



UNIVERSITÉ DE
MONTPELLIER



Flutter

HAI912I - Développement mobile avancé, IoT et
embarqué

Projet Flutter

Réalisé par :

M^{lle} Toudherth MEKHACHE N° 22214610

M^{lle} Rima ZOURANE N°

Référent :

M^r Bachar RIMA

Promotion 2023/2024

Table des matières

1	Projet Flutter	1
1.1	Introduction	1
1.2	Logiciels et outlis utilisés	1
1.3	Fonctionnalités de l'application Flutter	3
1.4	Conception et architecture	3
1.4.1	Modélisation UML :	4
1.4.2	Conception de l'architecture	6
1.4.3	Les aspects d'ergonomie et d'expérience utilisateur	6
1.5	Implémentation des Interfaces avec l'API	11
1.5.1	Interfaces Utilisateur (UI)	11
1.5.2	Intégration avec l'API Backend	22
1.5.3	Défis Rencontrés et Solutions Adoptées	23
1.5.4	Améliorations Futures	23
1.6	Conclusion	24

Video demonstrative : [Application de Température et de luminosité avec Flutter, en utilisant un backend Arduino, ESP 32](#)

Le lien GitHub : [Ce lien est un lien vers le code source de projet sur Github](#)

Chapitre 1

Projet Flutter

1.1 Introduction

L'évolution rapide de l'Internet des Objets (IoT) met en évidence le besoin crucial d'interfaces utilisateurs efficaces pour interagir avec les appareils connectés. Le framework Flutter se distingue comme une solution incontournable pour créer des interfaces réactives, facilitant l'échange d'informations entre les utilisateurs et les dispositifs IoT.

Notre projet s'inscrit dans cette perspective en élaborant une interface utilisateur conçue avec Flutter qui permet d'interfacer avec une API RESTful pour gérer les données des capteurs en temps réel et contrôler à distance une LED. L'accent est mis sur une expérience utilisateur fluide et une architecture de code modulaire et maintenable.

Nous avons donc développé une application Flutter qui allie esthétique et fonctionnalité, en mettant l'accent sur l'affichage des données des capteurs et sur l'interaction avec une LED, tout en respectant des normes de développement élevées et en adoptant une approche modulaire.

Ce rapport détaille le processus de création de l'interface front-end de notre système IoT, en fournissant une analyse méthodique du parcours de développement, des choix de conception jusqu'à l'intégration finale, et en soulignant la synergie de l'expérience utilisateur.

1.2 Logiciels et outlis utilisés

L'architecture de notre solution front-end a été conçue avec Android Studio et le SDK Flutter, nous permettant ainsi de créer une application cross-platform flexible et performante. Le langage de programmation Dart a été notre choix pour son efficacité et sa compatibilité avec Flutter.



FIGURE 1.1 – Logo Flutter et Dart



FIGURE 1.2 – Logo Android studio

Pour la persistance des données, nous avons intégré une base de données SQLite directement sur les appareils mobiles. Cette approche est particulièrement adaptée pour les appareils avec un espace de stockage limité et pour les utilisateurs qui n'ont pas besoin de stocker les données de température et de luminosité dans une bdd distante, à moins que la conservation des données historiques sur une longue période ne soit requise.



FIGURE 1.3 – Logo SQLite

Quant à la partie back-end de notre API, nous avons misé sur la flexibilité et la puissance de l'Arduino associé à un microcontrôleur ESP32 et à un affichage TTGO. Cette combinaison offre une interface riche pour la gestion des capteurs et la communication avec l'application front-end.



FIGURE 1.4 – Logo arduino



FIGURE 1.5 – Logo ESP32

En complément, l'intégration de ChatGPT-4 a joué un rôle clé dans le peaufinage de notre code en identifiant et corrigeant les erreurs de manière efficace. GitHub, quant à lui, a facilité le partage du code et la collaboration d'équipe, offrant une gestion des contributions et un suivi des versions des plus efficaces. Ces outils ont significativement optimisé notre flux de travail et ont renforcé l'harmonie de notre processus de développement. Figma, de son côté, a été essentiel pour le design UX/UI, en permettant de créer des maquettes d'interface utilisateur à la fois attrayantes et fonctionnelles.



FIGURE 1.6 – Logo ChatGPT-4



FIGURE 1.7 – Logo GitHub



FIGURE 1.8 – Logo Figma

1.3 Fonctionnalités de l'application Flutter

Notre application a été conçue pour offrir les fonctionnalités suivantes, en adéquation avec les maquettes Figma et leur implémentation via Flutter :

- Authentification et enregistrement des utilisateurs, assurant un accès sécurisé et personnalisé.
- Affichage des données récupérées par l'API backend, incluant les informations de température et de luminosité issues des capteurs et les enregistrer dans une BDD.
- Régulation de l'état de la LED normal (allumage et extinction) basée sur des seuils définis, en fonction des données des capteurs.
- Gestion interactive de l'état d'une LED connectée, permettant le contrôle direct via l'interface utilisateur.
- Récupération et affichage en temps réel de la localisation et de l'heure actuelle, intégrant les données contextuelles dans l'application.

Pour offrir une meilleure compréhension de l'infrastructure de notre backend, la figure 1.9 suivante illustre la configuration du microcontrôleur utilisé dans notre projet.

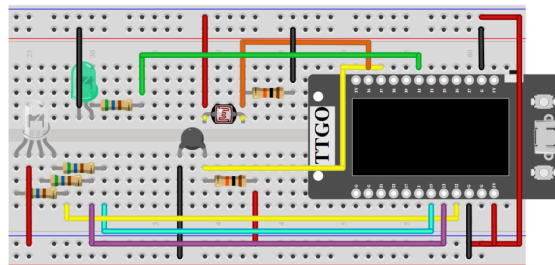


FIGURE 1.9 – Configuration de microcontrôleur

1.4 Conception et architecture

La conception de notre application a été une étape cruciale, nécessitant une modélisation approfondie pour déterminer précisément la structure et les comportements du système. Pour atteindre cet objectif, nous avons utilisé les diagrammes UML comme langage de modélisation standard. Ces diagrammes ont facilité la visualisation des concepts architecturaux et ont servi de moyen de communication efficace au sein de l'équipe de développement et avec les parties prenantes.

1.4.1 Modélisation UML :

Les diagrammes UML sont utilisés pour capturer les exigences fonctionnelles et définir les éléments structuraux de l'application.

1.4.1.1 diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des représentations graphiques des fonctionnalités systèmes et des interactions des utilisateurs. Ils sont fondamentaux pour identifier les exigences fonctionnelles et garantir que le développement est aligné avec les besoins des utilisateurs. Le diagramme de la figure 1.10 illustre les principaux scénarios d'utilisation autour de notre application Flutter, capturant l'essence des interactions utilisateur-système.

