Documentation du projet mobile Thinkty

Introduction

Thinkty est une application mobile qui permet aux utilisateurs de noter des idées, gérer des tâches et stocker des mots de passe de manière sécurisée. Son objectif est d'offrir une organisation efficace et une protection avancée des données grâce à un chiffrement.

Fonctionnalités principales

- **Checklist** : Ajouter, modifier et supprimer des tâches avec mise à jour en temps réel.
- **Prise de notes** : Écrire et organiser ses idées, avec la possibilité de modifier les notes.
- **Coffre-fort de mots de passe** : Stocker et protéger des identifiants de connexion avec un mot de passe général de l'application.
- **Page Profil** : Modifier son mot de passe général, utilisé pour le coffre-fort.
- **Sécurité avancée** : Toutes les données sensibles sont protégées via un mot de passe général et un chiffrement avant stockage dans la Base de donnée.

Architecture du projet

Technologies utilisées :

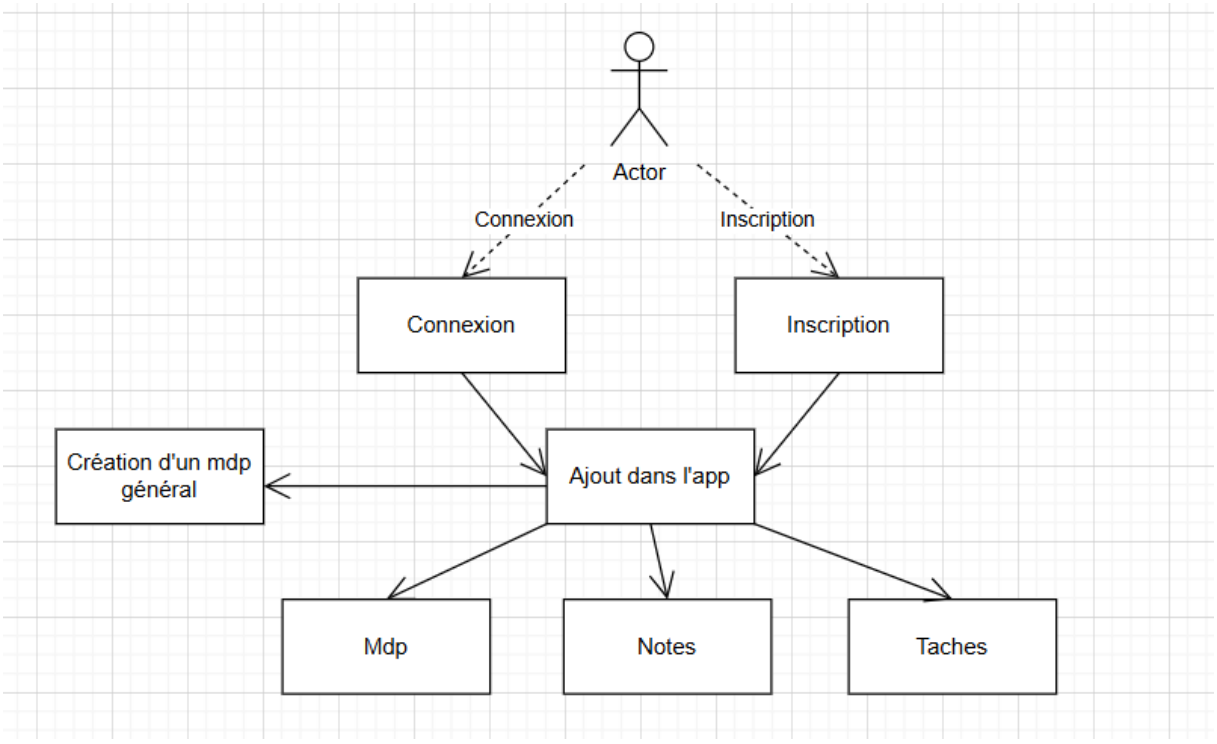
- **React Native** : Utilisé pour le développement mobile cross-platform (iOS et Android).
- **Expo** : Utilisé pour faciliter le développement et la gestion du projet React Native.

- **Supabase** : Base de données en temps réel utilisée pour stocker les données des utilisateurs (notes, mots de passe, tâches). Supabase offre une API PostgreSQL pour la gestion des données.
- **GitHub** : Utilisé pour la gestion du code source et le versionnage avec Git. Hébergement du projet sur GitHub pour une collaboration en équipe et un suivi efficace des changements.

