Documentation technique

Documentation Technique - PlanFlow

Stack : React Native

Table des Matières

1. Vue d’ensemble
2. Architecture du système
3. Technologies et dépendances
4. Structure des données
5. API et Services
6. Gestion de l’état
7. Navigation
8. Composants
9. Authentification et sécurité
10. Gestion des erreurs
11. Performance et optimisation
12. Tests
13. Maintenance

---

1. Vue d’ensemble

1.1 Description du projet

Application mobile clone de Trello développée avec React Native et Expo, intégrant l’API REST officielle de Trello pour la gestion complète de projets collaboratifs.

1.2 Objectifs techniques

- Architecture modulaire et maintenable
- Intégration complète de l’API Trello
- Gestion d’état locale avec AsyncStorage
- Navigation fluide et intuitive
- Interface utilisateur responsive

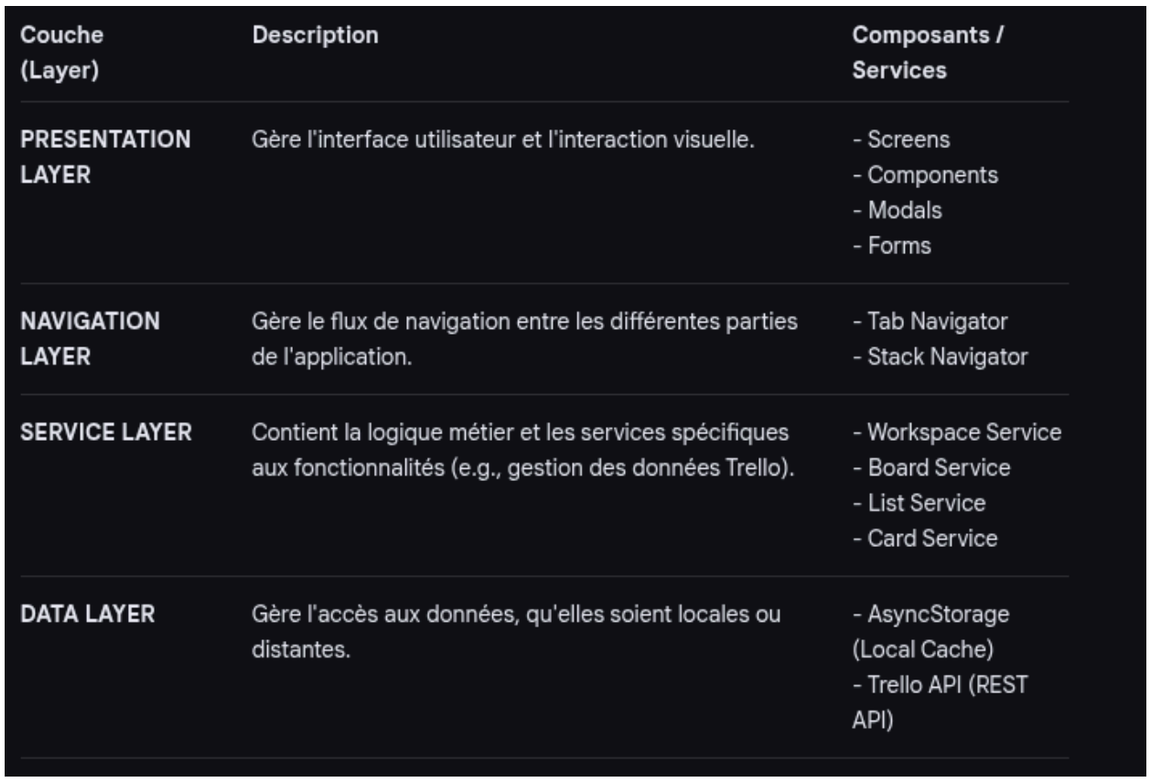
1.3 Stack technique

Catégorie	Technologie	Version
Framework	React Native	0.74+

Catégorie	Technologie	Version
Runtime	Expo	51.0+
Langage	JavaScript (ES6+)	-
Navigation	React Navigation	6.x
Stockage	AsyncStorage	1.23+
API	Trello REST API	v1