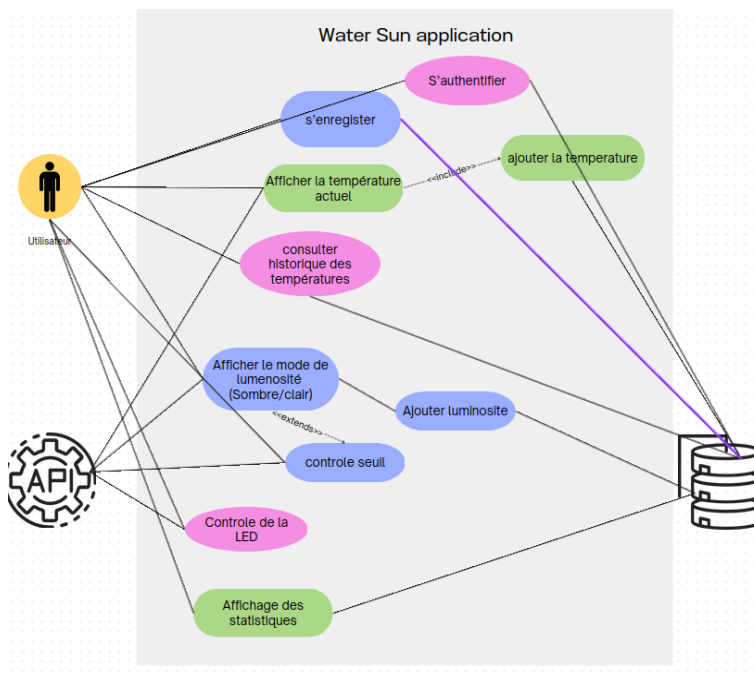


FIGURE 1.10 – Diagramme de cas d'utilisation

1.4.1.2 Diagramme de classe

Le diagramme de classes UML est un pilier du développement orienté objet. Il illustre la structure des classes qui composent notre application, définissant les attributs, les méthodes et les relations qui orchestrent la logique métier. La figure 1.14 suivante présente le diagramme de classe de notre application, soulignant comment les différentes entités interagissent pour fournir une expérience utilisateur cohérente et fonctionnelle.

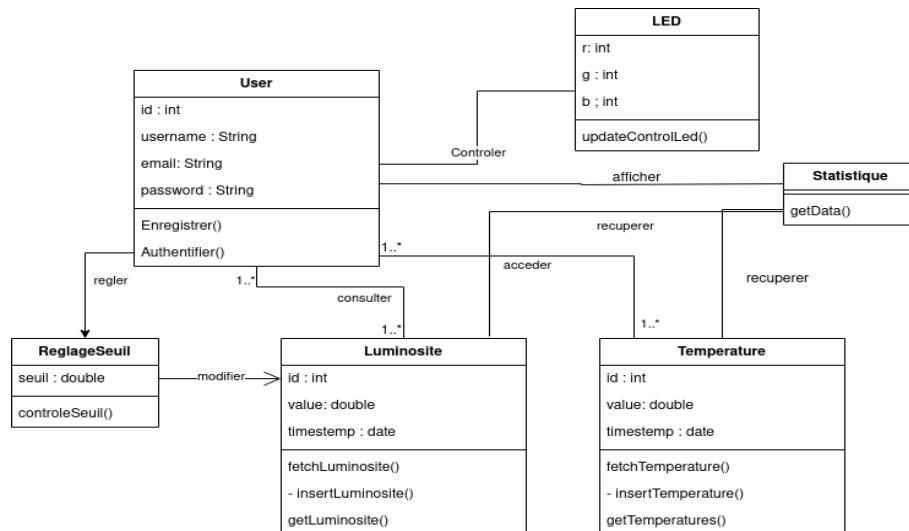


FIGURE 1.11 – Diagramme de classe

1.4.1.3 Diagramme d'État de Transition

le diagramme d'état de transition (voir Figure 1.12) capture le flux de l'expérience utilisateur, depuis l'ouverture de l'application jusqu'à la réalisation de tâches spécifiques telles que l'authentification, la visualisation des données des capteurs, le réglage de seuil de luminosité, le contrôle de la LED, et plus encore.

Chaque état représente une vue ou un statut particulier dans l'application, et les transitions illustrent les changements d'état déclenchés par les actions de l'utilisateur ou par des processus internes. Par exemple, un utilisateur peut passer de l'état "non authentifié" à l'état "authentifié" après avoir réussi le processus de connexion. De même, la modification de la luminosité ou de la couleur de la LED entraînerait un changement d'état reflété dans le diagramme.

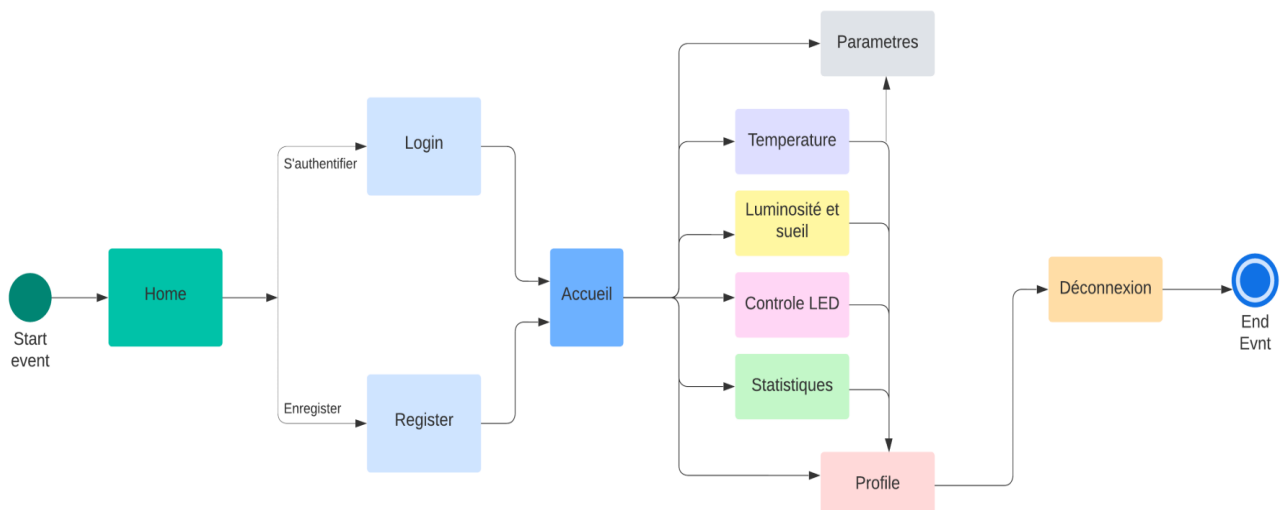


FIGURE 1.12 – Diagramme d'état de transition

1.4.2 Conception de l'architecture

Dans cette section, nous explorons l'architecture de l'application, en examinant comment les différents composants interagissent et les principes directeurs qui ont influencé nos choix architecturaux.

1.4.2.1 L'architecture Bloc :

Le modèle BLoC (Business Logic Component) est un patron de conception promu par Google, visant à séparer clairement l'interface utilisateur (UI) de la logique métier. Le cœur de BLoC repose sur les Streams pour la gestion des états dans les applications Flutter. Un BLoC reçoit des événements en entrée sous forme de Streams, les traite selon des règles métier, et produit des états en sortie également sous forme de Streams.

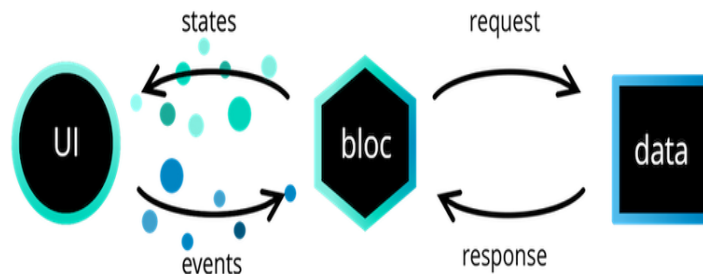


FIGURE 1.13 – Architecture Bloc

1.4.2.2 Utilisation des Streams dans BLoC :

1. **Événements (Inputs)** : Les événements sont les actions ou requêtes de l'utilisateur, transmises au BLoC comme un flux (Stream) d'événements. Par exemple, un appui sur un bouton ou un changement de texte dans un champ de saisie sont considérés comme des événements.
2. **États (Outputs)** : Le BLoC réagit à ces événements et génère des états, qui sont des représentations de la logique métier, comme les données chargées ou une valeur mise à jour. Ces états sont diffusés sous forme de Stream pour être écoutés et reflétés par l'interface utilisateur.

Les Streams sont utilisés à la fois pour gérer les entrées (événements) et les sorties (états) dans le BLoC, permettant ainsi une gestion fluide et réactive des interactions utilisateur et de la logique métier.

