



Murmure - Dossier Projet

Barthélémy POUSSET

Janvier 2026

—
Titre RNCP 6

Développeur Web et Application Mobile



Sommaire

1. Introduction.....	2
Partie 1 : Conception et maquettage d'une application web & mobile.....	4
1. Description du projet.....	5
2. Storyboard.....	7
3. Panel utilisateur & User Journey.....	12
4. WireFrame.....	16
5. Ulkit.....	18
6. Mockups.....	19
Partie 2 : Pilotage du projet d'une application web & mobile.....	20
1. Préparation des sprints de développement :.....	21
2. Tableau Kanban (Trello):.....	22
3. Analyse des compétences nécessaires :.....	23
4. Déroulé des sprints de développement :.....	23
5. Compte rendu reunion:.....	25
Partie 3 : Design de l'architecture BDD et développement d'une application mobile...29	
1. Schéma global d'architecture.....	30
2. Identification des librairies et services utilisées.....	31
3. Schéma de base de données.....	32
4. Routes et endpoints backend :.....	35
5. Schéma des composants React & React Native.....	36
6. Navigation et interface graphique (UX/UI):.....	38
7. Approche TDD:.....	40
8. Authentification:.....	41
Partie 4 : Mise en production du projet.....	42
1. Schéma de l'environnement de déploiement.....	43
Conclusion.....	45

1. Introduction

La santé mentale présente un enjeux crucial pour nous et la génération future. En 2021, 12,5 % de la population française rapporte avoir vécu un épisode dépressif caractérisé (EDC)¹.

C'est en 2021 que Santé publique France interroge 24 514 adultes âgés de 18 à 85 ans et présente un constat inquiétant:

“Le nombre de personnes touchées par la dépression en France est en augmentation”

En effet, **en 2017**, on parle de **9,8 %** de la **tranche 18-75 ans** qui est touchée par un EDC et quatre ans plus tard, **en 2021** c'est désormais **13,3 %** de cette même tranche d'âge qui est touchée.

Ici un tableau résumant les principales populations touchées dans l'étude de 2021 (notez le score particulièrement élevé des jeunes adultes):

Population	Chiffres
Hommes	9,3 %
Femmes	15,6 %
Jeunes adultes	20,8%
Enfant/Ado	8%

C'est fort de ce constat que j'ai eu l'idée d'une application mobile centralisant:

- des cours sur la psychologie;
- des outils pour aider à la relaxation (méditation, respiration);
- des fiches résumées type flashcard;
- un chat pour communiquer avec une présence bienveillante

L'univers de la psychologie a déjà été exploité maintes fois pour créer des applications (il suffit de chercher “relaxation” sur un store applicatif pour voir le nombre d'acteurs présent sur ce marché). Par conséquent il était nécessaire de se démarquer et l'utilisation de la **gamification**² à été ici un élément capital pour rendre l'expérience plus amusante et attrayante.

¹ Source: santé public france

<https://www.santepubliquefrance.fr/maladies-et-traumatismes/sante-mentale/depression-et-anxiete/documents/article/prevalence-des-episodes-depressifs-en-france-chez-les-18-85-ans-resultats-du-barometre-sante-2021>

² Appliquer les mécanismes du jeu à des contextes non ludiques. Exploite la tendance humaine à jouer, le système de récompense et permet d'augmenter l'engagement utilisateur.

J'ai toujours été fasciné par la gamification, le principe de rendre l'apprentissage amusant, jouer avec du contenu sérieux. Par ce biais, l'utilisateur retient aussi les informations plus efficacement.

J'ai donc pitché cette idée à ma classe et le projet a été retenu. Un équipe de 5 personnes a par la suite été constituée:

- Elisa Gomez BENEDI
- Stephanie DOLET
- Bat HUYNH
- Pierre FRANCES
- Barthélémy POUSSET (moi même)

J'en profite pour les remercier sincèrement. Chacun à apporté ses idées, son temps et un réel engagement qui à permis d'amener à la vie ce qui n'était qu'une vague idée dans mon esprit il y a encore 1 mois.

Vous trouverez dans les pages suivante les étapes clés de la réalisation de ce projet, qui structurent également ce dossier:

Partie	Objectif Principal	Compétence RNCP Associée
Partie 1 : Conception et maquettage	Définir le concept, la cible et la proposition de valeur unique (UVP). Concevoir l'UI et l'UX (storyboards, user journeys, wireframes, mock-ups).	Conception et maquettage d'une application
Partie 2 : Pilotage du projet	Organiser et répartir le travail au sein de l'équipe Définir l'architecture technique (user stories, board kanban, schéma de base de données, déroulé des sprints).	Pilotage du projet d'application web & mobile
Partie 3 : Développement	coder l'application (front et back) et la base de données code source, technologies, architecture, justification des choix).	Conception et manipulation de la base de données & Développement d'une application web et mobile
Partie 4 : Mise en production	Finaliser et déployer l'application, en assurant sa sécurité et sa stabilité (lien vers le site, schéma de l'environnement, TDD, authentification).	Mise en production d'une application web et mobile

Ce dossier est organisé de manière à suivre chronologiquement le processus de développement du projet, depuis l'idée initiale jusqu'à la mise en ligne du produit final.



Partie 1 : Conception et maquettage d'une application web & mobile

1. Description du projet

Problématique

En France, la santé mentale est un enjeu majeur, la prévalence de l'Épisode Dépressif Caractérisé (EDC) ayant grimpé de **9,8 % en 2017 à 13,3 % en 2021** (12,5 % de la population totale). Ce taux touche particulièrement les **femmes** (15,6 %) et, de manière critique, **un jeune adulte sur cinq (20,8 %)**. C'est en réponse à cette forte prévalence chez les jeunes qu'a été développée l'application mobile **Murmure**.

UVP (unique value proposition)

Proposer une application qui combine:

- Des **cours sur la psychologie** (fonctionnement du cerveau, des émotions et régulation de ces dernières)
- Des **fiches de mémorisation** une fois le cours terminé.
- Des **exercices physiques** (méditation, respiration) pour pratiquer les concepts vus dans les cours.
- Un **chatbot** représenté par une mascotte dans l'application qui, **connecté à une IA**, permettra de converser sur des sujet de psychologie (via un contexte fournis au LLM³)

Aussi l'application reposera grandement sur le **principe de gamification** en proposant une interface et une expérience utilisateur attrayante et encourageante.

L'interface doit reprendre les codes du jeux vidéo avec des **menus dessinés**, une **carte** pour l'écran de progression et la présence d'une **mascotte** disponible sur tous les écrans de l'application et qui dirigera vers le chat (Coco, représentée par un perroquet).

Cible

Notre application cible principalement les **jeunes adultes (18-25 ans)**, catégorie la plus touchée par la dépression (20,8 % selon Santé publique France 2021). Elle vise toute personne cherchant des outils de psychologie et de relaxation émotionnelle dans un format ludique.

Description courte et Devise

Application d'Apprentissage psychologique

Murmure: Explore ton monde intérieur

³ Modèle d'IA entraîné sur des données textuelles massives, le **LLM (Large Language Model)** comprend et génère du langage naturel pour de multiples tâches.

Spécificités

Nous avons souhaité inclure les fonctionnalités suivante dans notre app:

- UI/UX:
 - Utilisation de background dessinées sur plusieurs écrans (généré par LLM),
 - Présence de la mascotte sur tous les écrans,
 - Le menu chat apparaît en tiroir (page poussée du bas vers le haut et refermable en swipant vers le bas)
 - Navigation entre les écrans via des zones cliquables (ces zones seront animées),
- IA:
 - Connexion du chat à l'API d'Hugging Face⁴
 - Utilisation d'un contexte pour donner un rôle au LLM
- BDD/Data:
 - Stockage du contenu des cours et flashcards
 - Stockage des urls vers les méditations
 - Gestion de la connexion user
 - Gestion de la progression sur le cours
- Lecteur Audio
 - Utilisation de la librairie Expo-AV pour la lecture audio dans le menu méditation
- Vibration
 - Utilisation de la librairie Expo-haptics pour la gestion du module de vibration du téléphone dans le menu respiration

Nous avons disposé d'une équipe de 5 personnes et 10 jours ouvrés (2 semaines) pour planifier, organiser, développer, tester et déployer en production ce projet.

⁴ Service d'hébergement de modèles d'intelligence artificielle et de *machine learning* (LLM, modèles de langage, etc.).

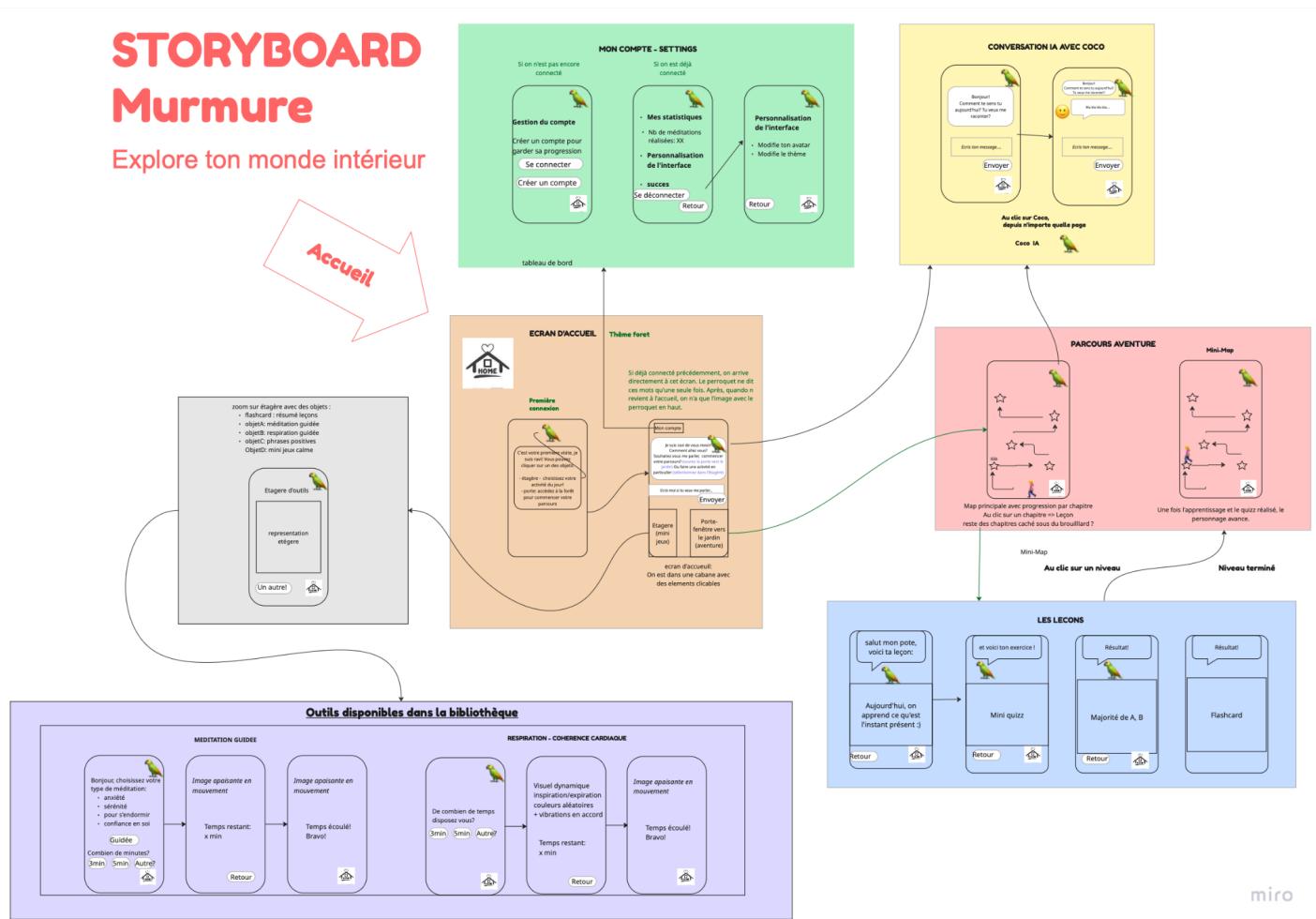
2. Storyboard

Pour la création du storyboard nous nous sommes servi de Miro, une application en ligne facilitant la création collaborative.

