Royaume du Maroc
HONORIS UNITED UNIVERSITIES

ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

École Marocaine Des Sciences de L'Ingenieur
Département Informatique

Rapport du projet de fin de module

Développement Mobile avec React Native

Option : 4IIR G9

MyStudyCompanion

Application d'assistance à la productivité étudiante

Réalisé par :
M. HANFAOUI Karim
Mme. KAFIF Imane

Encadré par :
Pr. Rachik ZINEB

Année Universitaire : 2025/2026

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué au succès de ce projet.

Nous remercions tout particulièrement notre encadrant, **Pr. Rachik Zineb** , pour ses directives claires concernant l'architecture mobile, sa disponibilité et ses conseils techniques précieux tout au long de ce module. Son expertise nous a permis de surmonter les obstacles techniques liés à l'environnement React Native.

Nos remerciements vont également au corps professoral de l'EMSI pour la qualité de la formation dispensée, ainsi qu'aux membres du jury qui ont accepté d'évaluer ce travail.

L'équipe projet
7 décembre 2025

Résumé

Ce projet, intitulé **MyStudyCompanion**, consiste en la conception et la réalisation d'une application mobile cross-platform dédiée à l'amélioration de la productivité des étudiants. L'objectif est de fournir une interface centralisée regroupant des outils essentiels tels qu'un minuteur Pomodoro, une gestion de tâches (To-Do List) et un emploi du temps synchronisé.

La solution a été développée en utilisant le framework **React Native** avec **Expo**. L'architecture technique repose sur une approche "Offline-First" : les données sont persistées localement via **AsyncStorage** pour garantir l'accès sans connexion, tout en étant synchronisées avec une API distante (Google Sheets) pour les mises à jour dynamiques. Le code suit une structure modulaire stricte.

Mots-clés : React Native, Expo, Mobile, Android, Offline-First, Productivité.

Abstract

This project, titled **MyStudyCompanion**, focuses on the design and development of a cross-platform mobile application aimed at enhancing student productivity. The goal is to provide a centralized interface combining essential tools such as a Pomodoro timer, a To-Do list manager, and a synchronized schedule viewer.

The solution was built using the **React Native** framework with **Expo**. The technical architecture follows an "Offline-First" approach : data is persisted locally via **AsyncStorage** to ensure accessibility without an internet connection.

Keywords : React Native, Expo, Mobile, Android, Offline-First, Productivity.

Table des matières

Table des figures

Chapitre 1

Introduction Générale

1.1 Contexte du Projet

L'essor des technologies mobiles a transformé notre manière d'interagir avec l'information. Dans le milieu académique, et spécifiquement au sein de l'EMSI, les étudiants sont confrontés à une charge de travail dense nécessitant une organisation rigoureuse. La gestion du temps est un défi majeur. La multiplication des supports (agenda papier, applications de notes éparses, groupes WhatsApp pour les annonces) crée une friction qui nuit à la productivité et augmente la charge mentale.

1.2 Problématique

Les solutions existantes sur le marché sont souvent soit trop complexes (outils de gestion de projet d'entreprise), soit trop simplistes (simples applications de notes). Il manque une solution *adaptée à l'étudiant*, qui centralise :

- La gestion du temps de concentration (Pomodoro).
- Le suivi des tâches académiques.
- L'accès rapide à l'emploi du temps, même sans connexion internet dans les amphithéâtres.

1.3 Solution Proposée

MyStudyCompanion est une application mobile développée pour répondre à ce besoin spécifique. Elle propose une architecture robuste permettant une utilisation fluide en mode déconnecté, une synchronisation légère avec le Cloud pour les données partagées (emploi du temps), et une interface utilisateur intuitive respectant les standards du Material Design.

Chapitre 2

Analyse et Conception

2.1 Spécification des Besoins

Avant d'entamer le développement, nous avons identifié les besoins fonctionnels via des User Stories.

