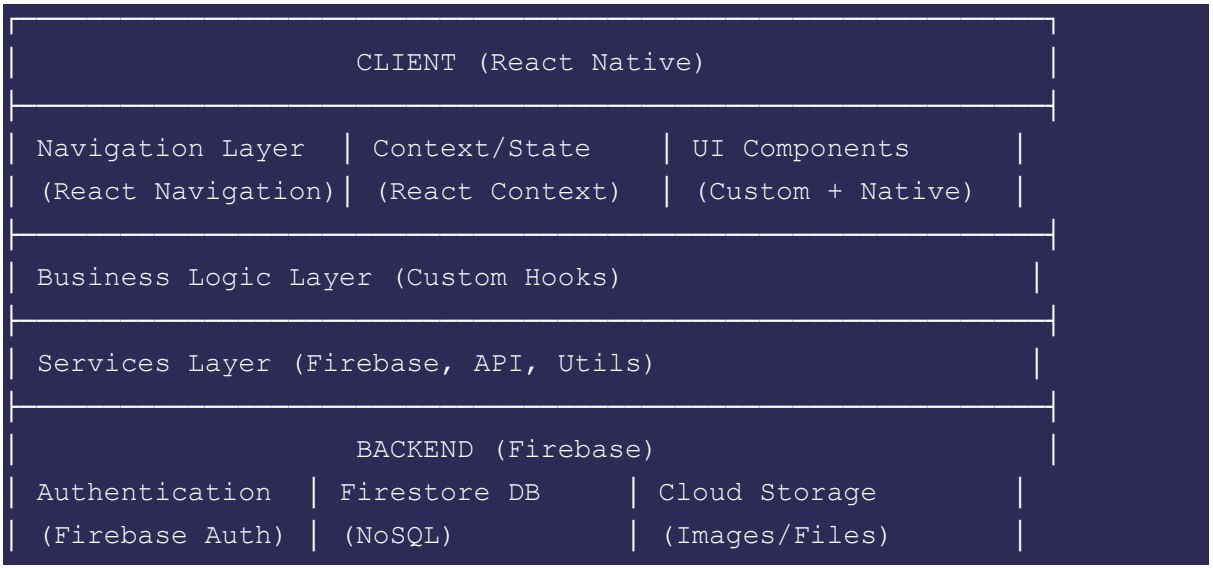
- Voyageurs en groupe (familles, amis, collègues)
- Organisateurs d'événements
- Communautés de voyage
- Utilisateurs entre 18 et 65 ans

## 1.4 Plateformes supportées

- iOS : Version 13.0 et supérieure
- Android : Version 6.0 (API 23) et supérieure
- Web : Version responsive (support secondaire)

# 2. Architecture technique

## 2.1 Architecture générale



```
└──────────────────────────────────────────────────────────────┘
```

## 2.2 Structure des dossiers

```
spontytrip/
├── api/                    # Services API externes
├── assets/                 # Images, fonts, icônes
├── components/             # Composants React Native
│   ├── activities/         # Composants d'activités
│   ├── checklist/          # Composants de checklist
│   ├── createTrip/         # Composants de création voyage
│   ├── expenses/           # Composants de dépenses
│   ├── home/               # Composants page d'accueil
│   ├── notes/              # Composants de notes
│   └── tripDetails/        # Composants détails voyage
├── config/                 # Configuration Firebase
├── constants/              # Constantes globales
├── contexts/               # Contexts React
├── hooks/                  # Hooks personnalisés
├── navigation/             # Configuration navigation
├── screens/                # Écrans de l'application
├── services/               # Services métier
├── styles/                 # Styles et thèmes
├── tests/                  # Tests unitaires
├── types/                  # Types TypeScript
└── utils/                  # Utilitaires
```

## 2.3 Patterns architecturaux

- MVVM : Séparation Model-View-ViewModel avec hooks
- Context API : Gestion d'état global pour l'authentification
- Custom Hooks: Logique métier réutilisable
- Component Composition : Composants modulaires et réutilisables