1.4.3 Les aspects d'ergonomie et d'expérience utilisateur

La conception de l'interface utilisateur pour notre application Flutter est une étape importante qui assure non seulement la fonctionnalité de l'application mais aussi son attrait esthétique et ergonomique. Nous avons adopté une approche méticuleuse pour développer une charte graphique cohérente et des maquettes détaillées, garantissant ainsi une expérience utilisateur fluide et intuitive.

1.4.3.1 Charte graphique :

Notre palette de couleurs a été soigneusement choisie pour favoriser une interface agréable et accessible. La combinaison de couleurs foncées et claires offre un contraste optimal pour tous les utilisateurs, y compris ceux ayant des besoins d'accessibilité. Notre objectif est de maintenir l'harmonie visuelle tout en utilisant les couleurs pour guider l'utilisateur à travers les actions et les informations importantes de l'application.

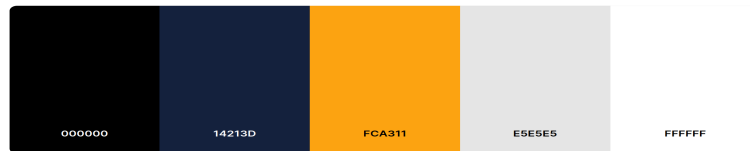


FIGURE 1.14 – La palette des couleurs de notre application

1.4.3.2 Les maquettes :

Chaque écran de l'application a été conçu pour maximiser l'utilité tout en minimisant la complexité. Les maquettes illustrent la disposition des éléments, le flux de navigation et l'interaction avec les fonctionnalités clés telles que l'authentification, le contrôle des dispositifs IoT comme les LEDs, et la visualisation des données des capteurs. Nous avons pris soin d'assurer que chaque vue soit à la fois esthétiquement plaisante et fonctionnellement riche, permettant aux utilisateurs d'accomplir leurs tâches avec efficacité et plaisir.

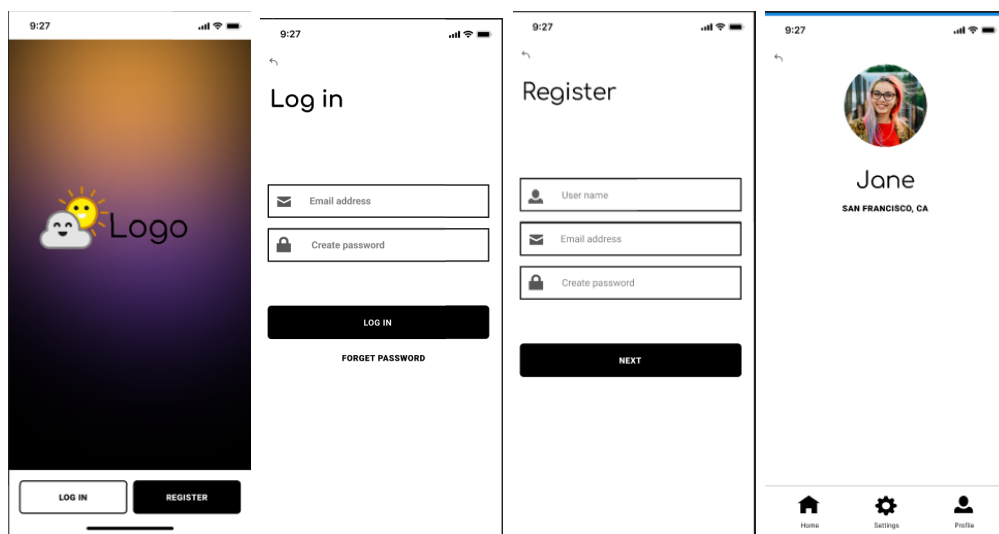


FIGURE 1.15 – Quelques maquettes de l'application -1-

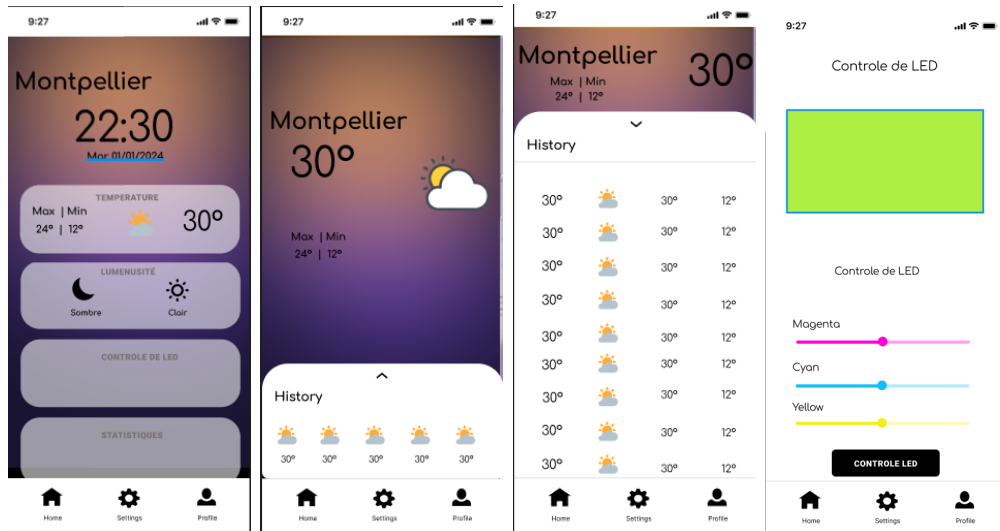


FIGURE 1.16 – Quelques maquettes de l'application -2-

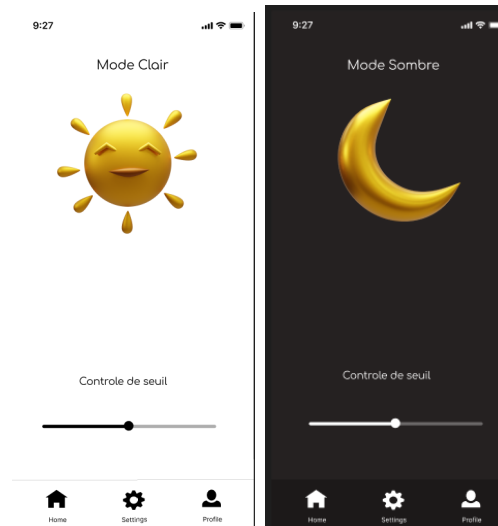


FIGURE 1.17 – Quelques maquettes de l'application -3-

L'accent mis sur la cohérence visuelle et l'ergonomie garantit que les utilisateurs de l'application bénéficient d'une expérience sans friction, qu'ils soient en train de se connecter, de visualiser des données ou de modifier les paramètres de leur dispositif. Nos maquettes servent de prototypes visuels qui guideront le développement de l'interface utilisateur, en s'assurant que le produit final répondra aux attentes des utilisateurs et se démarquera par sa qualité de conception.

1.4.3.3 Langage et theme de design

Dans le développement de notre application mobile avec Flutter, nous avons adopté une stratégie de conception hybride qui tire parti des principes de Material Design et des éléments de style Cupertino pour offrir une expérience utilisateur optimisée sur les deux principales plateformes mobiles, Android et iOS.

Material Design :

Material Design, un système de design complet conçu par Google, a été choisi comme fondement principal de notre interface utilisateur en raison de sa familiarité et de sa popularité auprès des utilisateurs d'Android. Ce langage de design est reconnu pour sa capacité à créer des interfaces cohérentes et intuitives, grâce à son ensemble exhaustif de règles et de composants. Les principes de Material Design ont été appliqués à travers l'application pour garantir une expérience utilisateur fluide et une esthétique visuelle agréable, avec l'utilisation de composants tels que des cartes, et d'autres outils.