2. Architecture du système

2.1 Architecture globale



Couche (Layer)	Description	Composants / Services
PRESENTATION LAYER	Gère l'interface utilisateur et l'interaction visuelle.	- Screens - Components - Modals - Forms
NAVIGATION LAYER	Gère le flux de navigation entre les différentes parties de l'application.	- Tab Navigator - Stack Navigator
SERVICE LAYER	Contient la logique métier et les services spécifiques aux fonctionnalités (e.g., gestion des données Trello).	- Workspace Service - Board Service - List Service - Card Service
DATA LAYER	Gère l'accès aux données, qu'elles soient locales ou distantes.	- AsyncStorage (Local Cache) - Trello API (REST API)

2.2 Flux de données

```
User Action → Screen → Service → API Call → Response → Update UI
```

2.3 Pattern architectural

- **Séparation des responsabilités** : Services, Composants, Navigation
- **Modularité** : Chaque entité (Workspace, Board, List, Card) a son propre service
- **Scalabilité** : Architecture permettant l'ajout facile de nouvelles fonctionnalités

3. Technologies et dépendances

3.1 Dépendances principales

```
"dependencies": {
  "@expo/metro-runtime": "^6.1.2",
  "@react-native-async-storage/async-storage": "2.2.0",
  "@react-native-picker/picker": "^2.11.4",
  "@react-navigation/bottom-tabs": "^7.8.5",
  "@react-navigation/native": "^7.1.20",
  "@react-navigation/native-stack": "^7.6.3",
  "@react-navigation/stack": "^7.6.4",
  "expo": "~54.0.25",
  "expo-linking": "~8.0.9",
  "expo-status-bar": "~3.0.8",
  "react": "19.1.0",
  "react-dom": "^19.1.0",
  "react-native": "0.81.5",
  "react-native-gesture-handler": "~2.28.0",
  "react-native-safe-area-context": "~5.6.0",
  "react-native-screens": "~4.16.0",
  "react-native-web": "^0.21.2",
  "react-navigation": "^5.0.0"
},
```

3.2 Environnement de développement

- **Node.js** : v18.x ou supérieur
- **npm** : v9.x ou supérieur
- **Expo CLI** : Installé automatiquement
- **Expo Go** : Application mobile pour les tests

3.3 APIs externes

Trello REST API v1

- **Base URL** : `https://api.trello.com/1`
- **Authentification** : API Key + Token OAuth
- **Format** : JSON
- **Documentation** : https://developer.atlassian.com/cloud/trello/rest/

---

4. Structure des données

4.1 Modèle de données Trello

Workspace (Organization)

```
{
  id: "string",  displayName: "string",  desc: "string",  url: "string",  membership
```

```
s: Array}
```

Board

```
{
 id: "string", name: "string", desc: "string", closed: boolean, idOrganization:
"string", url: "string", prefs: Object}
```

List

```
{
 id: "string", name: "string", closed: boolean, idBoard: "string", pos: number
}
```

Card

```
{
 id: "string", name: "string", desc: "string", idList: "string", idMembers: Arra
y<string>, due: "string|null", pos: number, url: "string"}
```

Member

```
{
 id: "string", fullName: "string", username: "string", avatarUrl: "string"}
```

4.2 Stockage local

Structure AsyncStorage

```
{
 "trello_token": "string" // Token d'authentification OAuth}
```

---

5. API et Services

5.1 Architecture des services

Tous les services suivent le même pattern :

```
// Pattern génériqueconst service = {

 getAll: async () ⇒ { /* ... */ },
  getById: async (id) ⇒ { /* ... */ },
   create: async (data) ⇒ { /* ... */ },
    update: async (id, data) ⇒ { /* ... */ },
```

```
    delete: async (id) => { /* ... */ }
}
```