- Service Layer : Abstraction des services externes

# 3. Technologies et dépendances

## 3.1 Stack technologique principal

| Technologie | Version | Rôle |
| --- | --- | --- |
| React Native | 0.79.4 | Framework mobile |
| Expo | 53.0.11 | SDK et outils de développement |
| TypeScript | 5.8.3 | Typage statique |
| Firebase | 8.10.1 | Backend-as-a-Service |
| React Navigation | 7.x | Navigation |

## 3.2 Dépendances principales

### 3.2.1 Interface utilisateur

- @expo/vector-icons : Icônes vectorielles
- expo-linear-gradient : Gradients
- expo-blur : Effets de flou
- react-native-svg : Graphiques vectoriels

### 3.2.2 Navigation

- @react-navigation/native : Navigation de base
- @react-navigation/stack : Navigation en pile
- @react-navigation/bottom-tabs : Navigation par onglets

### 3.2.3 Médias et interaction

- expo-image-picker : Sélection d'images
- expo-haptics : Retour haptique
- expo-sharing : Partage de contenu
- expo-clipboard : Presse-papiers

### 3.2.4 Utilitaires

- `@react-native-async-storage/async-storage` : Stockage local
- `react-native-calendars` : Calendriers
- `axios` : Requêtes HTTP

## 3.3 Outils de développement

- Jest : Tests unitaires
- ESLint : Analyse statique du code
- Expo CLI: Outils de développement
- Metro: Bundler JavaScript

# 4. Spécifications fonctionnelles

## 4.1 Gestion des utilisateurs

### 4.1.1 Authentification

- Inscription : Email/mot de passe avec validation
- Connexion : Authentification Firebase
- Déconnexion : Avec confirmation utilisateur
- Mot de passe oublié : Réinitialisation par email
- Suppression de compte : Conforme RGPD

### 4.1.2 Profil utilisateur

- Informations personnelles : Nom, email, photo
- Préférences : Notifications, thème
- Sécurité: Changement de mot de passe
- Historique : Voyages passés

## 4.2 Gestion des voyages

### 4.2.1 Création de voyage

- Informations de base : Titre, destination, dates
- Détails : Description, type, image de couverture
- Paramètres : Visibilité, code d'invitation
- Validation : Vérification des champs obligatoires

### 4.2.2 Participation au voyage

- Rejoindre : Via code d'invitation
- Membres : Gestion des participants
- Rôles : Créateur vs membre
- Permissions : Lecture/écriture selon le rôle

### 4.2.3 Gestion du voyage

- Titre, description, dates
- Avec confirmation
- Voyages terminés
- Temps réel entre participants

## 4.3 Checklist collaborative

### 4.3.1 Fonctionnalités principales

- Templates prédéfinis: Plage, montagne, citytrip, campagne
- Création d'items : Titre, description, catégorie
- Attribution : Assignation à un participant
- Statuts : En cours, terminé, en retard

### 4.3.2 Gestion des tâches

- Ajout/modification : Interface intuitive
- Suppression : Avec confirmation
- Complétion : Marquer comme terminé
- Filtrage: Par statut, assigné, catégorie

## 4.4 Gestion des dépenses

### 4.4.1 Suivi des dépenses

- Ajout de dépense : Montant, description, participants
- Répartition : Automatique ou personnalisée
- Catégories : Transport, hébergement, repas, activités
- Devises : Conversion automatique

### 4.4.2 Calculs financiers

- Soldes individuels : Doit/reçoit par participant
- Remboursements : Calcul automatique des dettes
- Résumé : Vue d'ensemble des finances
- Historique : Toutes les transactions

## 4.5 Notes partagées

### 4.5.1 Création et partage

- Création : Notes textuelles

- Partage : Visible par tous les participants
- Modification : Édition collaborative
- Suppression : Avec confirmation

### 4.5.2 Organisation

- Recherche : Recherche textuelle
- Filtrage : Par auteur, date
- Tri : Chronologique, alphabétique
- Archivage : Notes importantes