Adoption d'Éléments Cupertino :

Nous avons choisi d'incorporer des éléments spécifiques de Cupertino, notamment pour les boutons et les champs de saisie de texte. Cette décision a été guidée par la volonté de maintenir une esthétique et une ergonomie en ligne avec les attentes des utilisateurs d'iOS. Cupertino offre des widgets qui imitent l'apparence native d'iOS, ce qui permet aux utilisateurs de cette plateforme de se sentir plus à l'aise avec l'interface utilisateur. Par exemple, les boutons et les champs de saisie, et le AppBar utilisent le style Cupertino pour bénéficier de l'aspect visuel et de la réactivité caractéristique d'iOS.

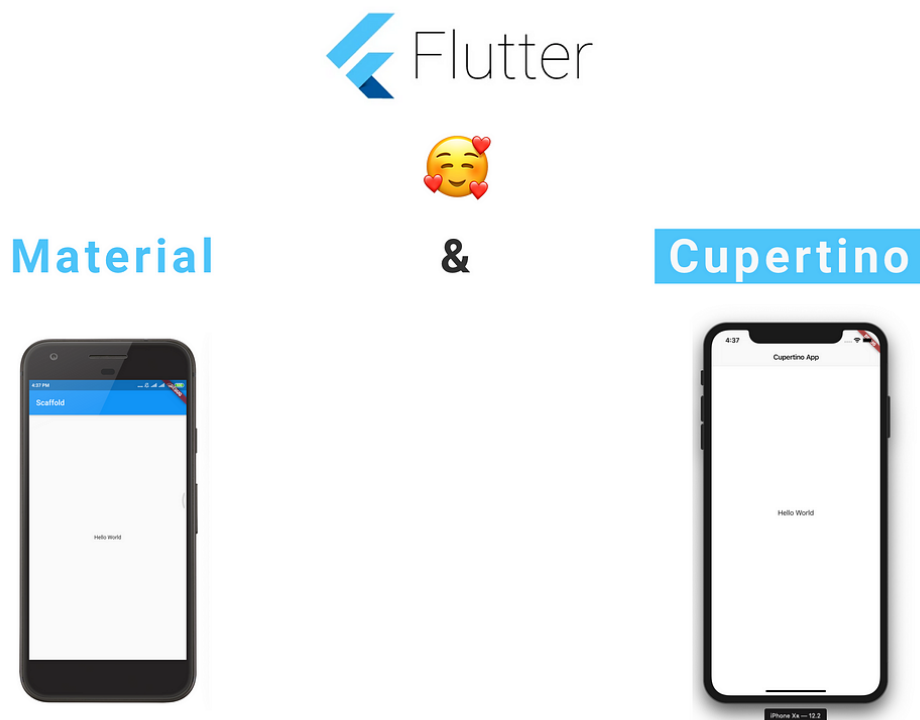


FIGURE 1.18 – Material design vs Cupertino

la combinaison des éléments de Material Design et de Cupertino dans notre application Flutter s'est avérée être une approche efficace pour accueillir les conventions de chaque plateforme. Elle a permis d'assurer une expérience utilisateur cohérente et satisfaisante, renforçant ainsi notre objectif de créer une application mobile à la fois unifiée et respectueuse des spécificités de chaque écosystème.

1.4.3.4 Animations

Dans le développement de notre application, nous avons intégré des animations pour améliorer l'expérience utilisateur en rendant l'interface plus dynamique et réactive. Deux approches principales ont été adoptées pour implémenter ces animations : les animations Tween et les AnimatedContainer.

1. Animations Tween

Les animations Tween ont été utilisées pour interpoler entre des valeurs de départ et de fin sur une durée définie, permettant une transition fluide pour des propriétés spécifiques des widgets. Cette technique est particulièrement efficace pour créer des animations personnalisées où le contrôle précis de la durée, de la courbe de vitesse et des valeurs intermédiaires est nécessaire. Par exemple, les Tween ont été appliquées pour animer la position et la taille des éléments, ainsi que pour les transitions de couleurs. L'utilisation de AnimationController en conjonction avec Tween offre une flexibilité complète dans la gestion du cycle de vie de l'animation et dans la synchronisation avec les interactions des utilisateurs.

2. AnimatedContainer

Pour les animations qui nécessitent des changements de plusieurs propriétés en même temps, comme la largeur, la hauteur et la couleur, AnimatedContainer a été utilisé. Ce widget est un choix pratique pour les animations simples où les états de début et de fin sont bien définis et où moins de contrôle programmatique est requis. En fournissant une durée d'animation et une courbe, AnimatedContainer gère automatiquement la transition entre l'état actuel et l'état modifié du widget. Par exemple, nous avons utilisé AnimatedContainer pour réaliser des effets de survol sur des boutons, où la couleur de fond et la taille changent en réponse aux interactions des utilisateurs.

Le choix entre Tween et AnimatedContainer a été dicté par la nature de l'animation requise et par la complexité des transitions. Alors que les animations Tween offrent plus de contrôle et sont idéales pour les animations sur mesure, AnimatedContainer permet une implémentation rapide et moins verbeuse pour des transitions standard. En combinant ces deux types d'animations, nous avons pu réaliser une interface utilisateur riche et vivante, qui répond aux actions des utilisateurs de manière intuitive et engageante.

1.4.3.5 Thème :

Dans notre application que nous avons appelé "Water Sun", le thème a été soigneusement conçu pour offrir une expérience utilisateur harmonieuse et attrayante. Le thème utilise la couleur bleue comme teinte principale, évoquant fiabilité et sérénité, et est défini par primarySwatch : Colors.blue dans ThemeData. La densité visuelle est adaptée à la plateforme, assurant une interface utilisateur agréable et accessible tant sur Android que sur iOS. Le choix typographique s'articule autour de la police 'Roboto', offrant une lisibilité optimale et un style moderne. Cette cohérence esthétique est complétée par une navigation fluide, avec des routes définies pour chaque écran principal, garantissant une expérience utilisateur intuitive et facile. Ce thème reflète l'objectif de créer une application à la fois fonctionnelle et esthétiquement plaisante.

1.5 Implémentation des Interfaces avec l'API

Dans le cadre du développement de notre application Water Sun, nous avons conçu et implémenté une série d'interfaces utilisateur (UI) qui communiquent efficacement avec notre API backend. Ces interfaces sont conçues pour être à la fois esthétiquement agréables et fonctionnellement robustes, permettant aux utilisateurs d'interagir avec l'application de manière intuitive et productive.

1.5.1 Interfaces Utilisateur (UI)

Chaque interface a été développée pour répondre aux besoins spécifiques de fonctionnalité, tout en maintenant une expérience utilisateur cohérente à travers l'application.

1.5.1.1 Interface Home

L'interface Home est la première vue présentée à l'utilisateur dans notre application Water Sun. Elle joue un rôle important en donnant la première impression de l'application, affichant le logo et le nom, et possède un arrière-plan capturé d'une animation que nous avons implémenter sur un autre projet flutter, offrant une expérience visuelle dynamique.

Cette interface contient deux boutons principaux : 'Login' et 'SignIn'. Le bouton 'Login' dirige les utilisateurs vers l'écran de connexion, tandis que 'SignIn' ouvre l'interface d'inscription, facilitant ainsi l'entrée de l'utilisateur dans l'application. la figure 1.19 montre l'interface implémenter avec flutter.