Acteur	**Description du besoin**	**Priorité**
Étudiant	En tant qu'étudiant, je veux visualiser mon emploi du temps pour savoir où est mon prochain cours.	Haute
Étudiant	En tant qu'étudiant, je veux lancer un minuteur de 25 minutes pour me concentrer.	Haute
Étudiant	En tant qu'étudiant, je veux ajouter des tâches personnelles pour ne rien oublier.	Moyenne
Étudiant	En tant qu'étudiant, je veux consulter mes données même sans 4G/Wifi.	Haute

2.2 Modélisation UML

Pour assurer une architecture cohérente, nous avons réalisé les diagrammes suivants avant le codage.

2.2.1 Diagramme de Cas d'Utilisation

Ce diagramme illustre les fonctionnalités offertes par MyStudyCompanion aux étudiants, ainsi que l'interaction avec le système externe Google Sheets.

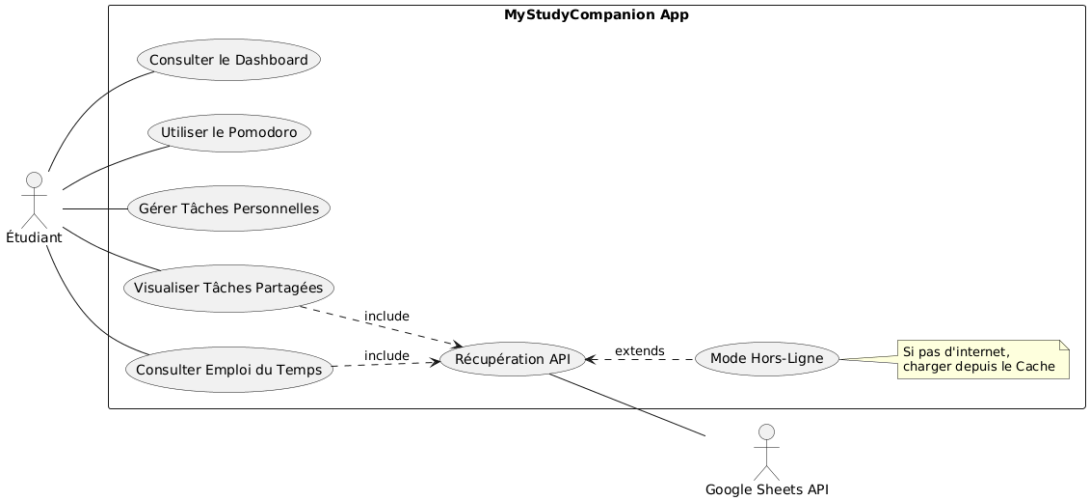

FIGURE 2.1 – Diagramme de Cas d'Utilisation

2.2.2 Diagramme de Séquence (Synchronisation)

Le diagramme suivant détaille le mécanisme de synchronisation "Offline-First". Il montre comment l'application privilégie les données fraîches de l'API mais bascule automatiquement sur le cache local (`AsyncStorage`) en cas d'erreur réseau.