## 4.6 Activités

### 4.6.1 Planification

- Création : Titre, description, date, lieu
- Détails : Horaires, prix, liens
- Catégories : Visite, restaurant, transport, etc.
- Validation : Système de votes

### 4.6.2 Gestion

- Timeline : Vue chronologique
- Filtres : Par date, catégorie, statut
- Modifications : Édition collaborative
- Statuts : Planifié, en cours, terminé

# 5. Modèle de données

## 5.1 Structure Firestore

### 5.1.1 Collection `users`

```typescript
interface User {
    id: string;
    email: string;
    displayName: string;
    avatar?: string;
    createdAt: Date;
    bio?: string;
    location?: string;
}
```

#### 5.1.2 Collection `trips`

```typescript
interface Trip {
    id: string;
    title: string;
    destination: string;
    startDate: Date;
    endDate: Date;
    description?: string;
    type: "plage" | "montagne" | "citytrip" | "campagne";
    coverImage?: string;
    creatorId: string;
    creatorName: string;
    inviteCode: string;
    members: TripMember[];
    memberIds: string[];
    createdAt: Date;
    updatedAt: Date;
}
```

#### 5.1.3 Collection `checklists`

```typescript
interface ChecklistItem {
    id: string;
    tripId: string;
    title: string;
```

```
    description?: string;
    category: string;
    assignedTo?: string;
    isCompleted: boolean;
    completedBy?: string;
    completedAt?: Date;
    createdBy: string;
    createdAt: Date;
}
```

#### 5.1.4 Collection `expenses`

```typescript
interface ExpenseItem {
    id: string;
    tripId: string;
    label: string;
    amount: number;
    paidBy: string;
    paidByName: string;
    participants: string[];
    participantNames: string[];
    date: Date;
    createdBy: string;
    createdAt: Date;
}
```

#### 5.1.5 Collection `activities`

```typescript
interface Activity {
    id: string;
    tripId: string;
    title: string;
    description?: string;
    location?: string;
    date: Date;
    startTime?: string;
    endTime?: string;
    createdBy: string;
    createdByName: string;
```

```
    votes: string[];
    validated: boolean;
    category: string;
    price?: number;
    currency?: string;
}
```

## 5.2 Relations entre entités

```
```
User (1) ⟷ (N) Trip
Trip (1) ⟷ (N) ChecklistItem
Trip (1) ⟷ (N) ExpenseItem
Trip (1) ⟷ (N) Activity
Trip (1) ⟷ (N) TripNote
```
```

## 5.3 Index Firestore

```
```json
{
    "indexes": [
        {
            "collectionGroup": "trips",
            "queryScope": "COLLECTION",
            "fields": [
                { "fieldPath": "memberIds", "arrayConfig": "CONTAINS"
},
                { "fieldPath": "updatedAt", "order": "DESCENDING" }
            ]
        },
        {
            "collectionGroup": "checklists",
            "queryScope": "COLLECTION",
```
```

```
            "fields": [
                { "fieldPath": "tripId", "order": "ASCENDING" },
                { "fieldPath": "createdAt", "order": "DESCENDING" }
            ]
        }
    ]
}
```

# 6. Sécurité et authentification

## 6.1 Authentification Firebase

### 6.1.1 Méthodes supportées

- Email/Password : Méthode principale
- Réinitialisation : Par email sécurisé
- Suppression : Avec confirmation double

### 6.1.2 Validation côté client

- Format email : Validation regex
- Complexité mot de passe : Minimum 8 caractères
- Confirmation : Double saisie
- Feedback : Messages d'erreur localisés

## 6.2 Règles de sécurité Firestore

### 6.2.1 Principe général

- Authentification obligatoire : Toutes les opérations
- Autorisation granulaire : Par collection et document
- Validation des données : Côté serveur