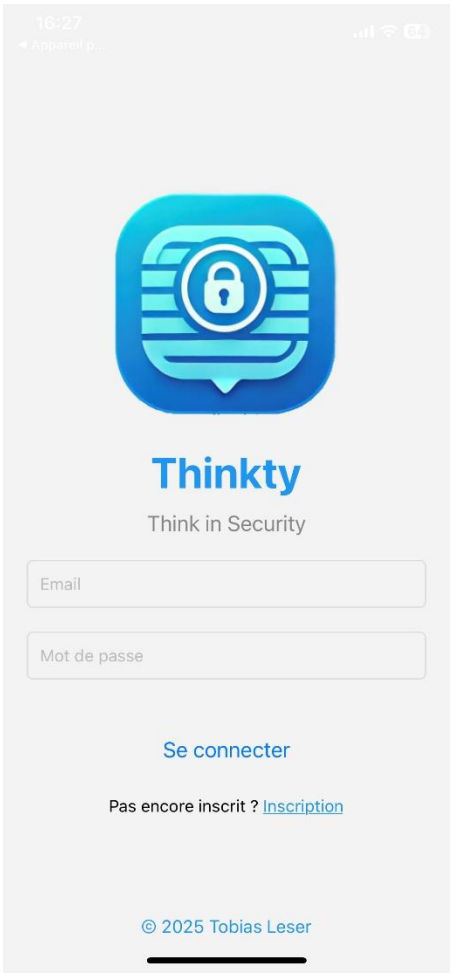
Modules principaux

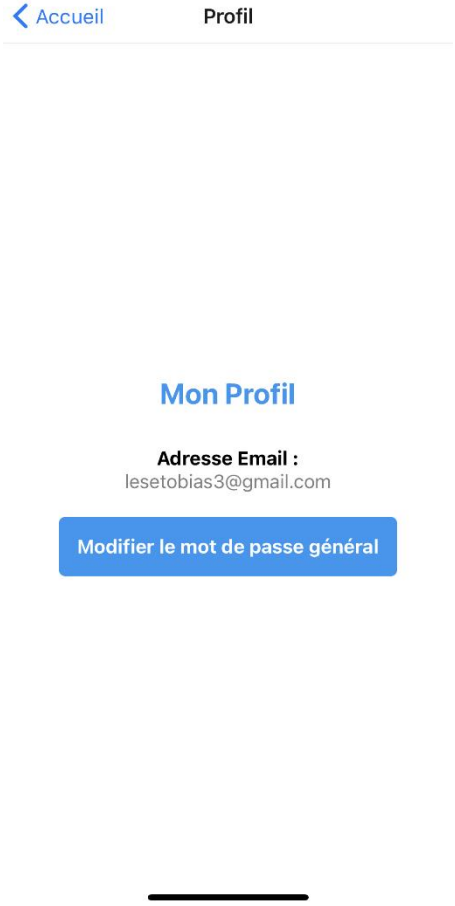
- **React Navigation** : Gère la navigation entre les différentes pages (profil, checklist, notes, coffre-fort).
- **Supabase.js** : Intégration avec Supabase pour la gestion des données utilisateurs.
- **Bcrypt.js** : Utilisé pour le hachage des mots de passe avant leur stockage.
- **ISAAC (Encryption)** : Utilisé pour le chiffrement des mots de passe stockés dans la base de données.

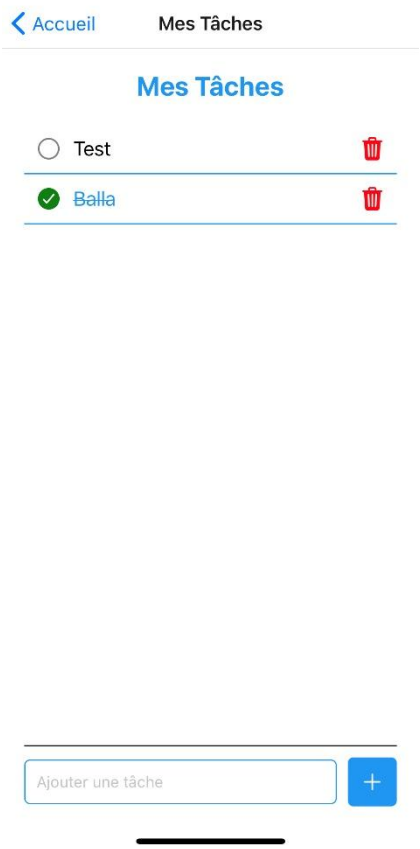
Schéma de cas d'utilisation :