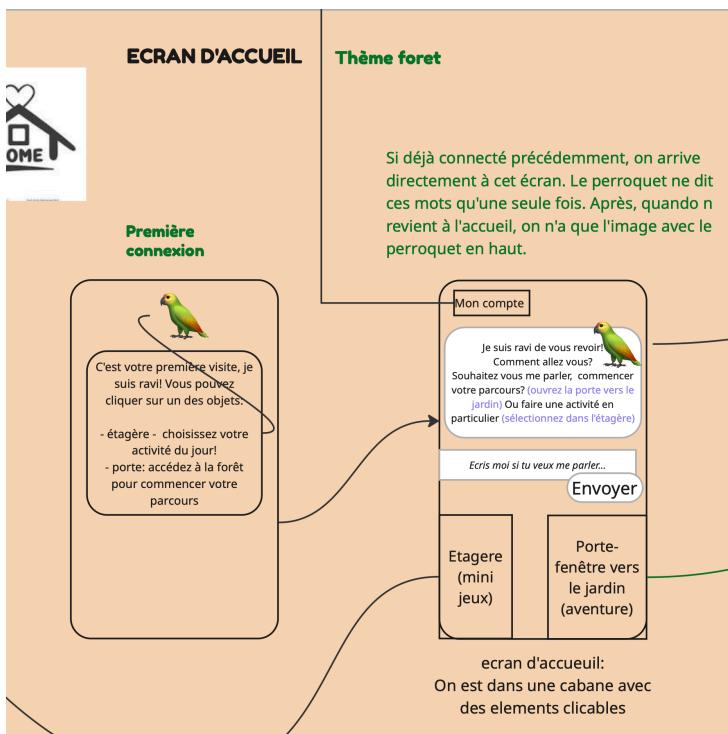
Voici le storyboard entier, on y trouve au centre la page d'accueil et autour les différents écrans vers lesquels il est possible de naviguer:

- L'écran "Mon compte" en vert
- L'écran "Chat" en Jaune
- L'écran "Map" en rose
- L'écran "Leçon" en bleu
- L'écran "Etagère" en gris
- L'écran "Méditation" et "Respiration" en violet

La navigation est représentée par un lien entre 2 écrans.



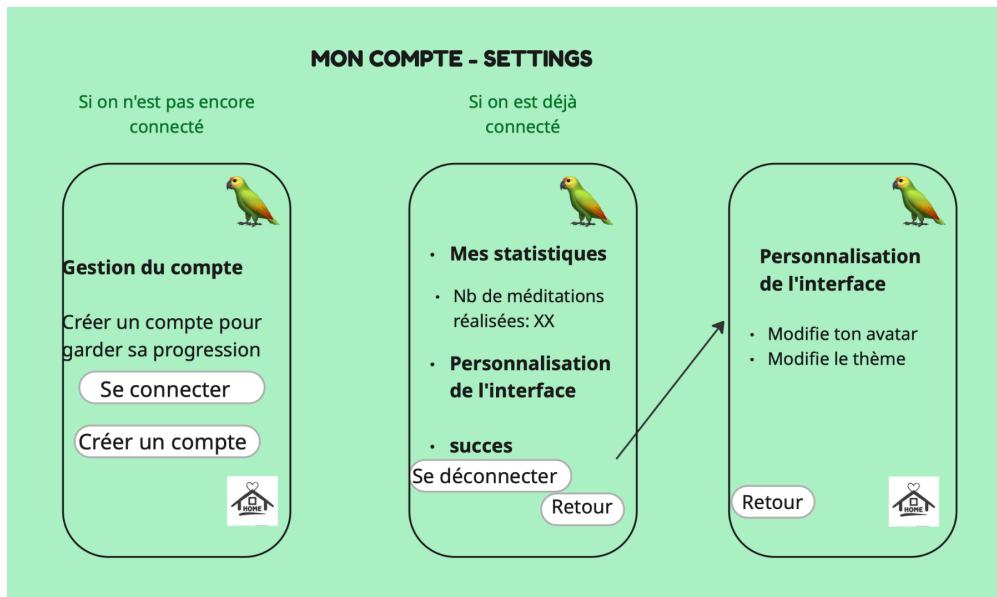
Écran d'accueil:



L'écran d'accueil comprend:

- une porte pour se rendre vers le jardin (Menu map)
- un étagère pour se rendre vers le menu etagere
- un bouton mon compte pour accéder au menu compte
- une image de Coco cliquable pour accéder au menu chat
- une bulle d'information qui explique le menu (et change lorsque l'utilisateur est connecté)

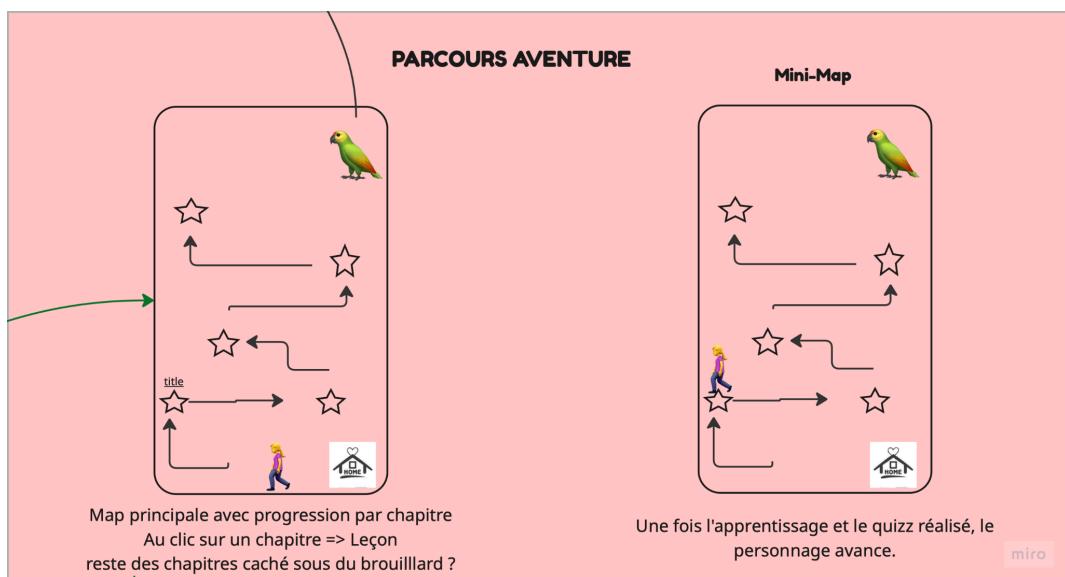
Ecran mon compte:



Permet d'accéder aux paramètres de compte de l'utilisateur pour gérer la création, connection, déconnection, suppression du compte.

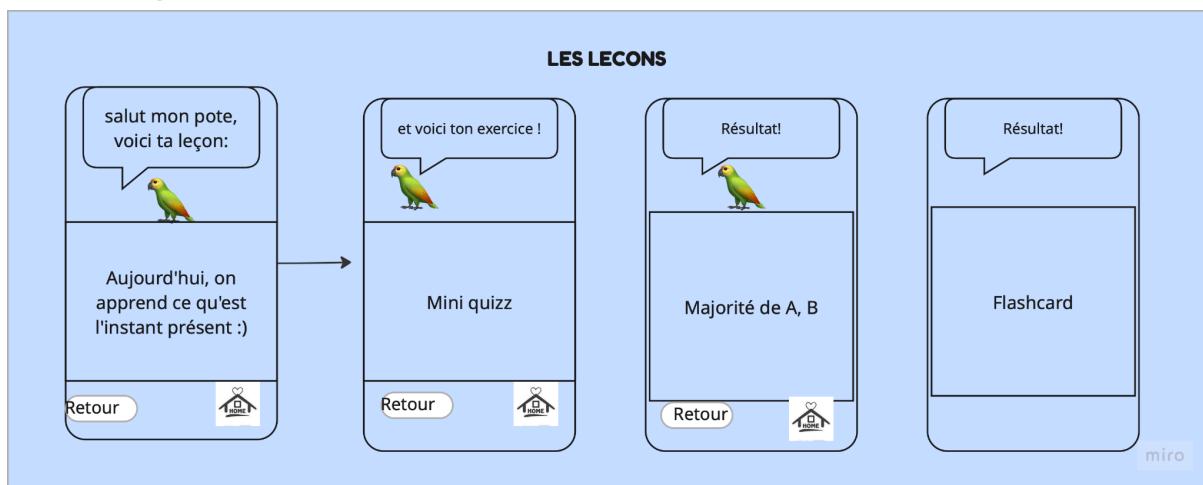
On note des fonctions de statistique de progression ainsi que de personnalisation de l'interface, ces 2 fonctionnalités ont été abandonnées en cours de développement par manque de temps ou déséquilibre avec l'objectif de l'application (la partie statistique a été abandonnée car potentiellement "oppressante" pour l'utilisateur).

Ecran Map:



L'écran de la map a été imaginé avec des points de passages (chapitres) consultables dans l'ordre uniquement (message si l'on tente d'accéder à un chapitre non débloqué).
L'utilisateur clique sur les balise chapitre directement pour accéder au chapitre en question (Menu leçon avec numéro de chapitre en paramètre).
Un bouton retour est présent pour retourner à l'écran d'accueil. Coco est présent en haut de l'écran pour discuter.

Ecran Leçon:



Ce menu se déroule en 3 étapes:

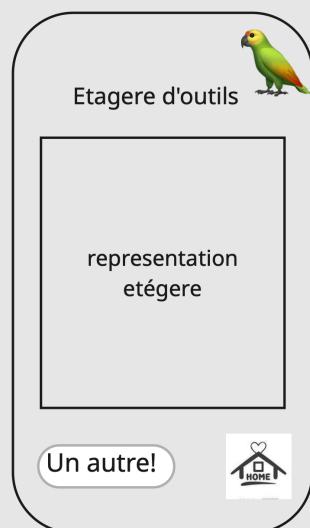
1. Ecran leçon: Affiche une leçon grand format (3mn de lecture max)
2. Ecran quizz: Affiche un quizz en lien avec la leçon, puis le résultat du quizz en fonction des réponses
3. Ecran flashcard: Affiche une flashcard, c'est à dire le contenu de la leçon résumée et facile à retenir

Un bouton retour est présent pour retourner à l'écran d'accueil. Coco est présent en haut de l'écran pour discuter.

Ecran Etagère:

zoom sur étagère avec des objets :

- flashcard : résumé leçons
- objetA: méditation guidée
- objetB: respiration guidée
- objetC: phrases positives
- ObjetD: mini jeux calme



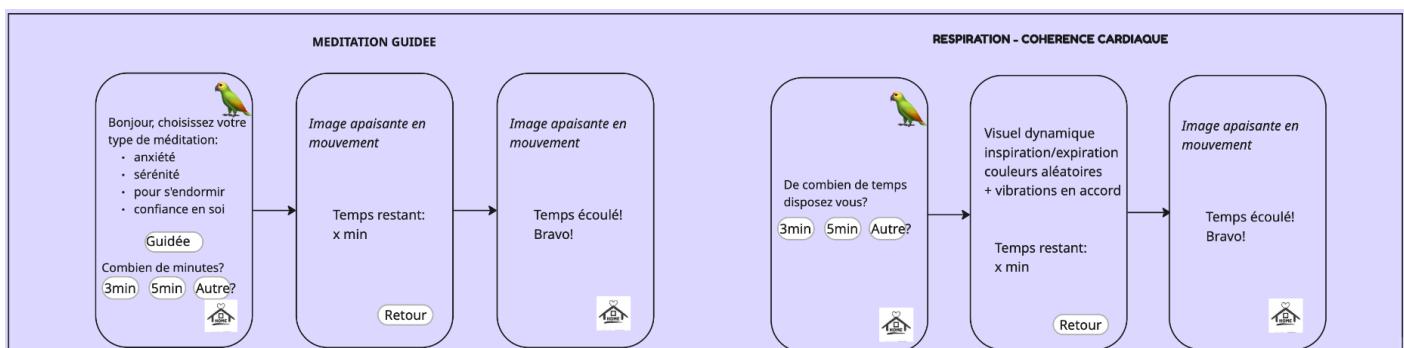
Le menu etagere permet d'accéder à différentes options tel que:

- Les flashcard débloquées via la map
- La méditation guidée
- L'écran de respiration
- Le chat via la présence de Coco

Un bouton retour est présent pour retourner à l'écran d'accueil. Coco est présent sur l'étagère pour discuter.