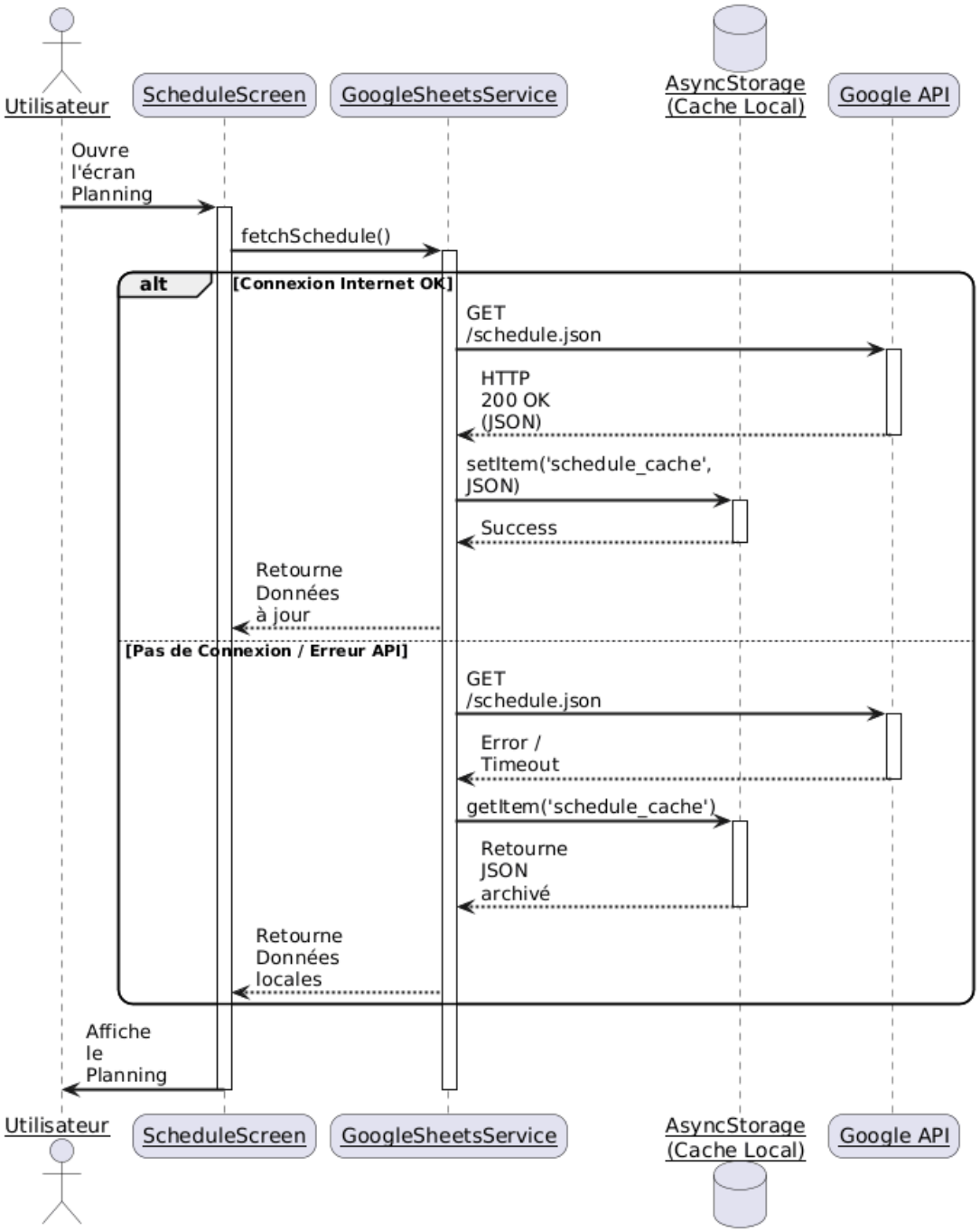

FIGURE 2.2 – Diagramme de Séquence : Récupération des données

2.2.3 Diagramme de Classes

Ce diagramme présente l'organisation modulaire du code source, séparant clairement les Vues (Screens), les Composants UI et la logique métier (Services).