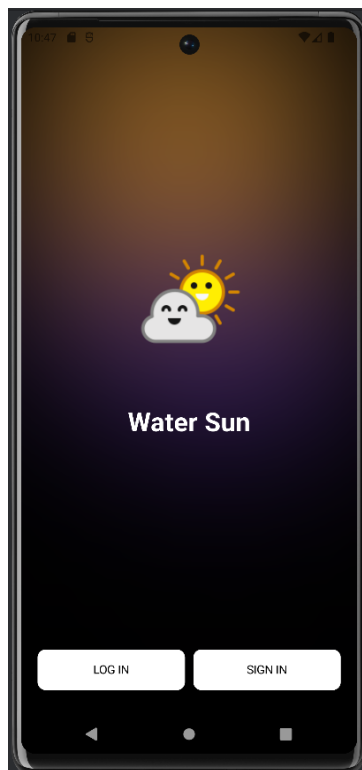


FIGURE 1.19 – Interface de Home

1.5.1.2 interface d'inscription

L'interface Registration est conçue pour permettre aux utilisateurs anonymes de créer un compte, étape essentielle pour accéder aux fonctionnalités complètes de l'application "Water Sun". Cette interface guide l'utilisateur à travers un processus d'inscription simple et intuitif.

Pour s'inscrire, l'utilisateur doit fournir un nom d'utilisateur, une adresse e-mail valide, et créer un mot de passe sécurisé. Ces informations sont essentielles pour établir un compte utilisateur unique et sécurisé. Le processus est conçu pour être aussi fluide et convivial que possible, en veillant à ce que l'utilisateur fournisse toutes les informations nécessaires tout en minimisant les obstacles à l'entrée.

Une fois l'inscription réussie, l'utilisateur est redirigé vers l'interface d'accueil de l'application. Cette transition marque le début de son expérience personnalisée au sein de l'application, où il peut désormais accéder à toutes les fonctionnalités offertes.

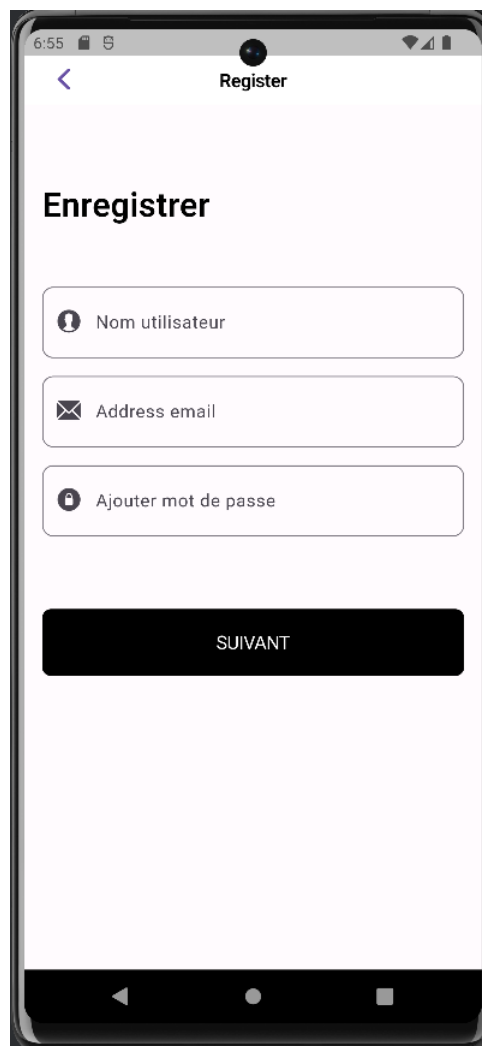
A smartphone mockup displaying the registration interface. The status bar at the top shows the time 6:55 and various icons. The app bar has a back arrow and the title "Register". The main heading is "Enregistrer". There are three input fields: "Nom utilisateur" with a person icon, "Address email" with an envelope icon, and "Ajouter mot de passe" with a lock icon. Below these fields is a large black button labeled "SUIVANT". The bottom of the screen shows the standard Android navigation bar.

FIGURE 1.20 – Interface de registration

1.5.1.3 Interface Connexion

L'interface de Connexion est obligatoire dans notre application WaterSun, offrant aux utilisateurs un accès sécurisé à leur compte et aux fonctionnalités de l'application. Cette interface est conçue pour être à la fois simple et sécurisée, assurant une expérience utilisateur fluide et fiable.

Pour se connecter, l'utilisateur doit entrer son adresse e-mail et son mot de passe. Ces informations sont ensuite vérifiées contre les données stockées dans notre base de données pour authentifier l'utilisateur. Cette étape est essentielle pour garantir que l'accès est accordé uniquement aux utilisateurs légitimes, renforçant ainsi la sécurité de l'application.

Une fois l'authentification réussie, l'utilisateur est redirigé vers l'interface d'accueil de l'application. Cela lui permet d'accéder à toutes les fonctionnalités disponibles, y compris celles fournies par notre API backend. Cette transition marque une étape importante dans l'expérience utilisateur, lui offrant un accès complet aux services et fonctions de WaterSun.



FIGURE 1.21 – Interface de connexion

1.5.1.4 Interface Accueil

L'interface d'accueil est le cœur de notre application WaterSun, où les utilisateurs peuvent interagir avec les diverses fonctionnalités offertes par l'API. Cette interface est conçue pour offrir une expérience utilisateur intuitive, permettant un accès facile à des informations et des contrôles clés.

Sur cette interface, les utilisateurs ont accès à des fonctionnalités telles que le suivi de la température, la luminosité, le contrôle des LED, et l'affichage des statistiques. Ces fonctionnalités sont intégrées de manière à fournir un aperçu immédiat et interactif des données pertinentes, améliorant ainsi l'engagement et l'expérience utilisateur.

En plus de ces fonctionnalités, l'interface d'accueil récupère et affiche également la localisation actuelle de l'utilisateur, ainsi que la date et l'heure du jour. Cette intégration offre un contexte personnalisé et améliore la pertinence des données présentées.

L'interface comprend également un BottomNavigationBar qui facilite la navigation dans l'application. Cet élément de navigation permet aux utilisateurs de basculer facilement entre l'écran d'accueil, leur profil utilisateur, et les paramètres de l'application. Cette barre de navigation est conçue pour être à la fois fonctionnelle et esthétiquement plaisante, assurant une expérience utilisateur cohérente et efficace.

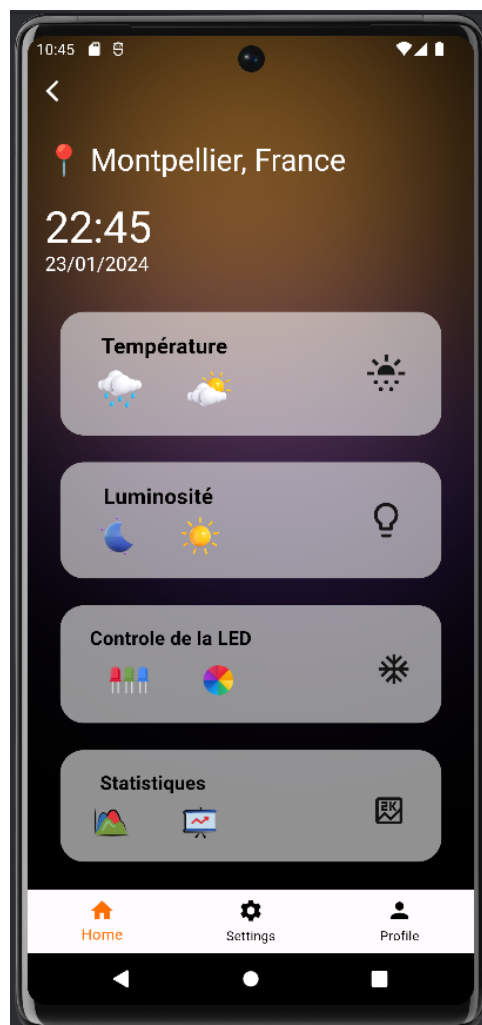


FIGURE 1.22 – Interface d'accueil

1.5.1.5 interface Temperature

L'interface Température est une composante essentielle de notre application Water-Sun, conçue pour fournir aux utilisateurs des informations précises et actualisées sur la température. Cette interface interagit directement avec le dispositif ESP32 ou utilise une API externe pour obtenir les données de température.