### 6.2.2 Règles spécifiques

```javascript
// Trips - Lecture : membres uniquement
allow read: if isAuthenticated() && canAccessTrip(tripId);

// Trips - Écriture : créateur uniquement
allow write: if isAuthenticated() && isTripCreator(tripId);

// Profils - Lecture : tous, Écriture : propriétaire
allow read: if isAuthenticated();
allow write: if isAuthenticated() && getUserId() == userId;
```

## 6.3 Protection des données

### 6.3.1 Données sensibles

- Mots de passe : Hashage Firebase
- Tokens : Gestion automatique
- Données personnelles : Chiffrement transport

### 6.3.2 Conformité RGPD

- Suppression : Suppression complète des données
- Portabilité : Export des données utilisateur
- Consentement : Acceptation explicite

# 7. Interface utilisateur

## 7.1 Système de design

### 7.1.1 Couleurs principales

````
####

```typescript
export const Colors = {
    primary: "#4DA1A9", // Bleu-vert principal
    secondary: "#7ED957", // Vert secondaire
    accent: "#6366F1", // Violet accent
    background: "#F8FAFC", // Fond clair
    surface: "#FFFFFF", // Surface blanche
    text: "#1E293B", // Texte principal
    textSecondary: "#64748B", // Texte secondaire
    error: "#EF4444", // Erreur
    warning: "#F59E0B", // Avertissement
    success: "#10B981", // Succès
};
```
````

### 7.1.2 Typographie

- Famille : Inter (système par défaut)
- Tailles : 12px à 32px
- Poids : Regular, Medium, SemiBold, Bold
- Hauteur de ligne : 1.5x par défaut

### 7.1.3 Espacement

```typescript
export const Spacing = {
    xs: 4, // Très petit
    sm: 8, // Petit
    md: 16, // Moyen
    lg: 24, // Grand
    xl: 32, // Très grand
    xxl: 48, // Extra grand
};
```

## 7.2 Composants d'interface

### 7.2.1 Composants de base

- Button : Boutons avec variants
- Card : Cartes avec ombres
- Modal : Modales avec animations
- Input : Champs de saisie stylisés

### 7.2.2 Composants spécialisés

- TripCard : Carte de voyage
- ChecklistItem : Item de checklist
- ExpenseCard : Carte de dépense
- ActivityCard : Carte d'activité

## 7.3 Navigation

### 7.3.1 Structure de navigation

```
AuthNavigator
├── Login
├── Register
└── ForgotPassword

MainAppNavigator (Tab)
├── Home
├── MyTrips
├── Discover
├── Explore
└── Profile

StackNavigator
├── TripDetails
├── CreateTrip
├── Checklist
├── Expenses
└── Activities
```

### 7.3.2 Animations

- Transitions : Slide, fade, scale
- Durée: 300ms par défaut
- Courbes : Ease-in-out
- Retour haptique : iOS/Android

# 8. Performance et optimisations

## 8.1 Optimisations React Native

### 8.1.1 Rendu des listes

- FlatList : Listes virtualisées
- WindowSize : Optimisation mémoire
- GetItemLayout : Calcul prédictif des tailles
- KeyExtractor : Clés optimisées

### 8.1.2 Gestion des images

- Expo Image : Mise en cache avancée
- Compression : Réduction taille avant upload
- Lazy Loading : Chargement différé
- Placeholder : Images de substitution

### 8.1.3 État et re-renders

- useMemo: Mémorisation des calculs
- useCallback : Stabilité des fonctions
- React.memo : Composants mémorisés
- Context optimisé : Séparation des contextes

## 8.2 Optimisations Firebase

### 8.2.1 Requêtes Firestore

- Indexes composés : Requêtes complexes
- Pagination : Limit et startAfter
- Offline : Mise en cache locale
- Batch operations : Opérations groupées

### 8.2.2 Gestion hors ligne

- Persistance : Données locales
- Synchronisation : Merge automatique
- Conflits : Résolution côté serveur

- Retry logic : Tentatives automatiques

### 8.3 Optimisations générales

#### 8.3.1 Bundle size

- Tree shaking : Suppression code mort
- Code splitting : Division du code
- Lazy imports : Chargement différé
- Asset optimization : Compression médias