On observe la présence de "phrases positives" ainsi que de "mini-jeux calmes". Ces 2 options sont aussi des fonctionnalités abandonnées par manque de temps ou incohérence avec le but de l'application.

Ecran Méditation et Respiration:

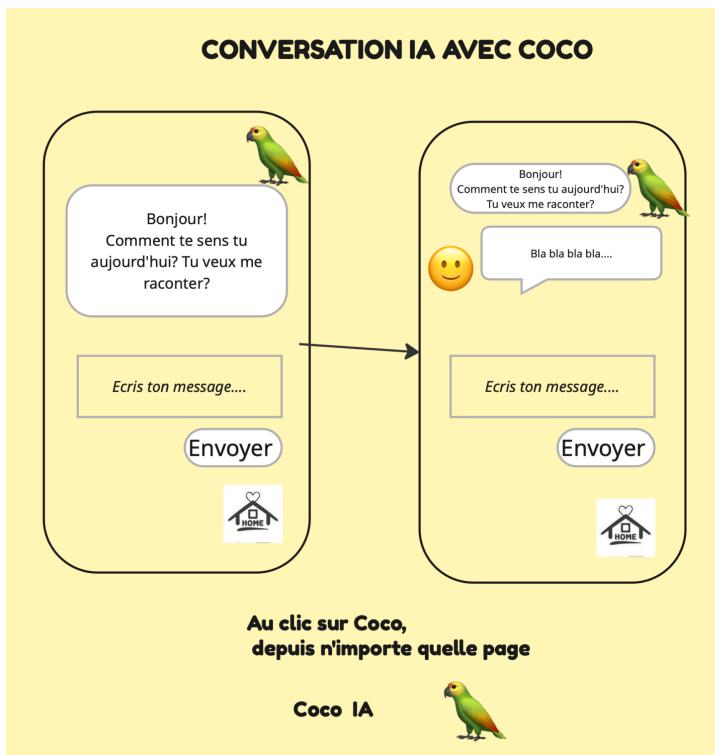


L'écran de méditation guidée comprend des options pour choisir son type de méditation, si la méditation est guidée ou non et sa durée

L'écran de respiration propose simplement une option de durée

Un bouton retour est présent pour retourner à l'écran d'accueil. Coco est présent en haut de l'écran pour discuter.

Ecran Chat IA:



L'écran de chat propose une conversation en temps réel avec une IA.

L'IA commence la conversation en demandant comment l'utilisateur se sent.

Un contexte est fourni à l'IA pour que le comportement de cette dernière soit encadré et respecte les règles que nous lui donneront.

3. Panel utilisateur & User Journey

Après la création du Storyboard Panel Utilisateur et Questionnaires

Après le storyboard, un **panel utilisateur** et un **questionnaire** ont été établis pour valider la pertinence de l'application, affiner l'expérience utilisateur (*User Journey*) et identifier les attentes de la cible (jeunes adultes).

Objectifs:

1. Valider la problématique et la Proposition de Valeur Unique (UVP).
2. Comprendre l'engagement envers la psychologie et les applis de relaxation.
3. Déterminer les parcours utilisateurs probables.
4. Identifier les fonctionnalités essentielles et abandonner les fonctionnalités inutiles.

Méthodologie :

Le panel était composé de profils cibles et profils plus éloignés. Le questionnaire portait sur l'âge, l'utilisation envisagée, la connaissance en psychologie et l'expérience des applis concurrentes.

Résultats:

Les retours ont confirmé l'intérêt pour la méditation/respiration, l'accompagnement sur les différents écrans. Aussi l'importance du renouvellement du contenu et que l'UI soit attrayante. Ces résultats ont validé l'importance de la gamification, de l'utilisation d'une mascotte et justifié l'abandon des statistiques et mini-jeux.

User Journeys (Parcours Utilisateur) :

L'analyse a conduit à définir deux parcours types :

1. **Parcours d'apprentissage** : Parcours centré sur l'apprentissage, l'acquisition et la consolidation des connaissances, soutenu par l'IA (Coco).
 2. **Parcours de relaxation** : Parcours orienté vers le réconfort, discussion avec le chat immédiatement, leçon puis consultation des outils de la bibliothèque
-

Panel utilisateur:

TESTEUR	AGE	CSP	TYPE DE BESOIN	Quel est votre niveau de connaissance et d'engagement sur la psychologie ?	A quelle fréquence pensez-vous utiliser cette application ?
<i>Exemple Cible</i>	15-45 ans	Adolescent, jeunes parents	Personnes en recherche de bien-être	Connaissance basse ou modérée	1 fois à 5 fois par semaine
Cholé Baylet	36 ans	BAC+5. Sans enfants	Ne l'utilisera pas au quotidien, uniquement ponctuellement.	Fait de la méditation et a lu des livres sur le développement personnel. A utilisé des applications similaires telles que Petit Bamboo, 7Mind.	A la maison après la journée. Entre midi et 2
Aubry Gamard	36 ans	Bac +3. Sans enfants	Utilisera l'application ponctuellement, en fonction de la qualité et renouvellement du contenu	A déjà utilisé ce type d'applications, un peu plus d'une dizaine de fois. Pas d'applications similaires à donner en exemple.	Le matin ou soir dans les transports
Jade Kerhervé	22 ans	Bac +3.. Sans enfants	Utilisera potentiellement l'application plusieurs fois par semaine	A déjà lu plusieurs livres sur la confiance en soi et le développement personnel, peut être tentée par ce type d'applications.	A la maison au calme
Pierre Beauchesne	30 ans	Bac. Parent de 2 enfants	Utilisez l'application tous les jours pour les outils de médiation, respiration. Espère que les cours seront fréquemment mis à jour	Présente un grand intérêt pour la psychologie et la compréhension des émotions. Regarde des vidéos sur le sujet, lis des livres.	Quand j'ai du temps dispo
Clovis Battello	36 ans	Bac +5. Sans enfants	N'utilisera sûrement pas l'application mais pourra la conseiller autour de soi	N'est pas touché par ce genre de sujet, ne ressent pas le besoin d'exercices de relaxation. Connais des personnes pour qui l'application pourrait être pertinente	Jamais

User Journey A (retenue):

Aubry Gamard	Main Screen	Map	Flashcard	Méditation	Cohérence cardiaque	Connexion Compte	Chat IA bot
Ressenti global							
Scenario propose	C'est la première fois que tu utilises l'application tu arrives sur cette page d'accueil	Navigation sur la map avec les leçons et idée de gamification	Leçon Quiz via des flashcard ; sélection de plusieurs réponses ; et un résultat attendu	Tu passes maintenant sur l'écran de la méditation	Tu enchaines avec un exercice sur la cohérence	Tu te connectes sur ton compte	Tu dialogue avec l'IA
Commentaire	J'ai deux éléments : étagère et porte qui me permet d'aller dans deux endroits différents ; on suppose qu'il faut cliquer dessus pour naviguer	Apprécie le concept de gamification ; possibilité de pouvoir apprendre des leçons et voir une progress bar	Les flachcards sont bien pensées ; on peut apprendre et se détendre en même temps. On est dans l'obligation de traverser la map pour y accéder ?	L'écran de méditation est bien pensé on peut sélectionner la durée, la possibilité de faire en guidée ou en solo	Je ne comprends trop l'utilité de la cohérence cardiaque ; le principe	Je crée un compte ; qui par la suite me permet de changer de nom ; déconnection	La possibilité de discuter avec le chatbot IA est un plus quand on a besoin d'échanger ; avoir des conseils
Action(s)	Création de deux boutons pour la navigation	Création de composant feux de camp allume ou éteint pour indiquer si un chapitre est terminé ou non	Création d'un screen étagère regroupant plusieurs screens et d'un lien flashcard dans l'étagère	Amélioration de l'écran de méditation	Remplacement par un exercice sur la respiration et introduction du mode vibration	Création de deux boutons pour la navigation	Intégration d'un chatbot IA persistant ; gardant l'historique de la conversation tout en naviguant dans différents screens

(Veuillez excuser la présence des soulignements du correcteur orthographique, il s'agit d'un screenshot direct d'une feuille excel)

User journey B (non retenue):

Pierre Beauchesne	Main Screen	Chat IA bot	Connexion Compte	Map	Leçon	Méditation	Respiration	Modification thème
Ressenti global								
Scenario propose	C'est la première fois que tu utilises l'application tu arrives sur cette page d'accueil	Tu discutes avec le ChatBot	Tu te connecte sur ton compte	Navigation sur la map, clic sur les chapitres	Leçon Quiz via des flashcard; sélection de plusieurs réponses ; et un résultat attendu	Tu passes maintenant sur l'écran de la méditation	Tu enchaine avec un exercice sur la respiration sophrologie	Tu te rends dans le menu mon compte et trouve des options de personnalisation de l'interface
Commentaire	J'ai deux éléments : étagère et porte qui me permet d'aller dans deux endroits différents ; on suppose qu'il faut cliquer dessus pour naviguer mais on "devine"	Le Bot me répond bien, j'engage une conversation et reçois des conseils. Je suis surpris que ça marche sans compte, est ce que ma conversation va rester du coup ?	Je trouve le bouton mon compte sur le mainscreen, création du compte. On est redirigé vers l'écran de connexion au lieu d'être immédiatement connecté, pas super pratique	La map est super cool et simple à comprendre. Ça engage l'utilisateur à poursuivre l'aventure et voir ce qu'il se passe ensuite. Cool de voir les feux de camp s'allumer au fur et à mesure de la progression	Je lis la leçon, RAS. Je passe au quiz, le résultat est intéressant, c'est bien réalisé. Enfin la flashcard, c'est parfait on y retrouve tous les concepts résumés en qq lignes, j'adore.	Méditation simple et efficace, on choisit sa durée, son type de méditation. Peut être ajouter des explications sur le fonctionnement de cette fonctionnalité	J'utilise le menu de respiration, c'est simple et efficace mais je dois regarder l'écran constamment pour suivre ma respiration ? pas pratique j'aimerai fermer les yeux	Option intéressante mais est-ce bien utile ? ça perd l'utilisateur dans le contenu intéressant de l'app. Aussi je ne sais pas ce que je vais modifier sur l'interface, je n'ose pas modifier les paramètres
Action(s)	Ajouter une bulle d'information pour l'utilisateur	Préciser à l'utilisateur que le dialogue n'est pas sauve quand on a pas de compte				Ajouter infobulle explicative	Ajouter le son ou bien vibration	Apparemment pas évident à utiliser... pas vraiment nécessaire non plus. Abandon ?