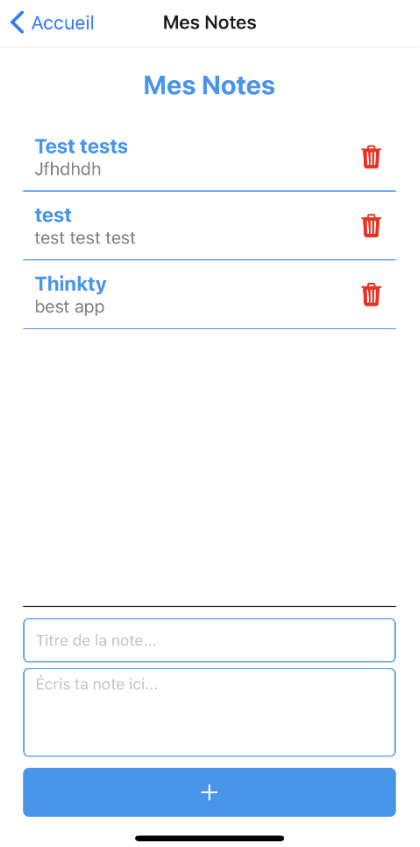


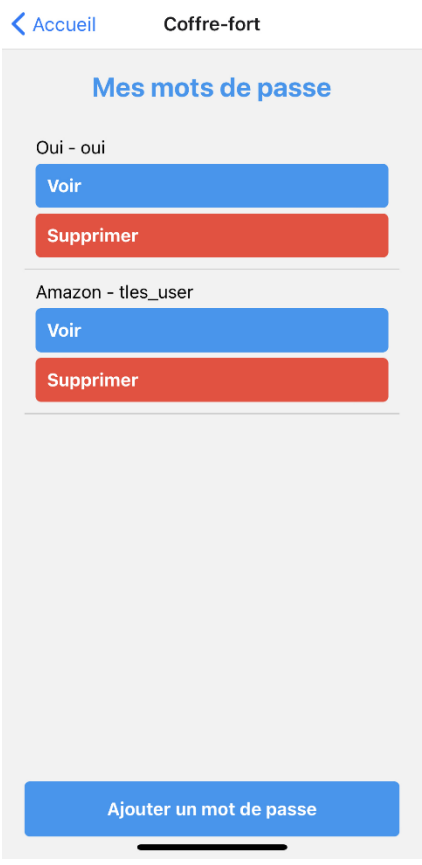
Quelques captures de Thinkty :