5.2 Workspace Service

Fichier : `services/workspaceService.js`

Méthodes

Méthode	Endpoint	Description
`getAll()`	`GET /members/me/organizations`	Récupère tous les workspaces
`getById(id)`	`GET /organizations/{id}`	Récupère un workspace spécifique
`create(data)`	`POST /organizations`	Crée un nouveau workspace
`update(id, data)`	`PUT /organizations/{id}`	Met à jour un workspace
`delete(id)`	`DELETE /organizations/{id}`	Supprime un workspace

5.3 Board Service

Fichier : `services/boardService.js`

Méthodes

Méthode	Endpoint	Description
`getByWorkspace(workspaceId)`	`GET /organizations/{id}/boards`	Récupère les boards d'un workspace
`getById(id)`	`GET /boards/{id}`	Récupère un board spécifique
`create(data)`	`POST /boards`	Crée un nouveau board
`update(id, data)`	`PUT /boards/{id}`	Met à jour un board
`delete(id)`	`DELETE /boards/{id}`	Supprime un board
`getTemplates()`	Local	Retourne les templates disponibles

5.4 List Service

Fichier : `services/listService.js`

Méthodes

Méthode	Endpoint	Description
`getByBoard(boardId)`	`GET /boards/{id}/lists`	Récupère les listes d'un board
`getById(id)`	`GET /lists/{id}`	Récupère une liste spécifique
`create(data)`	`POST /lists`	Crée une nouvelle liste
`update(id, data)`	`PUT /lists/{id}`	Met à jour une liste
`archive(id)`	`PUT /lists/{id}/closed`	Archive une liste

5.5 Card Service

Fichier : `services/cardService.js`

Méthodes

Méthode	Endpoint	Description
`getByList(listId)`	`GET /lists/{id}/cards`	Récupère les cartes d'une liste
`getById(id)`	`GET /cards/{id}`	Récupère une carte spécifique
`create(data)`	`POST /cards`	Crée une nouvelle carte
`update(id, data)`	`PUT /cards/{id}`	Met à jour une carte
`delete(id)`	`DELETE /cards/{id}`	Supprime une carte
`assignMember(cardId, memberId)`	`POST /cards/{id}/idMembers`	Assigne un membre
`removeMember(cardId, memberId)`	`DELETE /cards/{id}/idMembers/{memberId}`	Retire un membre
`getBoardMembers(boardId)`	`GET /boards/{id}/members`	Récupère les membres du board

5.6 Gestion du Token

Toutes les requêtes utilisent le pattern suivant :

```
const getToken = async () => {
  return await AsyncStorage.getItem("trello_token");};

  // Dans chaque méthode
  const API_TOKEN = await getToken();

  const response = await fetch(
  `${BASE_URL}/endpoint?key=${API_KEY}&token=${API_TOKEN}`,  { /* option
s */ }
);
```

---

6. Gestion de l'état

6.1 État local avec React Hooks

L'application utilise principalement les hooks natifs de React :

```
// État simple
const [data, setData] = useState([]);
// État de chargement
const [loading, setLoading] = useState(true);
// État de refresh
const [refreshing, setRefreshing] = useState(false);
// État des modals
```

```
const [modalVisible, setModalVisible] = useState(false);
const [selectedItem, setSelectedItem] = useState(null);
```

6.2 Pattern de mise à jour

```
// Création

const handleCreate = async (formData) => {
  try {
    await service.create(formData);
    loadData();

    } catch (error) {
    alert('Erreur lors de la création');  }
};

// Mise à jour
const handleUpdate = async (formData) => {
  try {
    await service.update(selectedItem.id, formData);
    loadData();

    } catch (error) {
    alert('Erreur lors de la modification');  }
};

// Suppression

const handleDelete = async (id) => {
  try {
    await service.delete(id);
    loadData();

    } catch (error) {
    alert('Erreur lors de la suppression');  }
};
```

7. Navigation

7.1 Structure de navigation

```
NavigationContainer
 └── TabNavigator (Bottom Tabs)
    ├── Home (Screen)
    ├── WorkspacesTab (Stack)
    │    └── WorkspaceStack
```

```
    │       ├── Workspaces (Screen)
    │       ├── Boards (Screen)
    │       └── Lists (Screen)
    └── Login (Screen)
```