(Veuillez excuser la présence des soulignements du correcteur orthographique, il s'agit d'un screenshot direct d'une feuille excel)

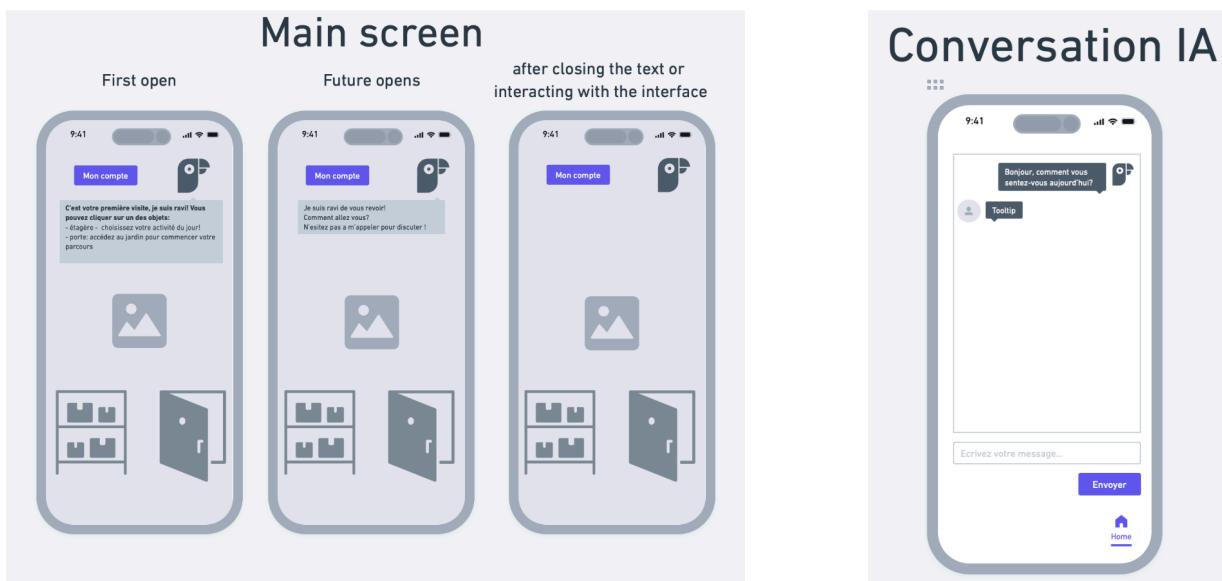
4. WireFrame

Nous avons adopté une approche itérative et centrée sur l'utilisateur pour la conception du wireframe. Le processus a intégré trois sources validées :

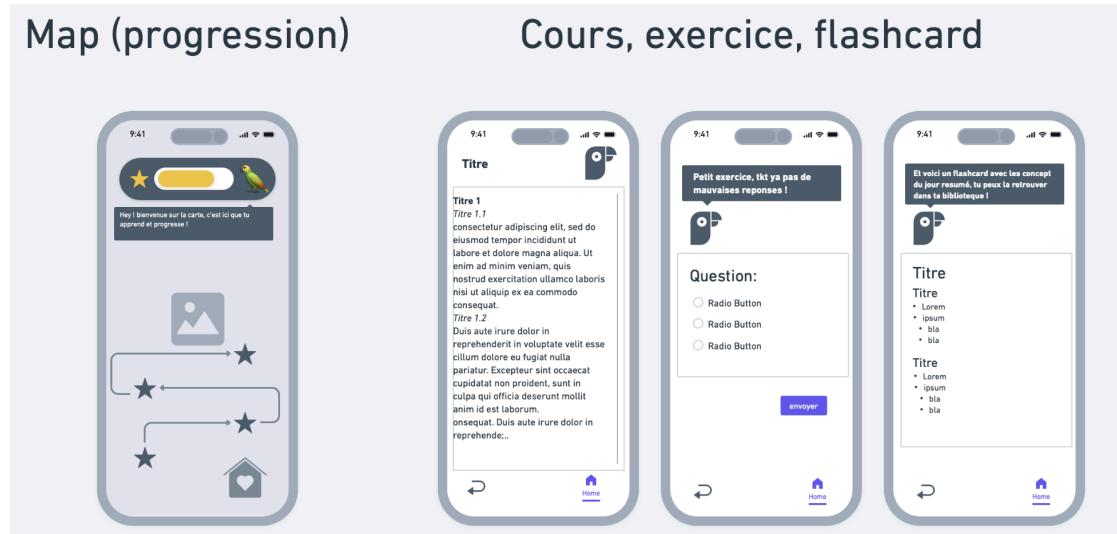
- Le Storyboard** : Utilisé comme feuille de route narrative pour identifier les écrans clés et le flux utilisateur.
- Les Retours du Panel Utilisateur** : Essentiels pour affiner les fonctionnalités, prioriser l'information et répondre aux besoins de l'audience cible.
- Les Users Journey (Parcours Utilisateurs)** : Assurant une navigation fluide et logique en minimisant les frictions et les clics nécessaires.

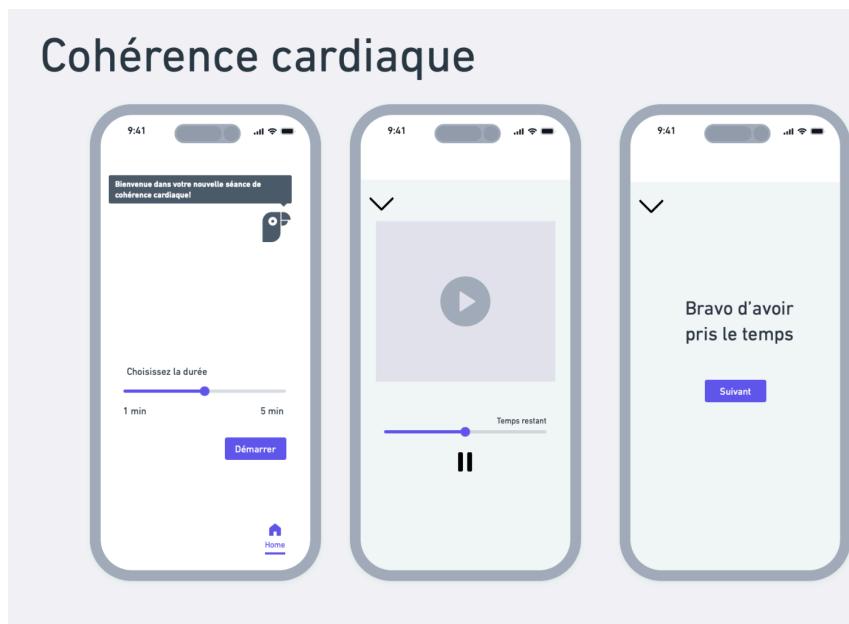
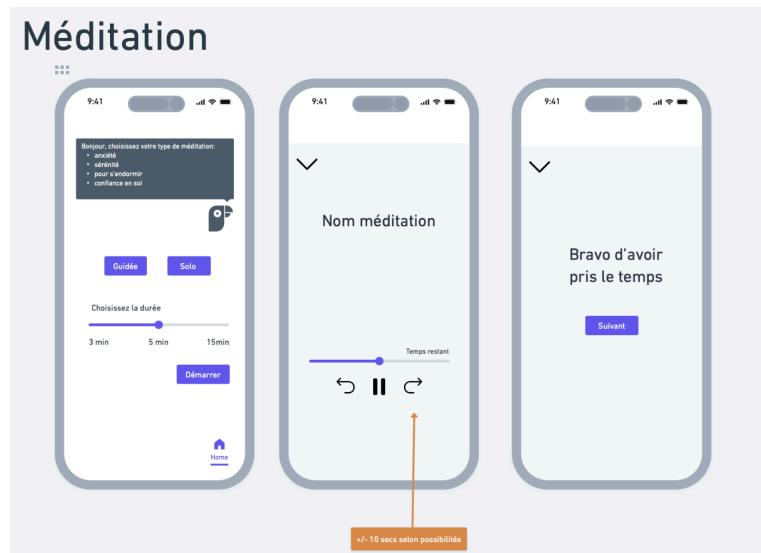
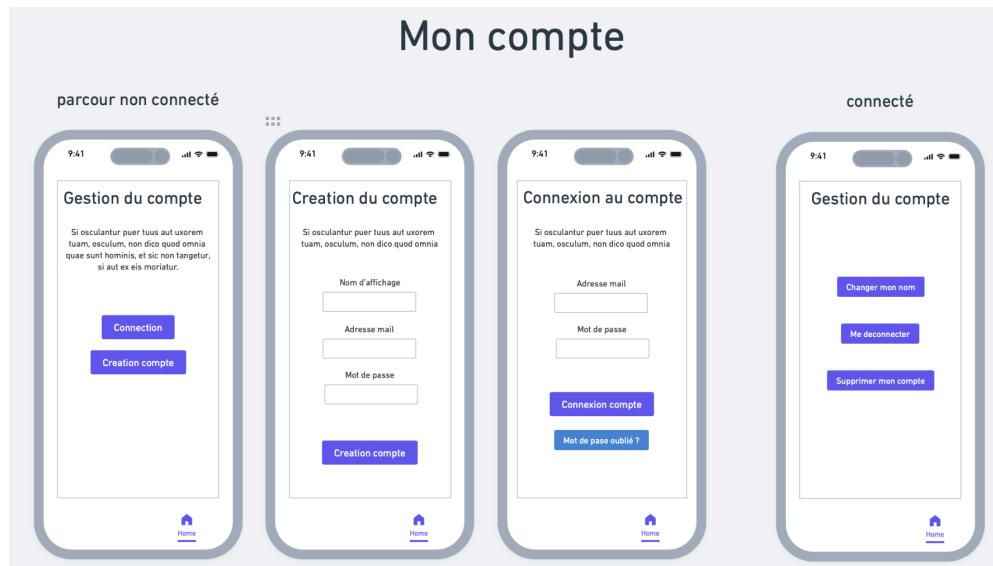
Nous avons utilisé **Whimsical**, une application en ligne collaborative et intuitive, pour créer les maquettes fonctionnelles.

Maquette wireframe:



Map (progression)



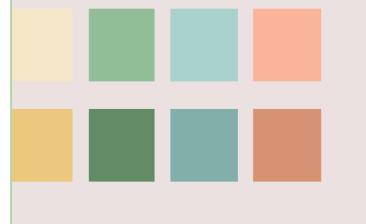


5. UIkit

L'UI Kit a été réalisée sur Figma, il s'agit d'une application en ligne permettant la création de maquettes et prototypes jusqu'à un design définitif. L'UI Kit définit une charte graphique pour l'application et une cohérence visuelle lorsque les développeurs vont travailler sur les différents écrans.

Nous avons décidé d'utiliser des couleurs claires au ton enfantin pour pousser le côté jeux de notre projet tout en donnant un visuel apaisant. Le vert sauge pour l'aspet rassurant et le jaune "or pale" ainsi que le "peche rosé" pour donner du dynamisme.

UI Kit: Palette de couleurs principale & secondaire.