Une fois les données de température capturées, elles sont converties en degrés Celsius pour assurer une compréhension universelle et facilitent la comparaison. Cette fonctionnalité est particulièrement utile dans des contextes où la précision des mesures de température est cruciale.

En plus d'afficher la température actuelle, cette interface enregistre chaque mesure dans un historique de températures. Cela permet aux utilisateurs de suivre l'évolution de la température au fil du temps. L'utilisateur peut également consulter les valeurs minimales et maximales capturées, offrant ainsi une perspective complète sur les variations de température.

L'historique des températures est présenté de manière intuitive, permettant aux utilisateurs de naviguer facilement à travers les données passées pour une analyse approfondie. Cette fonctionnalité renforce l'utilité de l'interface Température, la rendant non seulement un outil pour la consultation instantanée mais aussi un instrument d'analyse à long terme.

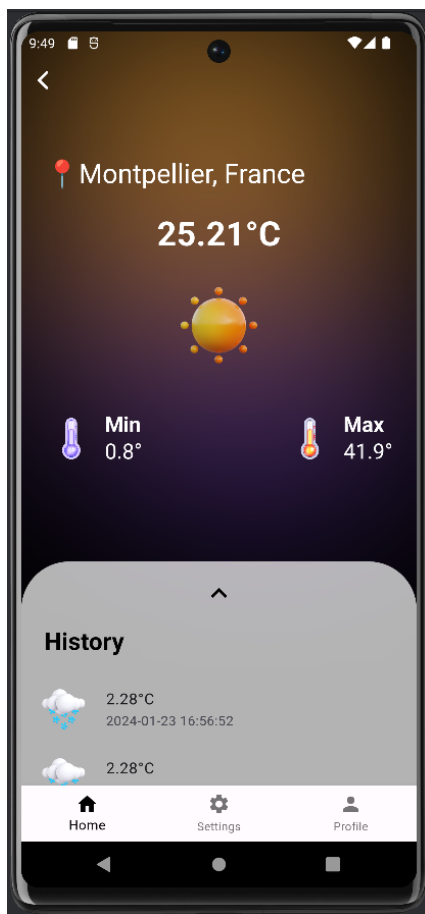


FIGURE 1.23 – Interface Temperature



FIGURE 1.24 – Interface liste des Températures

1.5.1.6 interface luminosité

L'interface luminosité dans notre application WaterSun joue un rôle crucial en capturant et affichant les données de luminosité fournies par notre API backend. Cette interface est dotée d'une fonctionnalité unique qui adapte son mode d'affichage en fonction des valeurs de luminosité capturées.

Le comportement de l'interface est le suivant :

Mode Clair : Activé lorsque la luminosité capturée est supérieure à 1000 et inférieure à 4000. Ce mode offre une interface lumineuse et claire, idéale pour des conditions de forte luminosité.

Mode Sombre : Activé lorsque la luminosité est inférieure à 1000 et supérieure à 500. Cette configuration offre une interface sombre, réduisant la fatigue oculaire dans des environnements à faible luminosité.

L'utilisateur a également la possibilité de modifier le seuil de déclenchement pour le changement de mode via un slider. Ce slider permet de régler le seuil de luminosité entre 500 et 4000, avec des limites imposées pour assurer la cohérence avec les capacités de notre API backend.



FIGURE 1.25 – UI mode clair

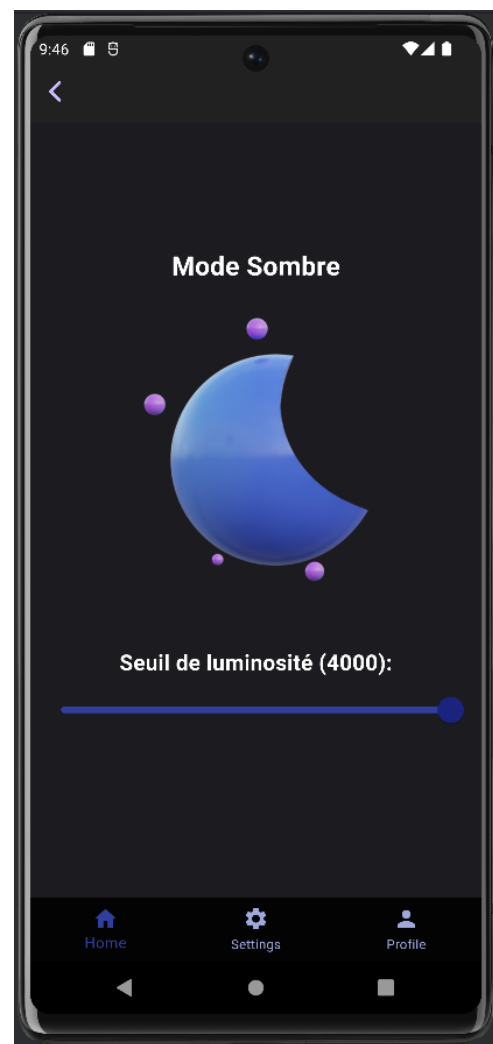


FIGURE 1.26 – UI mode sombre

1.5.1.7 Interface Contrôle de la LED

L'interface de contrôle de la LED de notre application WaterSun offre une fonctionnalité interactive qui permet à l'utilisateur de manipuler une LED RGB via un mélangeur de couleurs CMY (Cyan, Magenta, Jaune). Inspirée du modèle de mélange des couleurs soustractives [CMY](#), cette interface offre une expérience utilisateur intuitive et ludique pour la sélection des couleurs.

L'utilisateur ajuste la couleur à l'aide de trois curseurs correspondant aux composants CMY de la couleur finale :

Slider Cyan : Permet de régler la quantité de cyan dans le mélange.

Slider Magenta : Ajuste la quantité de magenta.

Slider Jaune (Yellow) : Contrôle la quantité de jaune.

Chaque ajustement est reflété en temps réel dans un aperçu de couleur, donnant à l'utilisateur un retour visuel immédiat de la couleur actuelle résultante. Un bouton 'Contrôler la LED' permet de confirmer la sélection et d'envoyer les valeurs à l'API backend qui contrôle physiquement la LED RGB.

Lorsque l'utilisateur a choisi et appliqué une couleur, si la LED RGB du côté backend change avec succès de couleur, un message de confirmation est affiché à l'écran sous forme de toast, informant l'utilisateur que la couleur a été mise à jour avec succès.

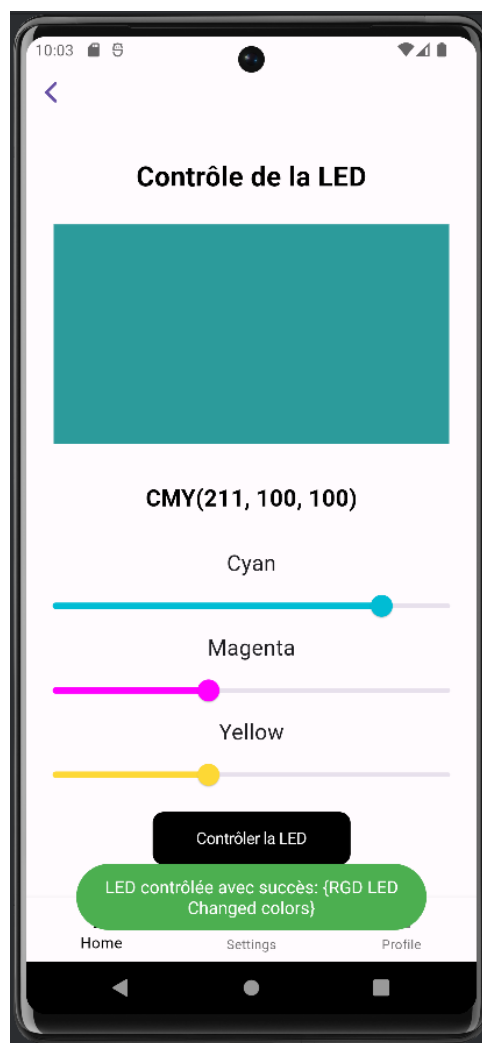


FIGURE 1.27 – Interface de controle de LED

Cette interface illustre l'engagement de notre application à fournir une interaction utilisateur enrichissante et efficace, en reliant l'expérience numérique de l'application aux éléments physiques contrôlés par l'utilisateur.