7.2 Configuration du Stack Navigator

Fichier : `navigation/AppNavigator.js`

7.3 Navigation entre écrans

```
// Navigation

simplenavigation.navigate('Boards', { workspace });

// Retour arrière

navigation.goBack();

// Accès aux paramètres

const { workspace } = route.params;
```

7.4 Modification dynamique du titre

```
useEffect(() => {
 navigation.setOptions({
  title: board.name
 });}, [board]);
```

---

8. Composants

8.1 Hiérarchie des composants

```
projet/
├── 📂 components/            # Composants réutilisables
│   ├──
│   ├──
│   ├──
│   ├──
│   └── 📂 modals/           # Modals de l'application
│       ├──
│       ├──
│       ├──
│       └──
├── 📂 navigation/           # Configuration de la navigation
```

```
│   └── AppNavigator.js       # Navigation principale (Tabs + Stack)
│
├── 📂 screens/               # Écrans de l'application
│   ├──
│   ├──
│   ├──
│   ├──
│   └──
│
├── 📂 services/              # Services API Trello
│   ├── boardService.js       # CRUD des boards
│   ├── cardService.js        # CRUD des cartes
│   ├── listService.js        # CRUD des listes
│   └── workspaceService.js   # CRUD des workspaces
│
├── App.js                    # Point d'entrée de l'application
├── package.json              # Dépendances du projet
└── README.md                 # Fichier Readme
```

8.2 Composants Card

WorkspaceCard

```
<WorkspaceCard
  workspace={workspace}
  onPress={handlePress}
  onEdit={handleEdit}
  onDelete={handleDelete}
/>
```

Props :
- `workspace` : Objet workspace Trello
- `onPress` : Fonction de navigation
- `onEdit` : Fonction d’édition
- `onDelete` : Fonction de suppression

BoardCard

```
<BoardCard
  board={board}
  onPress={handlePress}
  onEdit={handleEdit}
  onDelete={handleDelete}
/>
```

8.3 Composants Modal

WorkspaceModal

```
<WorkspaceModal
  visible={modalVisible}
  workspace={selectedWorkspace} // null pour création
  onClose={closeModal}
  onSubmit={handleSubmit}
/>
```

BoardModal

```
<BoardModal
  visible={modalVisible}
  board={selectedBoard}
  onClose={closeModal}
  onSubmit={handleSubmit}
/>
```

Fonctionnalités :
- Sélection de templates
- Validation des champs
- Mode création/édition

ListModal

```
<ListModal
  visible={modalVisible}
  list={selectedList}
  onClose={closeModal}
  onSubmit={handleSubmit}
/>
```

CardModal

```
<CardModal
  visible={modalVisible}
  card={selectedCard}
  boardId={boardId}
  members={members}
  onClose={closeModal}
  onSubmit={handleSubmit}
  onAssignMember={handleAssignMember}
  onRemoveMember={handleRemoveMember}
/>
```

Fonctionnalités avancées :
- Assignation de membres

- Description multilignes

9. Authentification et sécurité

9.1 Flux d’authentification

1. Utilisateur se connecte avec oAuth de trello et on récupère en arrière plan le token
   ↓
2. Token stocké dans AsyncStorage
   ↓
3. Validation du token via API
   ↓
4. Navigation vers Home si succès

9.2 LoginScreen

Fichier : `screens/LoginScreen.js`

9.3 Récupération du token

```
const getToken = async () => {
 return await AsyncStorage.getItem("trello_token");};
```

9.4 Sécurité

Bonnes pratiques implémentées

- Token stocké localement (AsyncStorage)
- Pas de token en clair dans le code
- Validation côté serveur (API Trello)
- HTTPS pour toutes les requêtes

Points d’amélioration futurs

- Cryptage du token dans AsyncStorage
- Gestion du refresh token
- Expiration automatique de session
- Déconnexion sécurisée