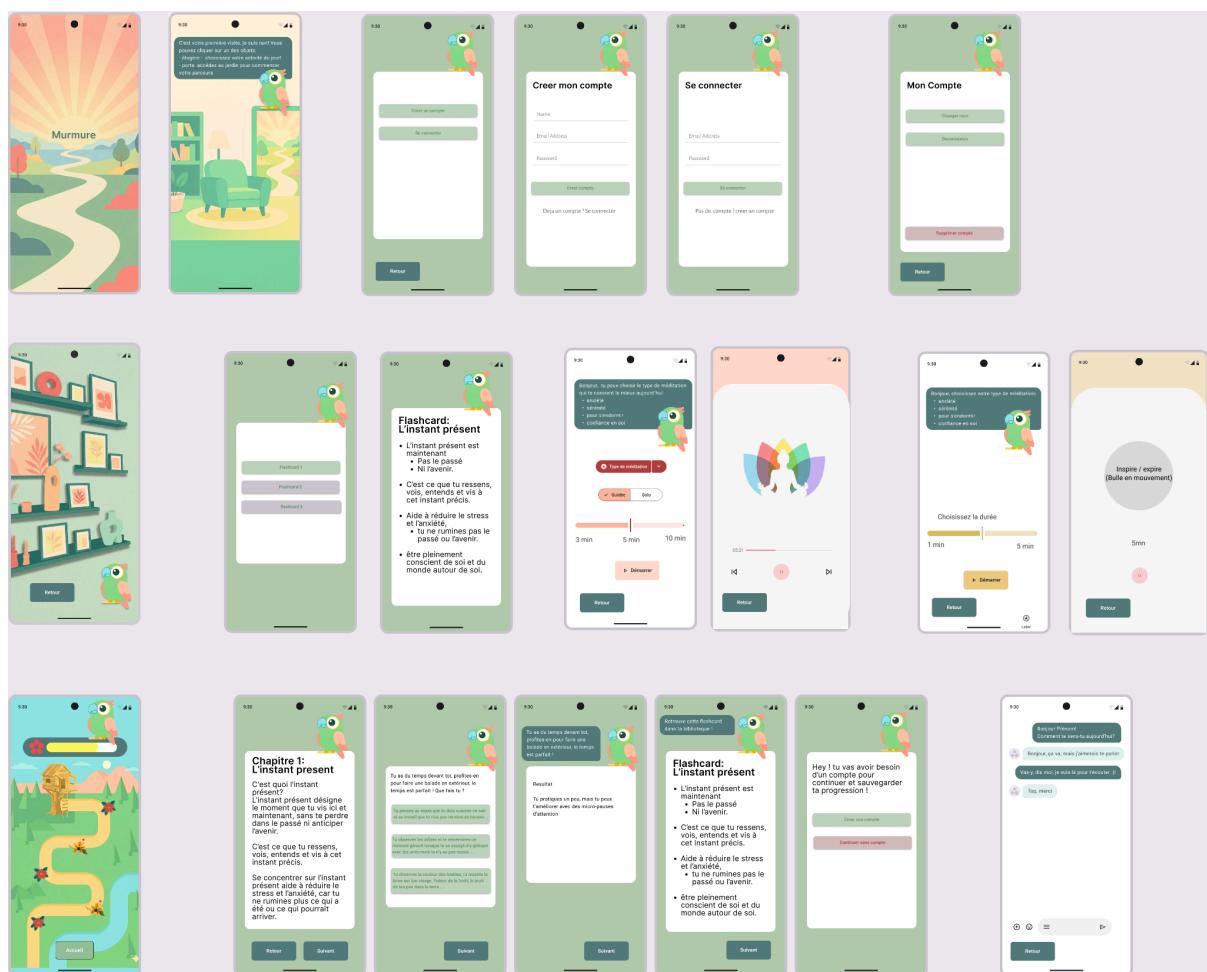
 Or pâle Primary color	 Bleu menthe Secondary color 2	Boutons 																																	
 Vert sauge Primary color	 Pêche rosée Primary color	Navigation 																																	
		Text inputs 																																	
		Google Font: Lexend => <table border="1"> <thead> <tr> <th colspan="2">Font sizes/line heights</th> <th colspan="2"></th> </tr> </thead> <tbody> <tr> <td>38 48</td> <td>28 48</td> <td>Title</td> <td></td> </tr> <tr> <td>22 24</td> <td></td> <td>Subtitle 1</td> <td></td> </tr> <tr> <td>16 24</td> <td></td> <td>Subtitle 2</td> <td></td> </tr> <tr> <td>16 24</td> <td></td> <td>Paragraph 1 (body copy)</td> <td>Input label</td> </tr> <tr> <td>14 16</td> <td></td> <td>Paragraph 1 (bold)</td> <td>BUTTON</td> </tr> <tr> <td>12 16</td> <td></td> <td>Paragraph 2 (small copy)</td> <td>Paragraph 2 light</td> </tr> <tr> <td></td> <td></td> <td>CAPTION</td> <td></td> </tr> </tbody> </table>		Font sizes/line heights				38 48	28 48	Title		22 24		Subtitle 1		16 24		Subtitle 2		16 24		Paragraph 1 (body copy)	Input label	14 16		Paragraph 1 (bold)	BUTTON	12 16		Paragraph 2 (small copy)	Paragraph 2 light			CAPTION	
Font sizes/line heights																																			
38 48	28 48	Title																																	
22 24		Subtitle 1																																	
16 24		Subtitle 2																																	
16 24		Paragraph 1 (body copy)	Input label																																
14 16		Paragraph 1 (bold)	BUTTON																																
12 16		Paragraph 2 (small copy)	Paragraph 2 light																																
		CAPTION																																	

6. Mockups

Le design final de l'application a également été réalisé sur Figma.

La création de la maquette détaillée sur Figma constituait la toute dernière étape du processus de conception, en s'appuyant directement sur toutes les étapes réalisées en amont :

- **User Journey et User Panel** : Les écrans présents sur le mockup font partie de la user journey retenue. Toutes les étapes du parcours sont présentes.
- **Wireframes (Maquettes Filaires)** : Les wireframes ont servi de squelette pour le design de chaque écran. Position des éléments (boutons, images, textes), design des fenêtres et modales.
- **UI Kit (User Interface Kit)** : Nous avons utilisé l'*UI Kit* créé précédemment dans Figma ce qui a permis d'appliquer rapidement et de manière cohérente l'ensemble de la charte graphique (couleurs, typographies, icônes, formes des composants) à tous les écrans. On est ainsi passé d'un *wireframe* fonctionnel à une maquette prête au développement.





Partie 2 : Pilotage du projet d'une application web & mobile

1. Préparation des sprints de développement :

Le pilotage du projet a été structuré autour de la méthode Agile, découpée en quatre sprints de deux jours chacun plus 2 jours de tests et préparation de la présentation pour l'école, pour un total de dix jours ouvrés de travail. Cette approche a apporté plus de flexibilité et de réactivité face aux contraintes de temps et contraintes techniques. En effet ce mode d'organisation permet de prioriser les tâches voire d'en abandonner certaines au fur et à mesure du projet.

Chaque fonctionnalité majeure a été décomposée en **User Stories**, formulées du point de vue de l'utilisateur. Ces *User Stories* ont servi de base à la planification de chaque sprint. Les users stories ont été découpées en tâches puis disposées au sein d'un **Backlog**, le backlog est représenté par une pile de tâches dont l'ordre est définie par la priorité de chaque tâche, les membres de l'équipe vont ensuite prendre ces tâches, se les attribuer lors des réunions de fin de sprint pour constituer leur nouveau sprint.

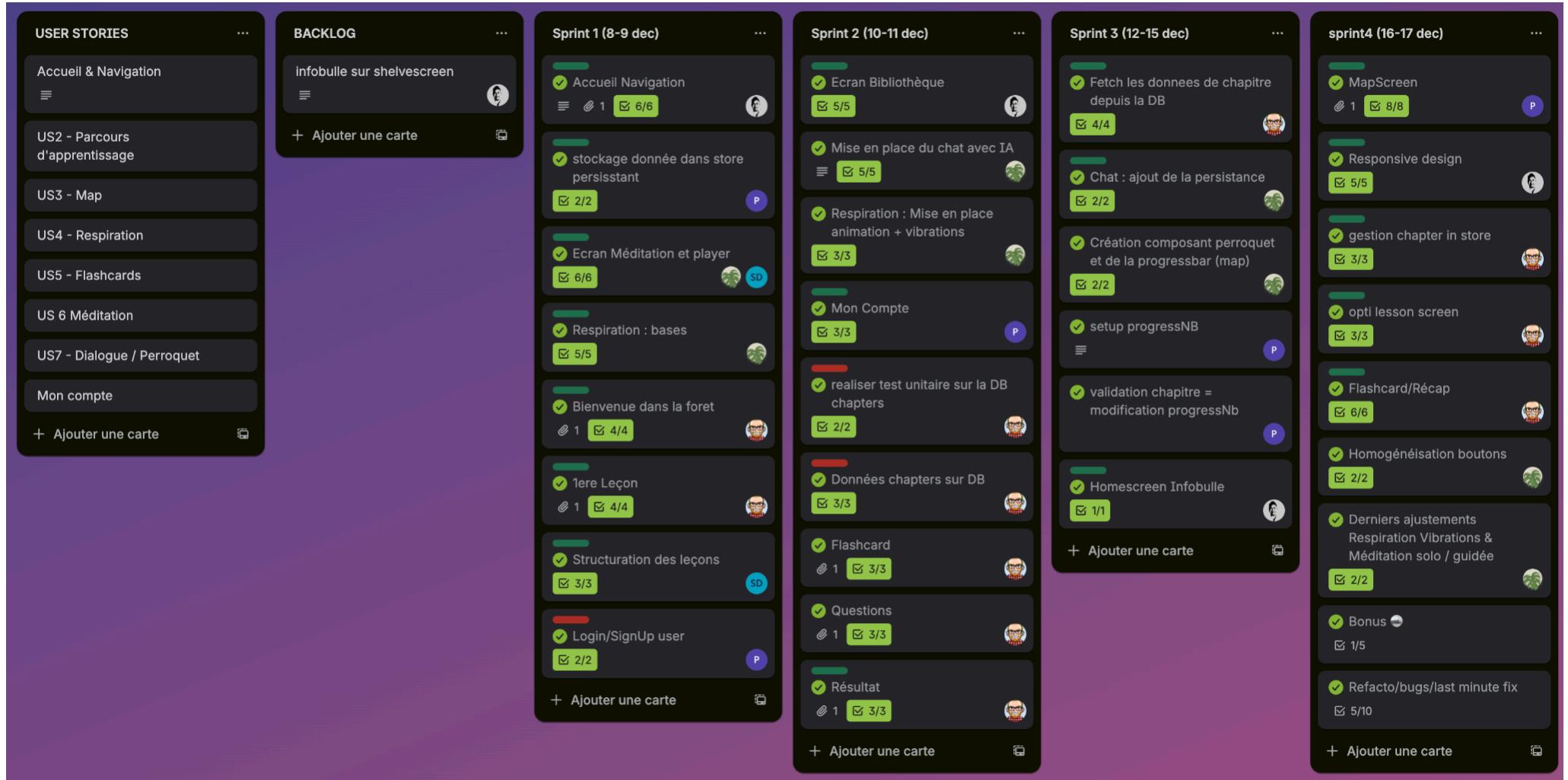
Le **tableau Kanban** sur Trello a matérialisé le flux de travail, donnant une visibilité globale pour les membres de l'équipe sur la progression; permettant l'identification rapide des blocages et une répartition équilibrée des tâches pour chaque sprint. Chaque fin de sprint était conclue par une réunion durant laquelle chaque développeur rapportait le travail réalisé, les problèmes rencontrés et donnait ainsi l'occasion de débattre sur la pertinence ou priorité des tâches non terminées et si cette dernière devrait ainsi être poussée au sprint suivant ou abandonnée.

Vous trouverez, page suivante notre **tableau Trello** et sur les liens suivant les différents documents qui ont servi à l'organisation de nos sprint.

Lien vers le tableau Kanban: [Trello: Organisation sprint](#)

Lien vers la définition des Users Stories: [Notion: User stories](#)

2. Tableau Kanban (Trello):



3. Analyse des compétences nécessaires :

Pour contribuer efficacement au développement de Murmure, en se basant sur les fonctionnalités attendues, les compétences techniques suivantes étaient nécessaire :

Global:

- **Git & GitHub** : Gestion du versioning et travail collaboratif.
- **Dotenv**: gestion des variables d'environnement
- **MVC (Modèle-Vue-Contrôleur)** : Patron d'architecture qui sépare l'application en trois composantes interconnectées pour gérer l'information (Modèle), l'affichage (Vue) et la logique métier (Contrôleur).

Backend:

- **Node.js & Express.js** : Conception de l'architecture serveur backend et d'une API REST pour la gestion des données.
- **MongoDB & Mongoose** : Modélisation des schémas des datas utilisateurs, méditations et chapitres.
- **Socket.io** : Communication bidirectionnelle en temps réel entre notre front et back pour le module de chat.
- **OpenAI & Hugging Face** : Exploitation et configuration de LLM pour le module de chat.
- **Jest & Supertest** : mise en place de tests unitaires et d'intégration.:
- **Render** : Déploiement et mise en production de l'environnement backend.

Frontend:

- **React Native & Expo** : Développement d'une application mobile cross-platform (iOS/Android).
- **React Navigation** : Gestion de la navigation entre les différents modules de l'application.
- **Redux Toolkit & Persist** : Centralisation de l'état global de l'application et mise en place d'un stockage persistant.
- **Socket.io (Client)** : Communication bidirectionnelle en temps réel entre notre front et back pour le module de chat.
- **Expo-AV** : Implémentation du lecteur audio pour les sessions de méditation.
- **Expo-Haptics** : Implémentation de retours vibratoires sensoriels, notamment pour guider les exercices de respiration (inspiration/expiration).
- **Jest & React Testing Library** : Validation de la fiabilité des composants et de la logique applicative via des tests unitaires et d'intégration.
- **EAS (Expo Application Services)** : Déploiement de l'application mobile sur la plateforme EXPO⁵.