#### 8.3.2 Runtime performance

- Debouncing : Limitation des appels API
- Throttling : Limitation des événements
- Memoization : Cache des résultats
- Background tasks : Tâches asynchrones

# 9. Configuration et déploiement

## 9.1 Configuration de l'environnement

### 9.1.1 Variables d'environnement

```bash
# Firebase Configuration
FIREBASE_API_KEY=your_api_key
FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com
FIREBASE_PROJECT_ID=your_project_id
FIREBASE_STORAGE_BUCKET=your_project.appspot.com
```

```
FIREBASE_MESSAGING_SENDER_ID=123456789
FIREBASE_APP_ID=1:123456789:web:abcdef

# API Keys
GOOGLE_MAPS_API_KEY=your_maps_api_key
CURRENCY_API_KEY=your_currency_api_key

# Environment
NODE_ENV=production
```

### 9.1.2 Configuration Firebase

```
```javascript
// config/firebase.ts
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";
import { getStorage } from "firebase/storage";

const firebaseConfig = {
    apiKey: process.env.FIREBASE_API_KEY,
    authDomain: process.env.FIREBASE_AUTH_DOMAIN,
    projectId: process.env.FIREBASE_PROJECT_ID,
    storageBucket: process.env.FIREBASE_STORAGE_BUCKET,
    messagingSenderId: process.env.FIREBASE_MESSAGING_SENDER_ID,
    appId: process.env.FIREBASE_APP_ID,
};

export const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
export const storage = getStorage(app);
```
```

## 9.2 Processus de déploiement

### 9.2.1 Environnements

- Development : Développement local
- Staging : Tests et validation
- Production : Version publique

### 9.2.2 Pipeline CI/CD

```yaml
# .github/workflows/deploy.yml
name: Deploy to Expo
on:
  push:
    branches: [main]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm ci
      - run: npm test
      - run: expo publish
        env:
          EXPO_TOKEN: ${{ secrets.EXPO_TOKEN }}
```

# 10. Roadmap et évolutions

## 10.1 Fonctionnalités prévues

### 10.1.1 Version 1.1 (Q1 2025)

- Notifications push : Alertes temps réel
- Mode hors ligne : Synchronisation différée
- Thème sombre : Support natif
- Langues : Français, Anglais, Espagnol

### 10.1.2 Version 1.2 (Q2 2024)

- Intégrations : Google Maps, Booking
- IA : Suggestions d'activités
- Partage social : Facebook, Instagram
- Widgets : Écran d'accueil

## 10.2 Améliorations techniques

### 10.2.1 Performance

- React Native 0.74 : Nouvelle architecture
- Hermes : Moteur JavaScript optimisé
- Flipper*: Outils de debug avancés
- Expo Route : Navigation nouvelle génération

### 10.2.2 Sécurité

- Biométrie : Authentification Touch/Face ID
- Chiffrement : Données sensibles
- Audit : Sécurité trimestriel
- Conformité: RGPD, CCPA

## 10.3 Métriques de succès

### 10.3.1 Utilisateurs

- MAU : 10,000 utilisateurs actifs mensuels
- Rétention : 70% à 7 jours
- Engagement : 15 minutes par session
- NPS : Score >8/10

### 10.3.2 Technique

- Performance : <1s temps de chargement
- Stabilité** : <0.5% taux de crash
- Couverture: >90% tests
- Disponibilité : 99.95% uptime

-

# Conclusion

Ce cahier de charge technique définit l'architecture complète et les spécifications détaillées de SpontyTrip, une application mobile collaborative de gestion de voyages. L'architecture basée sur React Native et Firebase offre une solution scalable, sécurisée et performante pour répondre aux besoins des voyageurs modernes.

Les choix technologiques privilégient la maintenabilité, la sécurité et l'expérience utilisateur, tout en garantissant une évolutivité future pour l'ajout de nouvelles fonctionnalités et l'expansion vers de nouveaux marchés.