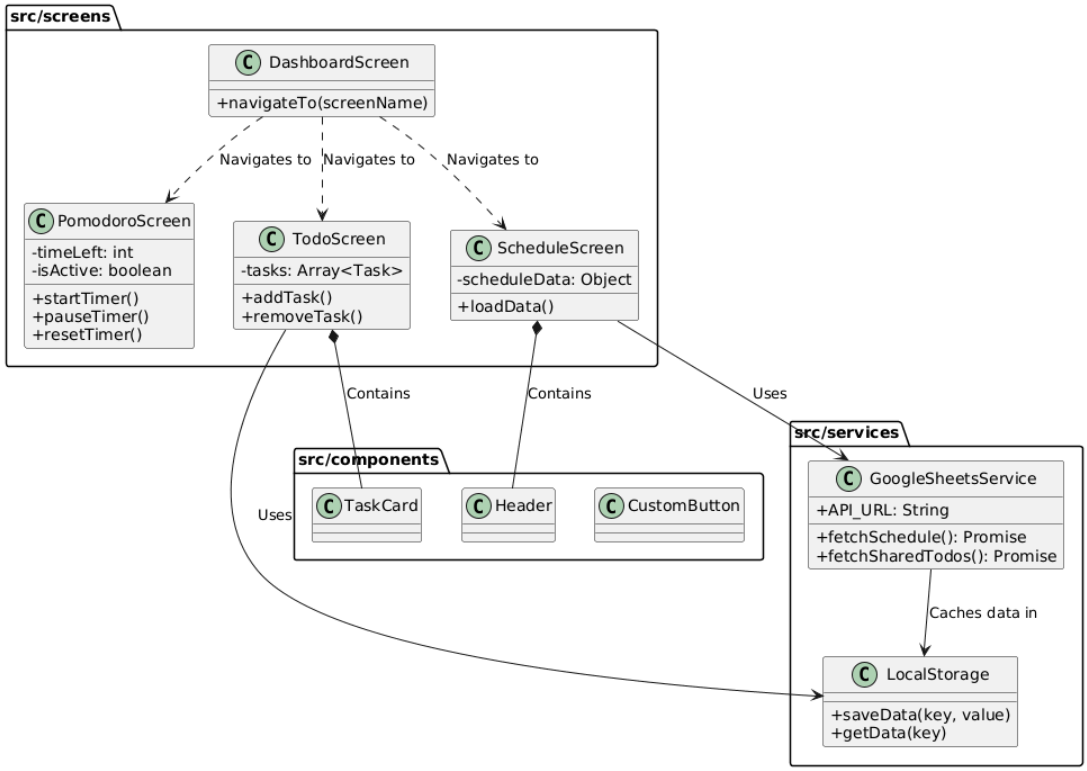

FIGURE 2.3 – Diagramme de Classes Simplifié

Chapitre 3

État de l'Art et Choix Technologiques

3.1 Framework : React Native

React Native est un framework open-source créé par Meta (Facebook). Contrairement aux applications "hybrides" classiques (Cordova, Ionic) qui utilisent une WebView pour rendre du HTML, React Native invoque les composants natifs de l'OS (Android/iOS).

3.1.1 Pourquoi React Native ?

— **Performance :** Grâce au "Bridge", le code JavaScript contrôle des vues natives, offrant une fluidité (60fps) bien supérieure aux WebViews.
— **Cross-Platform :** Une seule base de code (JavaScript/JSX) permet de générer des binaires pour Android (.apk) et iOS (.ipa).
— **Écosystème :** La communauté est immense, offrant des bibliothèques pour presque tous les besoins (Navigation, Stockage, Caméra).

3.2 L'Écosystème Expo

Nous avons choisi d'utiliser **Expo** pour ce projet. Expo est une surcouche à React Native qui simplifie considérablement le développement.

— **Expo Go :** Permet de tester l'application en temps réel sur notre appareil physique via un QR Code, sans avoir à compiler l'application à chaque modification.
— **EAS Build :** Service cloud qui permet de compiler l'application finale sans avoir besoin d'un environnement Android Studio ou Xcode complexe en local.

3.3 Gestion des Données

3.3.1 Persistance Locale : AsyncStorage

Pour répondre à l'exigence "Offline-First", nous utilisons `AsyncStorage`. C'est un système de stockage clé-valeur asynchrone, persistant et global pour l'application. C'est l'équivalent du `localStorage` sur le web, mais optimisé pour le mobile.