⁵ **Expo Dev** (ou **Expo Go**) est le site web ainsi que l'application mobile utilisée pour tester et prévisualiser l'application sur votre téléphone.

4. Déroulé des sprints de développement :

Chaque fin de sprint à fait l'objet d'une réunion de fin de sprint se déroulant de la manière suivante:

- Synthèse avancement (1ere partie de reunion)
 - Réussites: Présentation technique et explication à la team
 - Échecs: Abandon de la feature ? Transfert au sprint suivant ?
- Orga sprint suivant (2nd partie de reunion)
 - Attribution nouvelle tâche
 - Transfer tâche non terminé dans nouvelle fiche
 - création nouvelle fiche type "bugfix" en cas de bug constaté

Les compte rendu etaient directement rédigé sur Notion par le scribe désigné lors de chaque réunion

En plus des réunions de fin de sprint, chaque matin faisait aussi l'objet d'une courte réunion appelée *Morning* donnant l'occasion à chacun de faire part de ses avancements ou blocages et donnant ainsi l'occasion à tout le monde d'être informé d'un changement ayant un impact sur son travail.

Lien vers le compte rendu de nos réunion en ligne: [Notion: CR reunion](#)

Vous trouverez sur les pages suivantes le compte rendu de nos 4 réunions.

5. Compte rendu réunion:

Organisation du sprint n°1 :

Elisa G :

- Réalisation de l'écran meditationHome, et du player. Choix de expo-av. Reste à vérifier le fonctionnement en mode silence du téléphone + vérifications globales.
- Réalisation de l'écran respirationHome et du respirationCountDown. mise en place des animations, reste à revoir l'UI et finaliser les haptiques.

|Write, press 'space' for AI, '/' for commands...

Bat:

- Réalisation du splash screen
- Réalisation du WelcomeScreen + mise en place du préchargement des images en amont
- Réalisation du Homescrreen
 - Mise en place de deux Pulsing bouton pour navigation vers Map ou ShelvesScreen
- Réalisation de Mapscreen

Barthélémy P:

- Ecran de lesson qui peut être appelé avec un numéro d'index de lesson. celui ci comprend:
 - Titre et contenu de la leçon
 - quizz avec question et réponses sous forme de boutons
 - utilisation de route parameter pour servir le numero de chap que lessonSscreen utilisera

Pierre F:

- création du model Users
- création des routes post_users/signin et post_users/signup

Organisation du sprint n°2 :

Elisa G :

- Réalisation du chat, chatScreen et appel à une IA pour réponse.
 - sur front + back (route chat)
 - Server.js (plus moderne)
- Finalisation des screens et features de Respiration et Meditation.
 - vibration en inhale/exhale

Pierre F:

- développement de Mon Compte et de la modale menant à la création de compte
 - accountScreen (création, connexion, delete, edit name)

Bat:

- Réalisation de la ShelvesScreen
 - Mise en place des 3 pulsing pour navigation vers Meditation, Respiration, Flashcard
- Avancer sur le code de la bulle lors de la première connexion

Barthélémy P:

- avancer l'écran de quizz avec compteur de réponses puis résultat
 - fonction MODE pour trouver le plus de réponse A/B/C
- écran de flashcard
- chapterData in DB
 - Pourquoi en DB plutôt que direct hardcodé ? pour pouvoir facilement push de nouveaux chapitres dans le futur (pareil pour les meditations etc)
 - TDD sur les routes
 - quizz sous document
- faire fonctionner la modal pour quitter l'écran de lesson
 - cool trick pour choisir la navigation (function in useState)

Organisation du sprint n°3 :

Pierre:

- ProgressNumber sur mapscreen (lesson terminée update progressNum sur la DB)

Bat:

- relever salon
- 1ere bulle non connecté/2nd bulle connecté (via redux)
- infobulle display name
- List

Stephanie:

- rédaction contenu chapters

Elisa:

- Chat persistant
- Méditation silencieux (vibreurs fonctionne)
- menu méditation plus logique (+ coco explications)
- composant perroquet (coco te salut au début du chat)

Barth:

- fetch chap ⇒ store redux
- changer var affichage chap sur lessonScreen (useState ⇒ useSelector)
- changer schema chapters
- effet fade sur la scrollview de lessonscreen

Organisation du sprint n°4 :

CR test Clovis:

- Bat:

- salut Tim au lieu de connected
- le message d'accueil a disparu vite
- content de te REVOIR? alors que jamais venu/-
- responsive homeScreen
- la bulle cache mon compte/-
- btn compte perdu en haut/-
- safe view pour pas cacher le haut de l'écran / pulse décalé mettre des %/-

- Barth

- chap1/ombrage à la fin du texte (scrollbar)/
- perroquet même couleur que fond/-

- Elisa

- détente 5mn ne dure que 3mn/ solo pas de son pour l'instant/ préciser la diff guidé solo/-
- chat sans historique visible, si on descend le chat tout disparait, il faudrait que ce soit enregistré sur une session entière/-

- Pierre

- création compte marche pas
- chap3 cliquable alors que chap 2 pas fini (dev progression bar + feux de camp allumée / éteint)
- prend la porte (accède au jardin) => aimerais etre + guidé
- placer feux de camp mapscreen
- conversionScreen



Partie 3 : Design de l'architecture BDD et
développement d'une application mobile

Code source du projet:

Voici les liens vers le code source des application Backend et Frontend que nous avons développé:

Murmure Backend: <https://github.com/barthelemy pouset/Murmure-Back>

Murmure Frontend: <https://github.com/barthelemy pouset/Murmure-Front>

N'hésitez pas à vous référer à ces deux repository au fur et à mesure des explications données dans cette partie.

1. Schéma global d'architecture

- **Client (React Native / Expo) :**

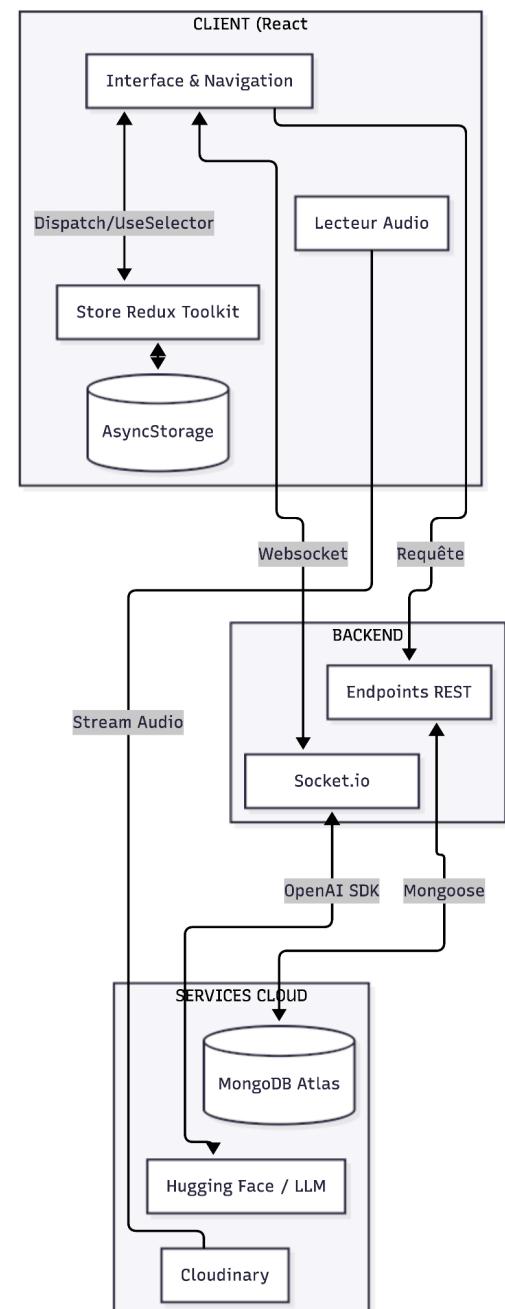
- Des requêtes (fetch) sont réalisées depuis le client vers les routes présente sur le backend pour récupérer les données Users, Chapters, etc.
- Le **Store Redux** enregistre les données récupérées depuis le backend (via Dispatch). **L'AsyncStorage** permet de persister les données du store même après un restart de l'app.
- Lorsque l'utilisateur ouvre le chat, un **websocket** est ouvert entre le client et le backend (cela permet l'échange des message de chat en direct)
- Le lecteur audio **Stream** les fichiers hébergés sur **Cloudinary** grace a la lib **expo-av**.

- **Serveur (Node.js / Express sur Render) :**

- **Express** gère les routes **REST** pour récupérer les utilisateurs, les chapitres et les méditations.
- **Socket.io Server** maintient la connexions actives en mémoire vive pour conserver l'historique du chat (sans passer par la base de données).
- La **Lib OpenAI** fait office de wrapper pour interroger le modèle Google Gemma hébergé sur Hugging Face via une API standardisée.

- **Stockage & Services :**

- **MongoDB Atlas** assure la gestion de l'instance de base de données MongoDB utilisée par notre application.
- **Cloudinary** sert de CDN pour stocker et diffuser les fichiers audio de notre app.
- Hugging Face fournit l'intelligence artificielle grâce au modèle **LLM gemma-2-2b-it**.



2. Identification des librairies et services utilisées

Description des libs utilisées

Backend

- **Node.js**: Environnement d'exécution JavaScript côté serveur
- **Express.js** (4.16.1): Framework web minimaliste pour API REST
- **Mongoose** (9.0.1): ODM (Object Document Mapper) pour MongoDB
- **Bcrypt** (6.0.0): Hachage de mots de passe
- **Uid2** (1.0.0): Génération de tokens uniques
- **Socket.io** (4.8.1): WebSocket pour chat en temps réel avec IA
- **Openai** (6.10.0): SDK pour communiquer avec l'API Hugging Face

Frontend

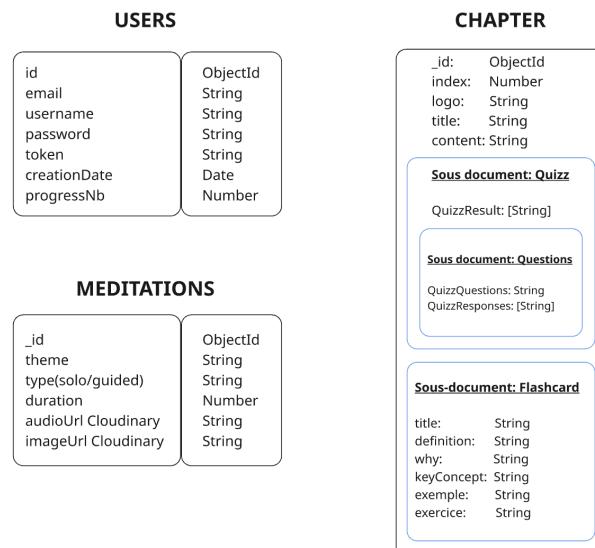
- **React** (19.1.0): Librairie UI déclarative
- **React Native** (0.81.5): Framework mobile cross-platform
- **Expo** (54.0.27): Toolchain pour développement React Native simplifié
- **React-native-safe-area-context** (5.6.0): Gestion des zones sûres (encoche, barre)
- **React-native-gesture-handler** (2.29.1): Gestion des gestes tactiles
- **Reduxjs/toolkit** (2.11.0): Toolkit utilisée pour la manipulation du store redux
- **React-redux** (9.2.0): INteraction React et Redux
- **Redux** (5.0.1): Store global d'état
- **Redux-persist** (6.0.0): Persistance du store (AsyncStorage)
- **Expo-av** (16.0.8): Lecteur audio/vidéo (utilisé pour la lecture des méditations)
- **Expo-haptics** (15.0.8): Librairie utilisée pour les vibration du menu respiration
- **Socket.io-client** (4.8.1): Client WebSocket pour la communication avec le back (utilisée pour le chat)

Webservice

- **Hugging Face** (modèle google/gemma-2-2b-it): Utilisation d'un modèle IA pour la conversation dans le chat
- **Cloudinary**: Service CDN pour l'hébergement des médiations
- **MongoDB Atlas**: Solution Cloud pour le provisionnement d'Instance mongoDB

3. Schéma de base de données

Le choix de la base de données **MongoDB** a été fait pour sa flexibilité et sa scalabilité, parfait pour un produit en phase de développement. L'utilisation de **Mongoose** facilite la modélisation des données côté backend grâce à l'utilisation de schémas.