1.5.1.8 Interface des Statistiques

L'interface des statistiques est un outil analytique puissant au sein de l'application WaterSun, permettant aux utilisateurs de visualiser des données clés de manière graphique et intuitive. Cette interface affiche des statistiques détaillées sur la température et la luminosité capturées et traitées par l'API backend et qui sont stockées dans la base de données sqlite.

Pour la température, l'utilisateur peut consulter un graphique linéaire représentant la température par minute, qui offre un aperçu temporel des variations de température. En complément, un graphique circulaire présente une répartition des différentes plages de température, offrant une analyse comparative de la distribution des valeurs.

En faisant défiler l'écran vers le bas, l'utilisateur peut également accéder à un graphique de la luminosité, qui trace une courbe représentant les variations de la luminosité au fil du temps. Cette visualisation permet à l'utilisateur de comprendre les tendances et les modèles dans les données de luminosité.

Les deux figures 1.28 1.29 ci-dessous illustrent l'interface des statistiques de température et de luminosité.

Ces outils de visualisation des données sont conçus pour être à la fois informatifs et esthétiquement plaisants, garantissant une expérience utilisateur à la fois enrichissante et engageante.

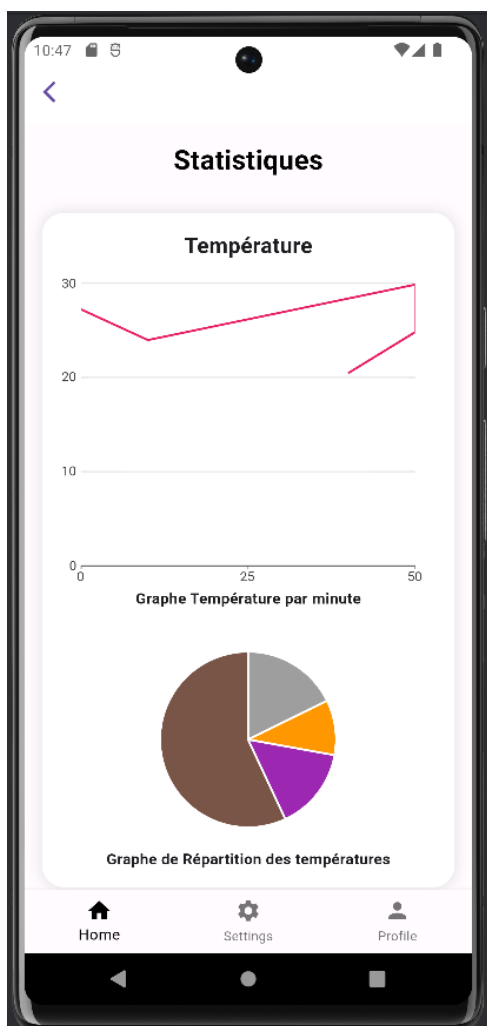


FIGURE 1.28 – UI statistique temperature

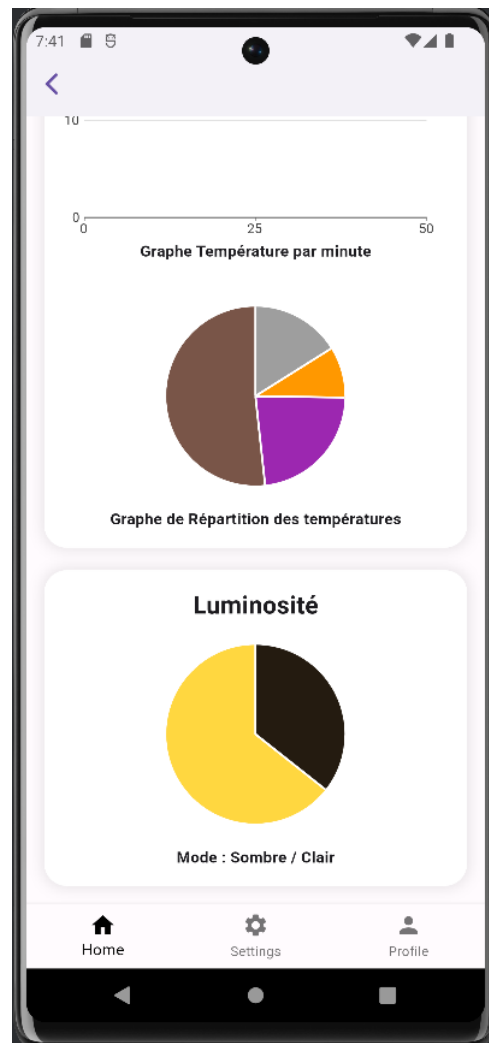


FIGURE 1.29 – UI statistique luminosité

1.5.1.9 interface profile utilisateur

L'interface du profil utilisateur est un espace personnel où les utilisateurs de l'application WaterSun peuvent visualiser et gérer leurs informations personnelles. Cette interface affiche la photo de profil de l'utilisateur, son nom ainsi que son adresse e-mail, offrant un résumé immédiat de l'identité de l'utilisateur au sein de l'application.

L'interface est conçue pour être épurée et centrée sur l'utilisateur, mettant en avant les informations les plus pertinentes. Un menu accessible depuis cette page offre des options supplémentaires telles que la déconnexion de l'application ou l'accès aux paramètres du profil. Ces fonctionnalités permettent une gestion facile de l'identité numérique de l'utilisateur et une personnalisation de son expérience d'application.

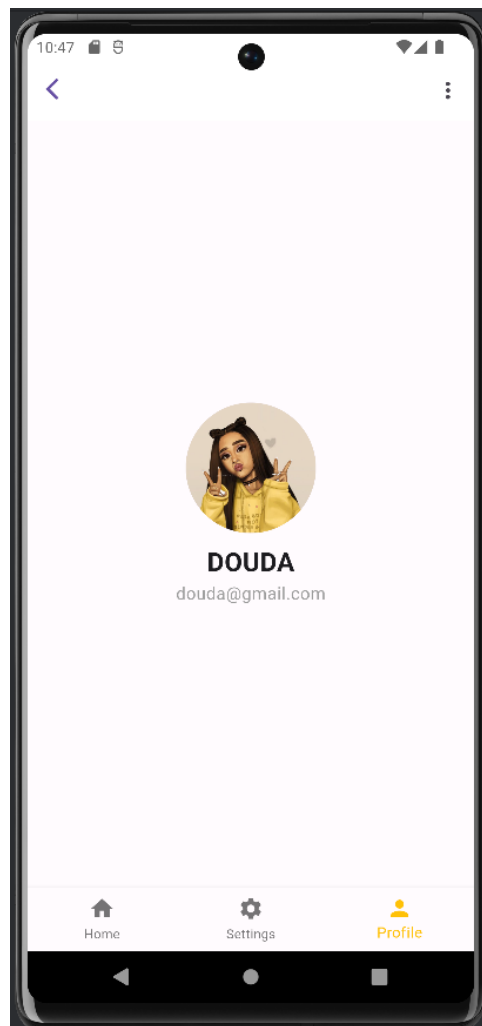


FIGURE 1.30 – Interface profile utilisateur

1.5.1.10 Interface Paramètres

L'interface Paramètres de l'application WaterSun offre aux utilisateurs la possibilité de personnaliser leur expérience selon leurs préférences. Accessible via le BottomNavigationBar, cette interface est un hub central pour ajuster divers aspects de l'application.