3.3.2 Backend : Google Sheets API

Plutôt que de développer un backend lourd pour un emploi du temps statique, nous avons utilisé Google Sheets comme CMS (Content Management System).

— L'administration met à jour le fichier Excel/Sheet.
— L'application mobile "fetch" ce fichier au format JSON.
— Cette approche réduit les coûts d'hébergement et simplifie la maintenance.

Chapitre 4

Architecture et Implémentation

4.1 Structure du Projet

Une bonne organisation du code est cruciale pour la maintenabilité. Nous avons adopté l'architecture suivante :

Arborescence des Fichiers

```
/projetDevMobile
|-- /assets             # Images, polices et icônes
|-- /src
|   |-- /components     # Boutons, Cartes, Headers (Réutilisables)
|   |-- /screens        # Les pages complètes (Vues)
|   |   |-- Dashboard.js
|   |   |-- Pomodoro.js
|   |   |-- TodoScreen.js
|   |   |-- SharedTodoScreen.js
|   |   |-- ScheduleScreen.js
|   |-- /services       # Logique métier et API
|   |   |-- googleSheetsService.js
|   |-- /navigation     # Configuration du Stack Navigator
|-- App.js              # Point d'entrée de l'application
|-- package.json        # Dépendances (npm)
|-- app.json            # Configuration Expo (Nom, Version, Slug)
```

4.2 Implémentation de la Navigation

Nous utilisons `@react-navigation/native-stack`. Voici comment nous avons configuré les routes principales dans `App.js`.

```
1 import { NavigationContainer } from '@react-navigation/native';
2 import { createNativeStackNavigator } from '@react-navigation/native-
     stack';
3 import Dashboard from './src/screens/Dashboard';
4 import Pomodoro from './src/screens/Pomodoro';
5
6 const Stack = createNativeStackNavigator();
7
```

```
export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Dashboard">
        <Stack.Screen name="Dashboard" component={Dashboard} />
        <Stack.Screen name="Pomodoro" component={Pomodoro} />
        {/* Autres  crans  ici */}
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Listing 4.1 – Configuration de la Navigation (App.js)

4.3 Le Service de Données (API)

Le fichier `googleSheetsService.js` est responsable de la communication avec l'extérieur. Il transforme les données brutes CSV/JSON en objets JavaScript exploitables par l'application.

```
export const fetchSchedule = async () => {
  try {
    const response = await fetch('URL_DE_VOTRE_GOOGLE_SHEET');
    const json = await response.json();
    return json;
  } catch (error) {
    console.error("Erreur API:", error);
    return null; // Retourne null pour d clencher le mode hors ligne
  }
};
```

Listing 4.2 – Service de récupération des données

4.4 Gestion du Minuteur Pomodoro

L'écran Pomodoro utilise le Hook `useState` et `useEffect` pour gérer le décompte des secondes. La difficulté principale était de s'assurer que le minuteur s'arrête correctement si l'utilisateur quitte l'écran.