La base de données est structurée autour des collections suivantes :

Collection	Rôle	Justification des choix de modélisation
Users	Stocke les informations des utilisateurs (authentification, progression, etc.).	La collection user n'a pas besoin de faire un lien direct avec la collection chapter car on stocke seulement le numéro de chapitre sur lequel l'utilisateur est rendu (et non le chapitre entier).
Chapters	La collection chapter possède le contenu des leçons ainsi que plusieurs sous-collections : Quizz contient les questions (aussi sous forme de sous-document) et le résultat à afficher par chapitre. Flashcard possédant le contenu de chaque FC par chapitre.	Le choix d'utiliser des sous-collections pour les quizz, questions et flashcards s'est montré pertinent car ces objets appartiennent directement à la logique des chapitres et ne seront jamais partagées avec d'autres tables.
Meditations	Stocke les détails des sessions de méditation (titre, durée, URL audio).	Modélisation simple, chaque document contient l'URL du fichier audio pour lecture via Expo-AV côté client.

L'architecture en sous-documents favorise la rapidité d'accès aux données sollicitées ensemble tel que le contenu d'un chapitre.

Justification du modèle de données:

Le choix de la construction de nos schémas à commencer par définir ce qui était nécessaire à stocker en BDD:

- Users:

Une collection pour les utilisateurs est obligatoire car nous reposant sur un principe de compte avec identifiant et progression individuelle.

- Chapters:

Concernant la partie chapitre nous nous sommes demandés s'il n'était pas intéressant d'avoir les données de chapitre directement stocké sur le front (pour économiser en charge serveur et en bande passante notamment) mais cela poserait la contrainte d'avoir à builder et publier une nouvelle application à chaque ajout de chapitre.

La solution de stocker les chapitres en BDD a donc été conservée pour sa flexibilité quant aux mises à jour de contenu futur.

- Méditations:

La même réflexion a été posée pour les méditations (Stocké sur l'app ou en BDD ?) et pour la même raison de simplicité de mise à jour le choix de les stocker en base de données a été privilégié.

Ensuite, nous nous sommes posé la question des dépendances entre les collections et plus particulièrement celle entre la progression de l'utilisateur (users -> progressNb) et les différents chapitres.

Était-il nécessaire de lier la progression de l'utilisateur via une clé étrangère pointant vers le chapitre sur lequel l'utilisateur était rendu ?

Nous avons décidé de faire plus simple en créant une variable de type number pour la progression, ainsi qu'ajouter un numéro d'index sur chaque chapitre permettant ainsi de pouvoir pointer un chapitre (ou plusieurs) au niveau du frontend, ce qui a considérablement simplifié notre schéma.

Enfin la partie qui a demandé le plus de réflexion a concerné les collections flashcard et quiz: Ces collections doivent être reliées à la collection chapters via des **clefs étrangères** ou via des **sous documents** ?

Après nos recherches nous avons conclu que ces collections seraient toujours appelées via la collection chapters. En effet au lancement de l'application frontend un unique Fetch est réalisé pour récupérer tout le contenu des chapitres directement. En faisant une sous collection, MongoDB n'a pas à gérer de lien avec d'autres collections et récupère ainsi un Objet unique contenant collection principale et sous collection directement.

Gestion du chat IA:

Il est à noter que la gestion du chat est réalisée de manière volatile, sans persistance en base de données, en s'appuyant sur l'objet en mémoire **conversationsByToken** au sein du backend directement (server.js).

En somme, la conversation est stockée en mémoire vive du serveur, par conséquent un redémarrage du serveur entraîne la perte de l'historique de conversation. Ce comportement peut être fixé lors d'une itération de travail supplémentaire intégrant la sauvegarde du chat en BDD.

Les échanges en temps réel sont orchestrés par la librairie Socket.io, qui utilise le token unique de l'utilisateur comme clé pour stocker et retrouver son historique de messages durant la session.

À chaque nouveau message, le serveur met à jour cet historique local avant de solliciter l'IA via l'API de Hugging Face, garantissant ainsi que le modèle dispose du contexte nécessaire pour répondre sans solliciter MongoDB.

Script d'import de données:

Un autre élément important durant le développement de notre app a été de pouvoir travailler sur les mêmes données tout en ayant son propre environnement de développement.

Le script seed.js (présent au sein du back dans le répertoire script/) assure l'initialisation automatisée de la base de données MongoDB en procédant à l'importation d'un jeu de données de référence (stockés dans le répertoire data/).

Lors de son exécution, le script réinitialise les collections existantes avant d'y injecter les nouvelles données, tout en s'assurant que ces données correspondent aux schémas que nous avons définis.

4. Routes et endpoints backend :

Les routes définies reposent essentiellement sur la structure de notre base de données, les différents appels vers le Back entraînent des opérations type CRUD sur la BDD permettant ainsi de créer (Create), lire (Read), mettre à jour (Update) et supprimer (Delete) les informations.

Route users:

Prefix	Endpoint	Type	Rôle / Description	Paramètres	Informations renvoyées
/users	/signup	POST	Inscription d'un nouvel utilisateur.	username, email, password	result, token, username, progressNb
/users	/signin	POST	Connexion d'un utilisateur existant.	email, password	result, token, username, progressNb
/users	/updateUsername	PUT	Modification du nom d'utilisateur.	token, newUsername	result, username
/users	/progress	PUT	Mise à jour du niveau de progression.	token, progressNb	result, progressNb
/users	/deleteUser	DELETE	Suppression définitive du compte.	token	result, message

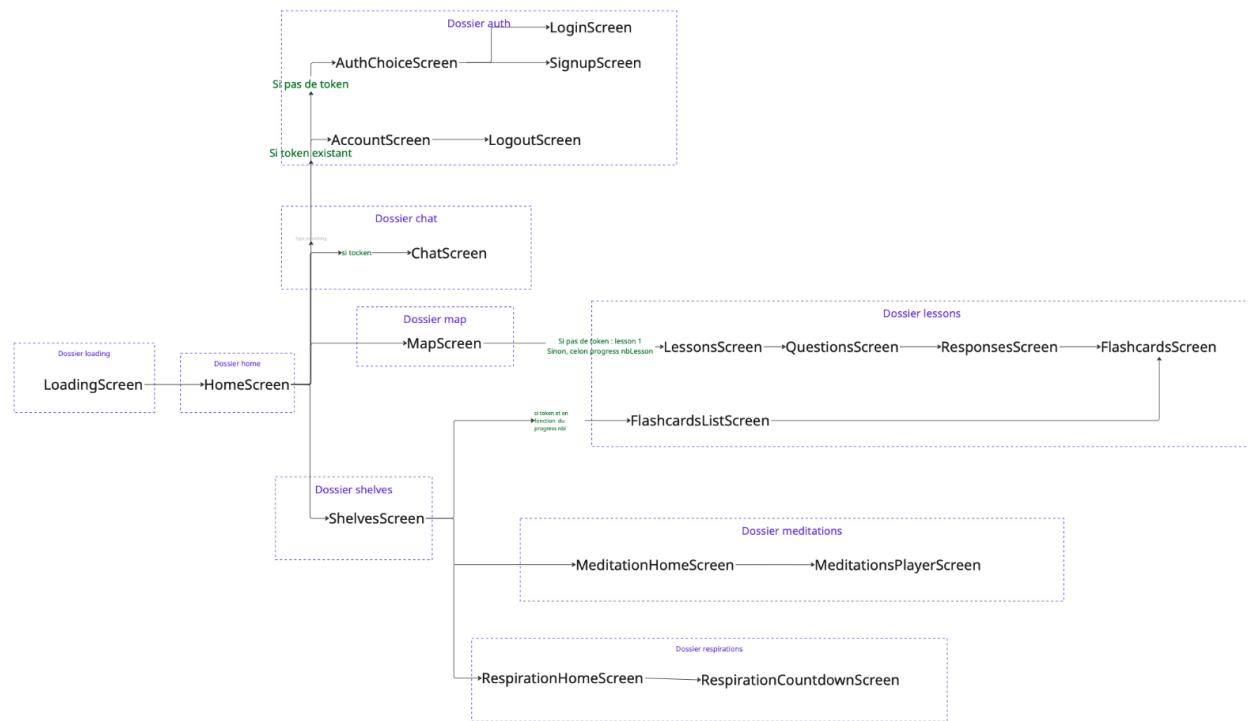
Route chapters:

Prefix	Endpoint	Type	Rôle / Description	Paramètres	Informations renvoyées
/chapters	/	GET	Récupération de tous les chapitres.	/	result, chapters
/chapters	/:id	GET	Récupération d'un chapitre spécifique.	:id	result, chapters

Route meditations:

Prefix	Endpoint	Type	Rôle / Description	Paramètres	Informations renvoyées
/meditation	/player	POST	Récupération de l'URL audio d'une méditation.	theme, mode, duration	result, audioUrl
/meditation	/	POST	Ajout d'une ressource (Admin).	theme, mode, duration, audioUrl, imageUrl	allMeditations

5. Schéma des composants React & React Native



L'architecture de cette application mobile React Native de type Single Page Application (SPA) repose sur une séparation des responsabilités (Separation of concerns), organisée principalement au sein du répertoire screens. Ce répertoire regroupe toutes les interfaces utilisateur (écrans) de l'application, dont voici la structure:

Accueil de l'utilisateur:

- **LoadingScreen.js** : “Splash Screen” affichant une image lors du chargement initial de l'app.
- **HomeScreen.js** : Écran d'accueil, servant de point de navigation principal vers la map, la bibliothèque et le menu compte.

Gestion du Compte Utilisateur:

- **SignUpScreen.js** : Permet la création d'un nouveau profil utilisateur.
- **SignInScreen.js** : Interface dédiée à la connexion des utilisateurs existants.
- **CompteScreen.js** : Affiche et permet la gestion du profil de l'utilisateur connecté.

Fonctionnalités d'apprentissage:

- **Carte**
 - **MapScreen.js** : Présente une carte visuelle pour le suivi de la progression de l'utilisateur à travers les différents modules ou chapitres d'apprentissage.
- **Leçons** (Contenu éducatif)
 - **LessonScreen.js** : Affiche le contenu spécifique d'une leçon.
 - **FlashcardScreen.js** : Propose un système de fiches (flashcards) pour faciliter la mémorisation et l'apprentissage.

Ecran étagère et outils de bien-être:

- **Etagères**
 - **ShelvesScreen.js** : Représente l'etagere où l'utilisateur peut retrouver et utiliser les différents outils dédiés au bien être.
- **Méditations** (Séances de méditation guidée)
 - **MeditationHomeScreen.js** : Hub pour la sélection des différentes sessions de méditation.
 - **MeditationPlayerScreen.js** : Lecteur dédié au lancement et au guidage d'une séance de méditation.
- **Respirations** (Exercices de respiration guidée)
 - **RespirationHomeScreen.js** : Permet de choisir parmi divers exercices de respiration.
 - **RespirationCountdownScreen.js** : Guide visuellement l'utilisateur pendant l'exercice, via un compte à rebours ainsi que via des vibrations de l'appareil.
- **Chat** (Discussion avec un IA)
 - **ChatScreen.js** : Fournit une interface de chat, avec assistant virtuel type IA.

La gestion de la navigation entre ces multiples écrans est centralisée par le composant **navigation/AppNavigator.js**, qui agit comme le routeur principal de l'application.