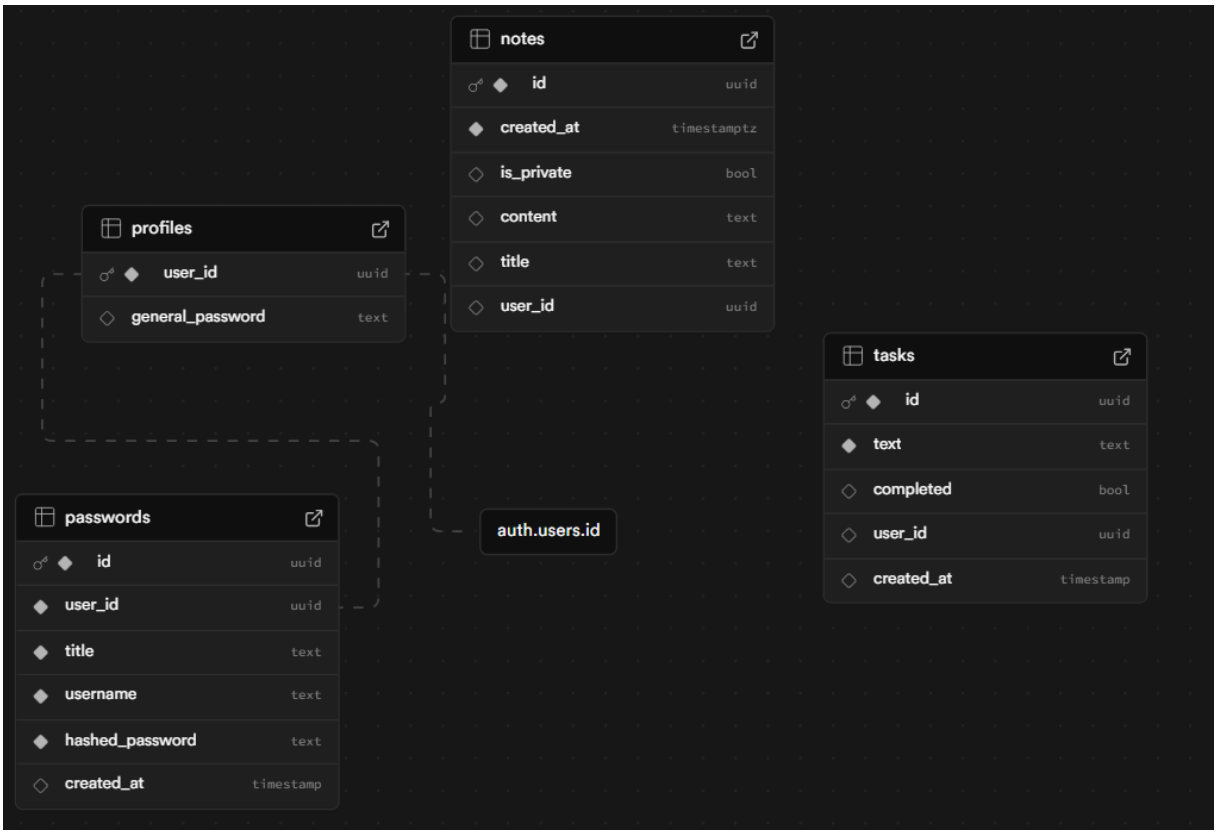








Structure de la Base de donnée Supabase :



Charte graphique :

1. Couleurs principales

Couleur primaire :

Hex : #2A2E36 (Fondateur du fond de l'application).

Couleur secondaire :

Hex : #4E9F3D (Utilisé pour les boutons, liens et actions principales).

Couleur d'arrière-plan :

Hex : #F4F4F9 (Fond général de l'application).

Couleur texte principal :

Hex : #333333 (Texte principal)

2. Typographie

Police principale : Roboto

Utilisation : Titres, textes principaux, boutons.

Police secondaire : Open Sans

Utilisation : Texte secondaire, descriptions.

3. Boutons et champs

Style des boutons :

Coins arrondis, couleur secondaire (#4E9F3D) pour les boutons primaires.

Texte centré, en blanc.

Survol : Changement de couleur vers un ton plus clair de vert (#6DC55A).

Champs de saisie :

Bordure fine (#E0E0E0), coins arrondis (5px).

4. Icônes

Style : Simple, linéaire, épurée. Utiliser la couleur primaire pour les icônes principales.

5. Illustrations

Style : Illustrations vectorielles simples, avec des accents de vert (#4E9F3D).

6. Accessibilité

Assurez-vous que le contraste entre le texte et l'arrière-plan est suffisant.

Interface intuitive avec une police lisible.

Backend

Thinkty utilise Supabase comme backend, qui fournit une base de données PostgreSQL, une authentification utilisateur, et des API REST/Realtime pour interagir avec les données en temps réel. L'objectif est d'avoir un backend léger et sécurisé, tout en évitant de gérer un serveur dédié.

Exemple : Authentification utilisateur :

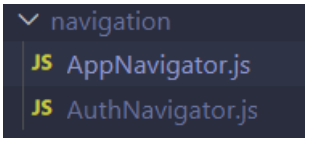
Thinkty utilise React Navigation pour organiser la navigation entre les écrans en fonction de l'état de connexion de l'utilisateur. L'application est divisée en deux navigateurs principaux :

- AuthNavigator : Contient les écrans d'authentification (connexion, inscription). Il est affiché si l'utilisateur n'est pas connecté.
- AppNavigator : Contient les écrans principaux (Checklist, Notes, Coffre-fort). Il est affiché si l'utilisateur est authentifié.

Le composant principal de navigation vérifie en temps réel avec Supabase si l'utilisateur est connecté ou non. Si un utilisateur se connecte ou se déconnecte, l'application bascule automatiquement entre AuthNavigator et AppNavigator.