```
useEffect(() => {
  let interval = null;
  if (isActive && seconds > 0) {
    interval = setInterval(() => {
      setSeconds(seconds - 1);
    }, 1000);
  } else if (seconds === 0) {
    setIsActive(false);
    alert("Session termin e !");
  }
  return () => clearInterval(interval); // Nettoyage
}, [isActive, seconds]);
```

Listing 4.3 – Logique du Timer (Extrait)

Chapitre 5

Présentation des Interfaces

Cette section illustre les résultats visuels du développement sur un appareil Android physique.

5.1 Tableau de Bord (Dashboard)

Le Dashboard est le point d'entrée central. Il permet une navigation intuitive vers les différents modules.

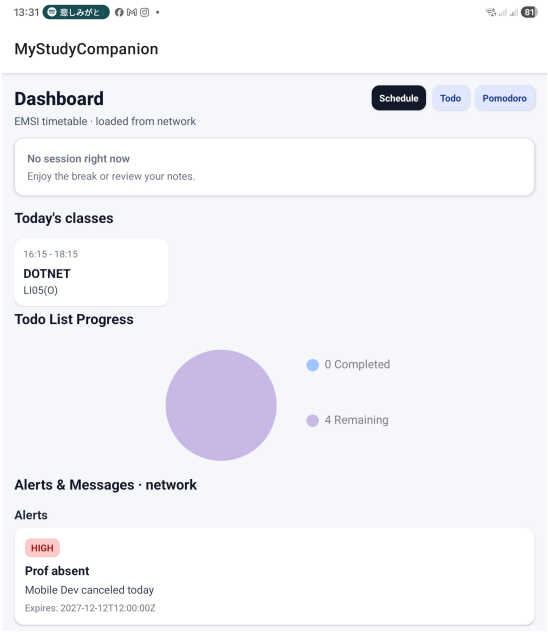

FIGURE 5.1 – Dashboard - Vue Principale

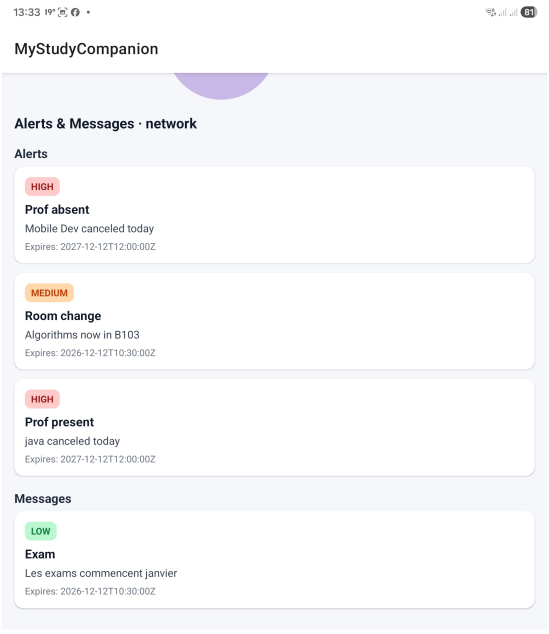

FIGURE 5.2 – Dashboard - Vue Menu

5.2 Gestion des Tâches (Task Management)

L'application propose une gestion avancée des tâches, séparant les tâches personnelles (stockées localement) des tâches partagées (synchronisées).

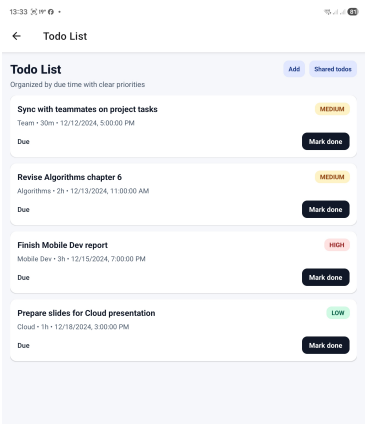

FIGURE 5.3 – Liste Personnelle

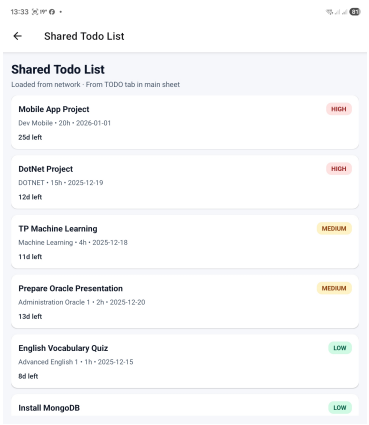

FIGURE 5.4 – Liste Partagée

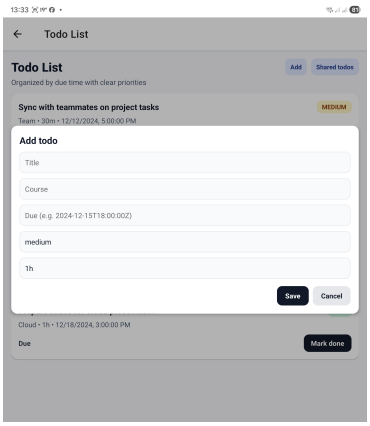

FIGURE 5.5 – CRUD (Édition)

Figure 5.3-5.5 : Workflow complet de gestion des tâches

5.3 Le Minuteur Pomodoro

L'interface de concentration (Focus) est épurée pour éviter les distractions.

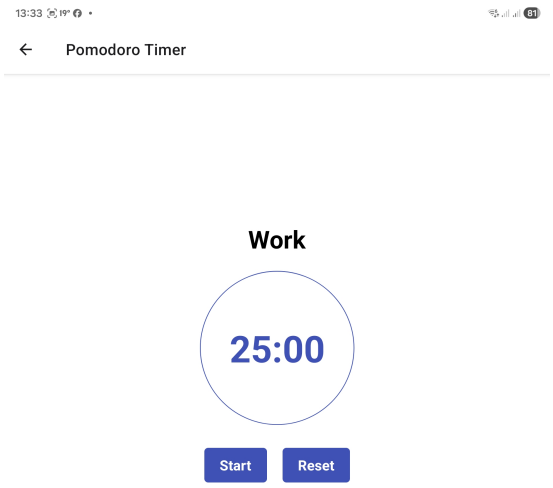

FIGURE 5.6 – Session de Concentration active

Chapitre 6

Gestion de Projet et Difficultés

6.1 Méthodologie Agile

Nous avons travaillé en mode itératif.

— **Sprint 1 :** Mise en place de l'environnement Expo et création du "Hello World" (Dashboard).
— **Sprint 2 :** Développement de la logique Pomodoro et Todo List locale.
— **Sprint 3 :** Intégration de l'API et finalisation du Design.

6.2 Versionning avec Git