6. Navigation et interface graphique (UX/UI):

Notre Frontend est construit avec le framework Expo, qui permet de simplifier le développement pour iOS et Android. Les principaux éléments de fonctionnement de notre interface sont les suivants :

- **Navigation :**
 - **react-navigation/native** et **react-navigation/native-stack** : React Navigation sert à gérer la navigation entre les différents écrans de l'application. C'est la solution de navigation standard dans l'écosystème React Native.
- **Gestion d'état (State Management) :**
 - **reduxjs/toolkit** et **react-redux** : L'état global de l'application (comme les informations de l'utilisateur ou la progression dans les chapitres) est géré par Redux.
 - **redux-persist** : Cette librairie est utilisée pour sauvegarder l'état Redux dans le stockage local de l'appareil, ce qui permet de conserver les données entre les sessions.
- **Fonctionnalités Spécifiques :**
 - **expo-av** : Gère la lecture de contenus audio et vidéo, utilisés pour les écrans de méditation ou de leçons.
 - **socket.io-client** : Utilisé pour la communication en temps réel vers le back, nécessaire au chat.
- **Composants réutilisables :**

Nous avons aussi créé différents composants propres à notre application (sous le dossier **/components**). Ces éléments sont les briques de base de notre UI et reprennent les code de notre Ulkit :

 - **Button.js** et **ChapterButton.js** : Des boutons personnalisés, avec une version spécifique pour la navigation entre les chapitres.
 - **ConfirmModal.js** : Une fenêtre modale pour demander une confirmation à l'utilisateur.
 - **DurationSelector.js** : Un sélecteur de durée, qui utilise probablement le composant `@react-native-community/slider`.
 - **Fire.js** : Un composant visuel, qui est utilisé pour afficher les checkpoint sur la map sous forme de feu de camp, allumé ou éteint (via 2 images).
 - **InfoBulleHome.js** : Une bulle d'information pour guider l'utilisateur.
 - **ParrotChatBtn.js** : Un bouton spécifique pour lancer la discussion avec le perroquet.

Gestion de différents formats d'écran

Pour assurer un design adaptatif et une expérience utilisateur cohérente sur une multitude d'appareils, notre approche combine plusieurs outils clés de React Native et une solution sur mesure.

La base de notre mise en page repose sur **ImageBackground en cover**, qui garantit que l'image de fond remplit toujours l'écran sans être déformée. Par-dessus, nous utilisons deux techniques principales :

- **Gestion des zones de sécurité (Notch, barre de statut)** : Nous utilisons le hook **useSafeAreaInsets** de react-native-safe-area-context. La valeur **insets.top** est récupérée et intégrée dans une formule pour ajuster dynamiquement la marge supérieure du conteneur principal. Cela pousse le contenu vers le bas juste ce qu'il faut pour éviter l'encoche, tout en gardant une position cohérente sur les appareils sans encoche.
- **Positionnement proportionnel des éléments** : Pour placer les éléments interactifs (comme les boutons flottants) sur l'image, nous avons créé le hook custom **useResponsiveImagePosition**. Ce hook compare les dimensions de l'image d'origine à celles de l'écran. Il calcule un ratio d'échelle (**scale**) et des décalages (**xOffset**, **yOffset**), puis fournit une fonction **getPos(x, y)**. Cette fonction traduit les coordonnées d'un point sur l'image d'origine en une position 'absolute' à l'écran. Le 'scale' est aussi utilisé pour redimensionner les composants eux-mêmes, assurant que tout reste proportionnel.

7. Approche TDD:

Pour certains développements spécifiques nous avons suivi une approche TDD (test driven development), il s'agit d'une approche où les tests unitaires sont écrits avant le code de production, servant de spécification et garantissant que la fonctionnalité nécessaire est implémentée en accord avec la définition de la user story.

Ainsi, la personne responsable du développement de cette fonctionnalité devra développer sa partie tout en garantissant le succès des tests unitaires permettant une couverture des fonctionnalités et un respect des spécifications attendues.

j'ai pour ma part réalisé le développement de la route chapters, j'ai donc réalisée en amont les tests concernant ce développement.

Le fichier **tests/chapter.test.js** regroupe des tests d'intégration (développés avec Jest) qui valident à la fois les routes liées aux chapitres et la structure des données renvoyées par ces routes. Ces tests couvrent la récupération de l'ensemble des chapitres, l'accès à un chapitre précis via son numéro d'index, et la gestion des cas d'erreur potentiels tel qu'un chapitre inexistant, un pb de communication avec le serveur et code réponse attendu (**404**: chapitre non trouvée; **400**: id de chapitre invalide).

8. Authentification:

Le processus d'authentification des utilisateurs est géré de la manière suivante au niveau du backend :

1. **Validation des entrées** : Avant tout traitement, le serveur vérifie que les champs requis (comme email, password, etc.) sont bien présents dans la requête grâce au module `checkBody`. Cela prévient les erreurs dues à des données manquantes.
2. **Hachage du mot de passe** : Lors de la création d'un compte (/signup), le mot de passe n'est jamais stocké en clair dans la base de données. Il est transformé en une chaîne de caractères sécurisée (un "hash") à l'aide de la librairie `bcrypt`.
3. **Vérification sécurisée du mot de passe** : Au moment de la connexion (/signin), le mot de passe fourni par l'utilisateur est comparé au hash stocké en base de données. La fonction `bcrypt.compareSync(...)` effectue cette comparaison de manière sécurisée, sans jamais avoir à déchiffrer le mot de passe original.
4. **Authentification par token** : Une fois l'utilisateur inscrit ou connecté avec succès, le serveur lui attribue un token unique (généré par la librairie `uid2`). Ce token sert de "laissez-passer" pour les requêtes suivantes. Au lieu de renvoyer son email et mot de passe à chaque fois, l'application cliente envoie simplement ce token pour prouver l'identité de l'utilisateur.

En résumé, la sécurité repose sur le non-stockage des mots de passe en clair, l'utilisation d'un algorithme de hachage robuste, et un système de session basé sur des tokens pour les utilisateurs authentifiés.



Partie 4 : Mise en production du projet

1. Schéma de l'environnement de déploiement

Voici l'approche utilisée pour la "semi" mise en production de notre application. Semi, car la partie frontend repose sur le système de preview intégré au framework de développement mobile Expo. Ce système permet de simuler une application en ligne sans avoir à publier celle-ci sur un store. Idéal donc pour tester et présenter un POC fonctionnel de notre application.

Backend:

Pour le déploiement du back, le site Render.com est utilisé, il s'agit d'un site qui propose des instances serverless pour vos applications. Render supporte plusieurs langages et frameworks de développement.

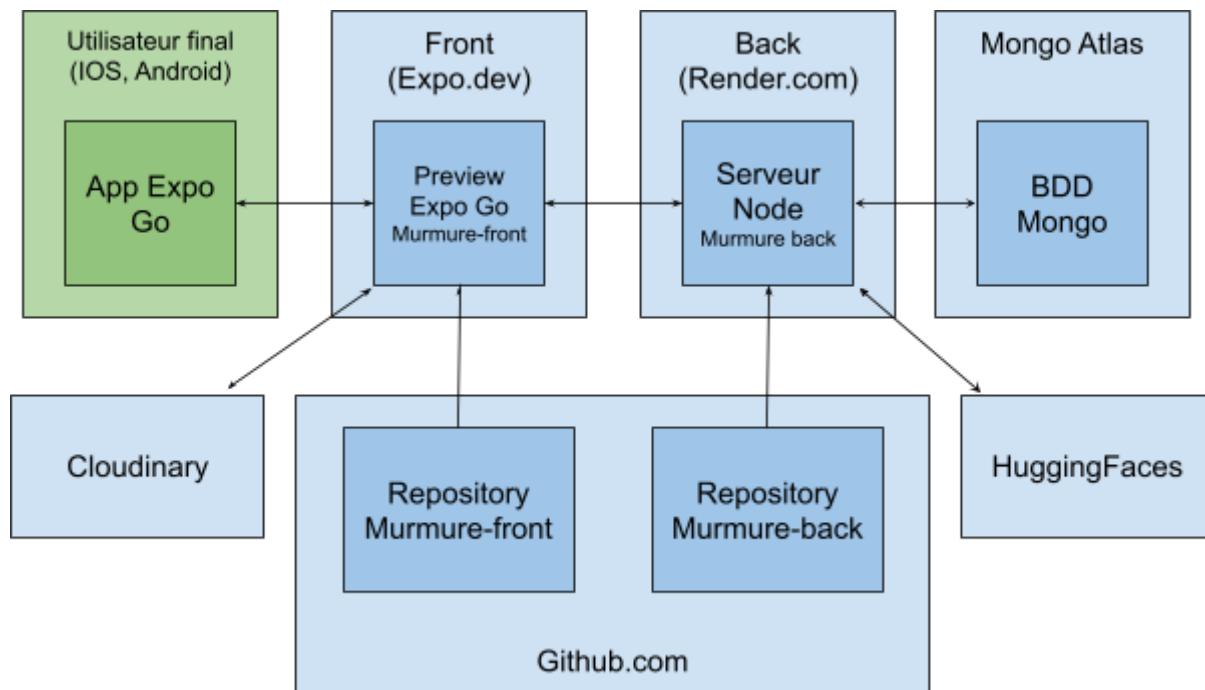
Nous avons décidé d'utiliser **Render** pour le déploiement de notre back car il est l'un des seuls acteur gratuit du marché à supporter les **WebSocket** nécessaire à notre chat.

Render assure le déploiement du backend via une intégration **GitHub**. Render récupère le contenu du repository back, *build* l'application Node.js/Express, et met à jour le serveur en production. Cela permet un déploiement rapide et simplifié.

Frontend

Le déploiement du front-end utilise **EAS (Expo Application Services)** et **Expo Go**.

EAS permet de *builder* l'application et de la rendre testable instantanément via **QR Code** sur iOS/Android, simulant la production sans passer par les stores. Pour mettre à jour notre front, il est possible de relancer un déploiement avec l'outil AES-CLI depuis notre terminal directement.



Pour tester le projet:

Voici le lien vers la partie backend de l'application en production sur Render.com:
<https://murmure-back.onrender.com>

Si votre téléphone est muni de l'application EXPO GO (à télécharger sur le store), vous pouvez flasher ce QR code et ainsi tester l'application:



Conclusion

Ce projet est le fruit de 2 semaines d'organisation, de réflexion et de débat avec mon équipe. La constitution de cette dernière était très intéressante car chacun avait sa spécialité et l'envie de travailler sur une partie précise de l'application. La répartition du travail s'est vraiment faite de manière naturelle.

Ayant apporté cette idée à ma classe j'avais une vision de ce à quoi je voulais que le projet ressemble, ce qui serait pertinent pour l'utilisateur final en termes de contenu, et de présentation. Lors de nos premières réunions organisationnelles nous avons longuement débattu sur la manière dont nous souhaitions rendre l'application pertinente au niveau du contenu sans la rendre ennuyante. Et aujourd'hui je suis très fier de pouvoir présenter l'application que nous avons créé.

Tout le déroulement du développement s'est lui aussi passé sans soucis, ce qui est vraiment remarquable quand j'entend le retour des autres équipes de travail pour qui tout ne s'est pas passé aussi simplement. J'ai eu la chance de réunir une équipe de personnes réellement motivée par le sujet de la psychologie, par l'envie d'apprendre et donner de leur énergie pour ce projet.

En organisant correctement nos sprint, en échangeant de manière ouverte tout au long du développement nous avons réussi à créer un projet dont nous sommes tous très fiers et qui répond vraiment à l'ambition que nous avions au démarrage du projet.

Je tiens à remercier sincèrement mes camarades pour l'investissement qu'ils ont tous donné. Je n'hésiterai pas une seconde à retravailler avec vous.

Merci Elisa, merci Stéphanie, merci Bat, merci Pierre.

Après ces trois mois de formation, je me sens prêt à proposer mes compétences et travailler dans une entreprise pour participer au développement de leur produit. J'avais le souhait de devenir développeur spécialisé en Mobile et cette formation a confirmé mon désir.

Je remercie aussi la capsule pour la qualité de la formation dispensée.