Cette architecture garantit que seuls les utilisateurs authentifiés peuvent accéder aux données sensibles, comme les notes privées et le coffre-fort de mots de passe.

```jsx
import React from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import MainScreen from '../screens/MainScreen';
import AboutScreen from '../screens/AboutScreen';
import TasksScreen from '../screens/TasksScreen';
import NotesScreen from '../screens/NotesScreen';
import NoteDetailScreen from '../screens/NoteDetailScreen';
import ProfileScreen from '../screens/ProfileScreen';
import PasswordScreen from '../screens/PasswordScreen';

const Stack = createStackNavigator();

const AppNavigator = () => {
    return (
        <Stack.Navigator>
            <Stack.Screen name="Accueil" component={MainScreen} />
            <Stack.Screen name="À propos" component={AboutScreen} />
            <Stack.Screen name="Mes Tâches" component={TasksScreen} />
            <Stack.Screen name="Mes Notes" component={NotesScreen} />
            <Stack.Screen name="Détail de la note" component={NoteDetailScreen} />
            <Stack.Screen name="Profil" component={ProfileScreen} />
            <Stack.Screen name="Coffre-fort" component={PasswordScreen} />
        </Stack.Navigator>
    );
};

export default AppNavigator;
```




```jsx
import React from 'react';
import { createStackNavigator } from '@react-navigation/stack';
import LoginScreen from '../screens/LoginScreen';
import RegisterScreen from '../screens/RegisterScreen';

const Stack = createStackNavigator();

const AuthNavigator = () => {
    return (
        <Stack.Navigator screenOptions={{ headerShown: false }}>
            <Stack.Screen name="Login" component={LoginScreen} />
            <Stack.Screen name="Register" component={RegisterScreen} />
        </Stack.Navigator>
    );
};

export default AuthNavigator;
```

Sécurité

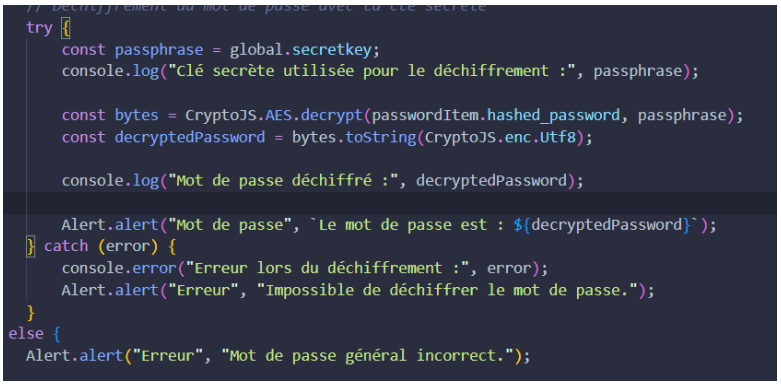
Thinkty intègre plusieurs technologies pour garantir la protection des données des utilisateurs :

Bcrypt.js : Hachage des mots de passe utilisateur avant stockage pour empêcher toute récupération en cas de fuite.

AES-256 : Chiffrement des mots de passe enregistrés dans le coffre-fort, rendant leur lecture impossible sans la clé de déchiffrement.

ISAAC : Génération sécurisée de mots de passe aléatoires, évitant les failles des générateurs classiques.

Chiffrage du mot de passe dans la BDD.



```
try {
    const passphrase = global.secretkey;
    console.log("Clé secrète utilisée pour le déchiffrement :", passphrase);

    const bytes = CryptoJS.AES.decrypt(passwordItem.hashed_password, passphrase);
    const decryptedPassword = bytes.toString(CryptoJS.enc.Utf8);

    console.log("Mot de passe déchiffré :", decryptedPassword);

    Alert.alert("Mot de passe", `Le mot de passe est : ${decryptedPassword}`);
} catch (error) {
    console.error("Erreur lors du déchiffrement :", error);
    Alert.alert("Erreur", "Impossible de déchiffrer le mot de passe.");
}
else {
Alert.alert("Erreur", "Mot de passe général incorrect.");
```