12. Tests

12.1 Types de tests recommandés

Tests unitaires

Tests d’intégration

12.2 Configuration des tests

```
{  "scripts": {
       "test": "jest",
        "test:watch": "jest --watch",
        "test:coverage": "jest --coverage"  }}
```

---

13. Maintenance et évolution

13.1 Checklist de maintenance

- [ ] Mise à jour des dépendances
- [ ] Tests de régression
- [ ] Optimisation des performances
- [ ] Correction des bugs signalés

13.2 Roadmap technique

Court terme (1-3 mois)

- [ ] Implémentation du cache local
- [ ] Mode hors ligne basique
- [ ] UI updates
- [ ] Tests unitaires complets

Moyen terme (3-6 mois)

- [ ] Synchronisation en temps réel
- [ ] Notifications push
- [ ] Mode sombre
- [ ] Recherche avancée

Long terme (6-12 mois)

- [ ] Drag & drop natif
- [ ] Widgets
- [ ] Intégration calendrier

---

14. Annexes

14.1 Conventions de code

Nommage

```
// Composants : PascalCaseconst
WorkspaceCard = () => {};

// Services : camelCase avec suffixe 'Service'

const workspaceService = {};

// Fonctions : camelCase

const handleCreate = () => {};

// Constantes : UPPER_SNAKE_CASE

const API_KEY = "...";
```

14.2 Glossaire

Terme	Définition
Workspace	Organisation/Équipe Trello
Board	Tableau de projet Trello
List	Colonne d'un board
Card	Tâche/élément d'une liste
Member	Utilisateur assigné
Token	Jeton d'authentification OAuth

Documentation officielle

- **Trello API** : https://developer.atlassian.com/cloud/trello/rest/
- **React Native** : https://reactnative.dev/docs/getting-started
- **Expo** : https://docs.expo.dev/
- **React Navigation** : https://reactnavigation.org/docs/getting-started
- **AsyncStorage** : https://react-native-async-storage.github.io/async-storage/

Outils de développement

- **Postman** : Test des endpoints API Trello
- **React Native Debugger** : Débogage avancé
- **Expo DevTools** : Interface de développement Expo
- **Flipper** : Débogage React Native natif

14.3 Diagrammes techniques

Diagramme de séquence - Création d'une carte

Diagramme de séquance - creating-card.pdf

14.4 Sécurité - Checklist

✅ Implémenté

- [x] ~~HTTPS pour toutes les requêtes~~
- [x] ~~Token stocké en local~~
- [x] ~~Validation des entrées utilisateur~~
- [x] ~~Gestion des erreurs d'authentification~~
- [x] ~~Pas de données sensibles en logs~~

⚠️ À améliorer

- [ ] Cryptage du token AsyncStorage
- [ ] Implémentation du refresh token

14.5 Code Review Checklist

Avant de soumettre une PR

- [ ] Code testé manuellement
- [ ] Commentaires clairs ajoutés
- [ ] Documentation mise à jour
- [ ] Conventions de nommage respectées
- [ ] Gestion des erreurs implémentée
- [ ] Performance considérée
- [ ] Accessibilité vérifiée
- [ ] Responsive sur différentes tailles d'écran

---

15. Conclusion

15.1 Points forts de l'architecture

✅ **Modularité** : Services séparés et composants réutilisables

✅ **Maintenabilité** : Code structuré et bien documenté

✅ **Scalabilité** : Architecture permettant l'ajout de fonctionnalités

✅ **Séparation des responsabilités** : Présentation, logique, données

15.2 Axes d'amélioration

⚠️ **Tests** : Augmenter la couverture de tests

⚠️ **Performance** : Implémenter le cache et l'optimistic UI

⚠️ **Sécurité** : Renforcer le stockage et l'authentification

⚠️ **Offline** : Ajouter le support hors ligne

15.3 Contact technique

Pour toute question technique concernant cette documentation :

- 📧 Email : josue.houndenou@epitech.eu

Version de la documentation : 1.0.0

Dernière mise à jour : 21 Novembre 2025

Auteurs : Équipe de développement mobile