L'utilisation de Git a été centrale. Chaque fonctionnalité a été développée sur une branche dédiée avant d'être fusionnée (*merge*) sur la branche principale `main`.

Dépôt GitHub

Le code source complet est disponible ici : `https://github.com/carteeeltheboss/projetDevMobile.git`

Accès au Code

6.3 Difficultés Techniques Rencontrées

6.3.1 Problème de Cache

Problème : Lors des premiers tests, les données de la Todo List disparaissaient au redémarrage de l'application. **Solution :** Nous avons mal implémenté `AsyncStorage.setItem` qui attend une chaîne de caractères (String). Nous avons dû utiliser `JSON.stringify()` pour sauvegarder les objets et `JSON.parse()` pour les lire.

6.3.2 Compatibilité Android

Certains composants d'interface (comme les ombres `shadow`) s'affichaient bien sur iOS (simulateur) mais pas sur Android. Nous avons dû utiliser la propriété `elevation` spécifique à Android pour corriger ce défaut visuel.

Chapitre 7

Conclusion et Perspectives

Ce projet de fin de module nous a permis de consolider nos acquis en développement mobile. Nous avons réussi à livrer une application **MyStudyCompanion** fonctionnelle, répondant aux exigences de l'EMSI : 6 écrans, connexion API, stockage local et documentation complète.

Les perspectives d'évolution sont nombreuses :

— **Mode Sombre (Dark Mode) :** Utilisation de l'API `Appearance` de React Native.

— **Notifications Push :** Intégration de Firebase pour prévenir l'étudiant 10 minutes avant un cours.

— **Version iOS :** Test et déploiement sur l'Apple Store (actuellement testé via Expo Go).

Bibliographie

[1] Documentation officielle React Native, https://reactnative.dev

[2] Documentation Expo, https://docs.expo.dev

[3] React Hooks Documentation, https://reactjs.org/docs/hooks-intro.html

[4] AsyncStorage Docs, https://react-native-async-storage.github.io/async-storage/