Les utilisateurs peuvent activer ou désactiver les notifications grâce à un interrupteur, ce qui leur permet de contrôler la réception d'informations importantes de l'application. Une option pour changer le thème de l'application est également disponible, permettant un basculement entre un thème clair et sombre, adaptant ainsi l'interface aux conditions de luminosité ou aux préférences personnelles.

De plus, un lien vers une page "À propos" est inclus, fournissant des informations supplémentaires sur l'application, telles que la version actuelle et les crédits de développement.



FIGURE 1.31 – Interface parametres

L'interface est conçue pour être intuitive, permettant une navigation et des ajustements faciles sans distraire de l'expérience globale de l'application. Ce design épuré et centré sur l'utilisateur facilite la gestion des préférences personnelles et la découverte de fonctionnalités supplémentaires de l'application.

1.5.2 Intégration avec l'API Backend

Notre application WaterSun communique avec un backend via une API RESTful pour interroger et manipuler les données de température, de luminosité, et contrôler une LED RGB. Les logs ci-dessous illustrent les interactions réussies entre l'application et l'API backend.

1.5.2.1 Log de Température

Les requêtes de température montrent que l'application interroge avec succès l'API, envoie des données de température en temps réel, et l'application flutter affiche ces données et les insère dans la base de données locale. Voici un exemple de log lors de l'ajout d'une mesure de température :

```
I/flutter (12051): Tentative d'ajout dans la BDD
I/flutter (12051): La date de l'ajout 2024-01-24 19:13:54.801629Z
I/flutter (12051): URI : http://192.168.43.10/temperature
I/flutter (12051): Insertion réussie de la temperature avec l'ID: 182
```

FIGURE 1.32 – Log de requête de température

1.5.2.2 Log de Luminosité

De manière similaire, l'application récupère les données de luminosité et adapte l'interface utilisateur en fonction des valeurs reçues, en activant le mode sombre ou clair selon le seuil défini. dans ce cas le seuil défini est supérieure a 1555 donc le mode sombre est actif.

```
I/flutter (12051): Chargement...
I/flutter (12051): Value lumiere : 1555
I/flutter (12051): URI luminosité : http://192.168.43.10/light
I/flutter (12051): Insection des données de luminosité ...
I/flutter (12051): is Dark mode ? true
I/flutter (12051): Insérer avec succes
```

FIGURE 1.33 – Log de requête de luminosité

1.5.2.3 Log de Contrôle de la LED

Les requêtes de contrôle de la LED montrent que l'application envoie des commandes pour modifier la couleur de la LED RGB via l'API, pour le moment le backend n'a pas capture cette requetes, donc il est en mode chargement (a savoir des fois il ya des difficultés de communication avec l'esp 32 a cause de matériel qui n'est pas bien).

```
I/flutter (12051): 100 196 100
I/flutter (12051): URL de la requête: http://192.168.43.10/led?magenta=196&cyan=100
I/flutter (12051): Chargement...
```

FIGURE 1.34 – Log de contrôle de la LED

Ces logs démontrent la robustesse et l'efficacité de l'intégration de l'application avec le backend, validant ainsi la communication bidirectionnelle et la réactivité du système dans son ensemble.

1.5.3 Défis Rencontrés et Solutions Adoptées

Tout au long du développement de notre application WaterSun, nous avons rencontré divers défis techniques, parmi lesquels l'utilisation de l'architecture BLoC (Business Logic Component) s'est révélée particulièrement complexe.

Défi : Utilisation de l'Architecture BLoC

Notre intention initiale était d'utiliser des classes séparées pour les états et les événements dans l'architecture BLoC, afin de récupérer et de gérer les données de l'API de manière asynchrone. Cependant, nous avons rencontré des difficultés pour récupérer les données périodiquement et de façon récurrente.

Solution : Streams et Timer

Pour résoudre ce problème, nous avons opté pour l'intégration de streams au sein de nos BLoCs, ce qui nous a permis de récupérer les données de manière réactive. Nous avons également implémenté un timer qui déclenche la récupération des données toutes les deux minutes, ce qui a été ajusté à cinq secondes pour les besoins de nos tests.

Cette approche nous a permis de garantir que l'application pouvait récupérer les données les plus récentes sans nécessiter d'interaction utilisateur, assurant ainsi que les informations affichées restent actualisées et pertinentes.

1.5.4 Améliorations Futures

Dans la perspective d'une évolution constante de l'application WaterSun, plusieurs améliorations sont envisagées pour enrichir davantage l'expérience utilisateur et optimiser la performance de l'application.

Intégration d'une Base de Données Firebase

Nous prévoyons d'intégrer une base de données Firebase pour bénéficier de sa capacité de synchronisation en temps réel et de sa gestion simplifiée des données. Ceci permettra une meilleure scalabilité de l'application et une expérience utilisateur plus fluide, en particulier pour les fonctionnalités qui nécessitent une persistance des données et une collaboration en temps réel entre les utilisateurs.

Dynamisation de l'Interface Paramètres

L'interface des paramètres, actuellement statique, sera repensée pour devenir dynamique. Cela inclura l'implémentation de fonctionnalités interactives qui répondent aux actions des utilisateurs, telles que l'activation ou la désactivation des notifications, et la personnalisation du thème de l'application. Cette amélioration rendra l'interface plus intuitive et répondra mieux aux besoins spécifiques des utilisateurs.

Amélioration de l'Interface Profil Utilisateur

L'interface de profil utilisateur fera l'objet d'une refonte pour offrir plus d'options de personnalisation et de gestion du profil. Cela pourrait inclure la modification des informations personnelles, la gestion des préférences de l'application, et l'intégration de fonctionnalités sociales telles que l'ajout d'amis ou le partage de données.

Ajout de Nouvelles Fonctionnalités

Nous envisageons également d'ajouter de nouvelles fonctionnalités pour étendre les capacités de l'application. Cela pourrait inclure l'intégration de capteurs supplémentaires, l'amélioration des outils analytiques pour les données environnementales, et l'introduction de modules d'apprentissage automatisé pour prédire les tendances des données recueillies.

Ces améliorations sont guidées par notre engagement à offrir une application de haute qualité qui non seulement répond aux besoins actuels de nos utilisateurs mais est également adaptable aux évolutions technologiques futures.

1.6 Conclusion

En conclusion, le développement de l'application WaterSun a été un parcours enrichissant et instructif qui a mis en évidence la puissance et la flexibilité de Flutter comme plateforme de développement d'applications mobiles. Nous avons réussi à créer une interface utilisateur intuitive, à intégrer une communication fluide avec l'API backend et à implémenter une série de fonctionnalités qui améliorent l'expérience utilisateur.

Les défis rencontrés en cours de route, en particulier ceux liés à l'architecture BLoC et à la récupération périodique des données, nous ont permis d'approfondir notre compréhension des modèles de conception réactive et de la gestion d'état dans les applications complexes. Les solutions adoptées, telles que l'intégration des streams et l'utilisation des timers, ont non seulement résolu les problèmes en question mais ont également ouvert la voie à des améliorations futures.

En regardant vers l'avenir, nous envisageons plusieurs améliorations pour WaterSun. L'adoption d'une base de données Firebase offrira une persistance des données plus robuste et une synchronisation en temps réel. Nous prévoyons de rendre l'interface des paramètres dynamique, avec des contrôles interactifs pour les utilisateurs, et d'enrichir l'interface de profil pour une personnalisation plus poussée. De plus, nous explorons de nouvelles fonctionnalités qui continueront à faire évoluer WaterSun et à répondre aux besoins changeants de nos utilisateurs.

Ce projet a démontré la capacité de notre équipe à s'adapter, à innover et à surmonter les obstacles techniques avec détermination. Nous sommes confiants que les fondations établies ici serviront de tremplin pour de futures améliorations et que WaterSun continuera à se développer et à prospérer dans le paysage des technologies mobiles.