Déchiffrement du mot de passe après comparaison

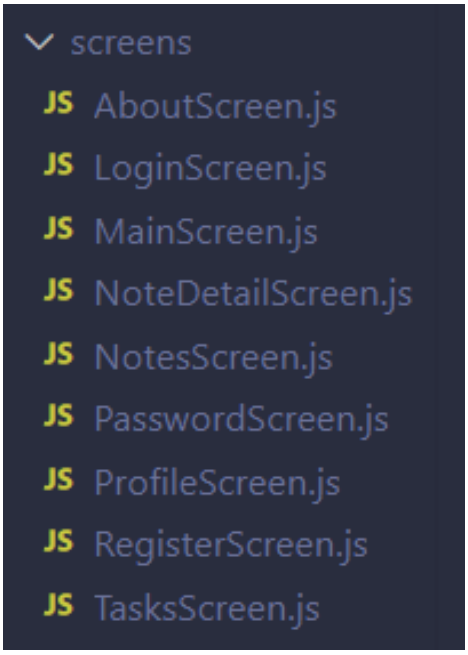
```
try {
    const passphrase = global.secretkey
    console.log('clé secrete', passphrase)
    const encrypted = CryptoJS.AES.encrypt(password, passphrase).toString();

    const bytes = CryptoJS.AES.decrypt(encrypted, passphrase);
    const decrypted = bytes.toString(CryptoJS.enc.Utf8);
    console.log([password, encrypted, decrypted])
    const { data, error } = await supabase
        .from('passwords')
        .insert([{
            user_id: userId,
            title,
            username,
            hashed_password: encrypted
        }]);
```

Metro.config.js

```
const { getDefaultConfig } = require("expo/metro-config");

module.exports = (async () => {
    const defaultConfig = await getDefaultConfig(__dirname);

    defaultConfig.resolver.extraNodeModules = {
        crypto: require.resolve("react-native-crypto-js"),
    };

    return defaultConfig;
})();
```

Frontend

Thinkty est développé avec React Native et utilise Expo pour simplifier le déploiement et le développement mobile. L'interface est conçue pour être intuitive et fluide, avec une navigation gérée par React Navigation. Les écrans principaux incluent la Checklist, le Coffre-fort et les Notes, accessibles uniquement après authentification. L'application applique également une gestion des états en temps réel avec Supabase, permettant une synchronisation instantanée des données entre les utilisateurs et la base de données.



Déploiement

Lors du premier lancement de l'application, Thinkty génère et stocke une clé secrète dans App.js qui sera utilisée pour chiffrer et déchiffrer les mots de passe enregistrés dans le coffre-fort. Cette clé est unique pour chaque installation et est stockée localement sur l'appareil.

Si elle n'existe pas elle est créée, si elle existe déjà elle est lue.

```
// Définition du chemin du fichier pour stocker la clé
const secretKeyPath = FileSystem.documentDirectory + 'secret.key';

// Fonction pour générer une clé aléatoire
const generateSecretKey = () => {
  let buf = new Uint8Array(32); // Taille raisonnable pour une clé de chiffrement
  buf.forEach((_, i) => (buf[i] = Math.floor(isaac.random() * 256)));
  return btoa(buf); // Conversion correcte
};

// Fonction pour récupérer ou créer la clé
const getSecretKey = async () => {
  try {
    const fileInfo = await FileSystem.getInfoAsync(secretKeyPath);
    let secretKey;

    if (fileInfo.exists) {
      // Lire la clé existante
      secretKey = await FileSystem.readAsStringAsync(secretKeyPath);
      console.log('Clé secrète lue :', secretKey);
    } else {
      // Générer une nouvelle clé et l'enregistrer
      secretKey = generateSecretKey();
      await FileSystem.writeAsStringAsync(secretKeyPath, secretKey);
      console.log('Nouvelle clé secrète générée et stockée :', secretKey);
    }

    global.secretkey = secretKey; // Assigner la clé globalement
    setIsKeyLoaded(true); // Mettre à jour l'état pour signaler que la clé est chargée
  } catch (error) {
    console.error('Erreur lors de l'accès à la clé secrète:', error);
  }
};
```

```
// Charger la clé secrète au démarrage
useEffect(() => {
  getSecretKey();
}, []);

// Attendre que la clé soit chargée avant d'afficher l'UI
if (!isKeyLoaded) {
  return null; // Évite d'afficher l'UI tant que la clé n'est pas chargée
}
```

Déploiement depuis Github

Ouvrir powershell

git clone https://github.com/votre-repo/Thinkty.git

cd Thinkty

npm install

SUPABASE_URL=supabase_url

SUPABASE_ANON_KEY=supabase_clé

npx expo start

Conclusion

Thinkty est une application mobile conçue pour simplifier la gestion des tâches, notes et mots de passe en toute sécurité. Grâce à React Native, Supabase et des algorithmes de chiffrement avancés, elle offre une expérience fluide et sécurisée. L'application est facilement déployable et extensible, avec une architecture bien structurée permettant d'ajouter de nouvelles fonctionnalités à l'avenir.