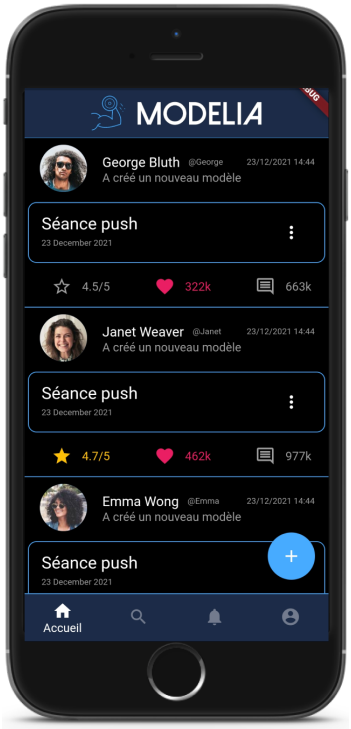
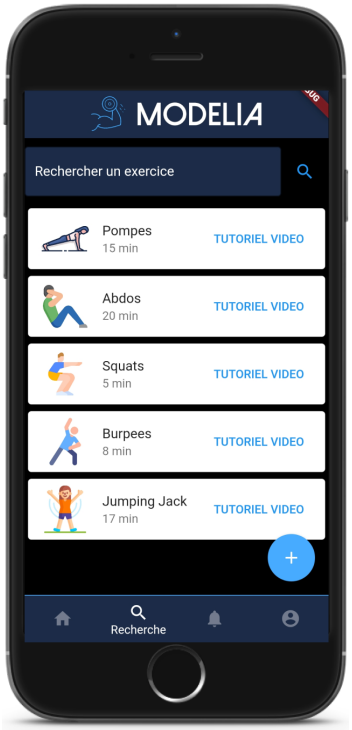
DOCUMENT EXPLICATIF

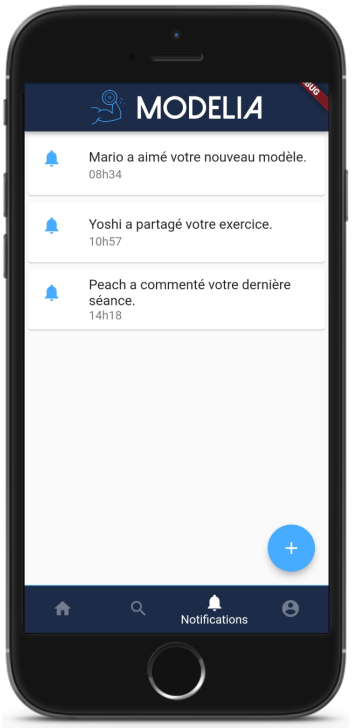
TECHNOLOGIES MOBILES

Mickael Gomez

L'application **MODELIA** développée en React Native dans le cadre du projet platine a été en partie recréée en Flutter. Le nouveau projet reprend 3 sections principales :







FIL D'ACTUALITÉ

Le fil d'actualité dans l'accueil de l'application présente une liste de modèles et séances postés par d'autres utilisateurs. Chaque post peut être enregistré en favoris ou recevoir un like, et un menu contextuel liste un ensemble de fonctionnalités supplémentaires disponibles dans l'application originale (enregistrer le modèle, créer un nouveau modèle à partir de celui sélectionné, etc...)

RECHERCHE D'EXERCICES

Une liste d'exercices pré-enregistrés dans l'application (200+ dans l'originale) s'affiche à l'écran. Pour chaque exercice, un lien vers une vidéo de démo est disponible, ainsi qu'une description et une durée moyenne conseillée. La liste est dynamique, et peut être filtrée via le champ de recherche.

NOTIFICATIONS

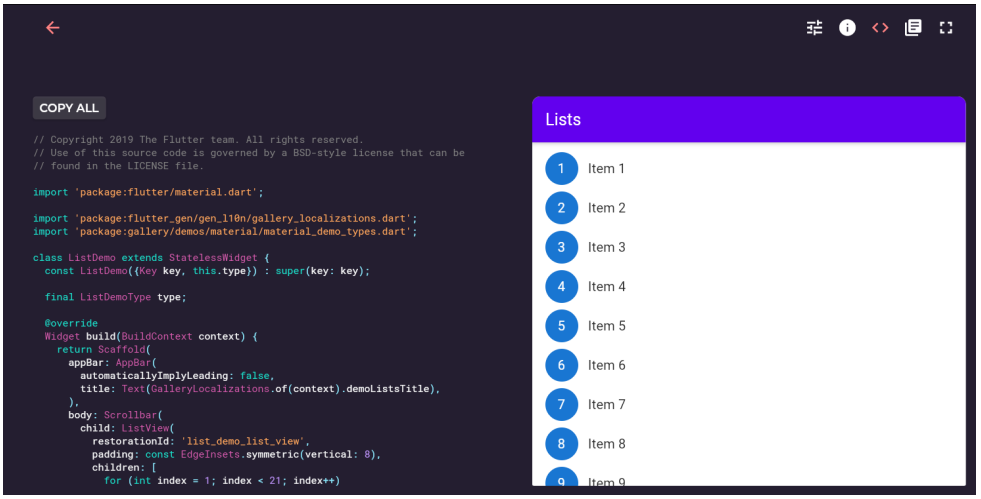
Les notifications permettent simplement de garder une trace des événements qui ont lieu sur l'application : une simple liste d'événements est donc affichée, avec le nom de l'utilisateur qui en est à l'origine et l'heure à laquelle il a eu lieu.

FLOATING BUTTON

Pour finir, un floating button en bas à droite de l'écran est présent sur tous les écrans pour permettre de créer un nouveau modèle ou une séance à tout moment. Dans la mesure où l'utilisation de champs de formulaires a déjà été faite dans la section *Recherche d'exercices*, l'objectif était de tester l'implémentation du floating button lui-même, et le formulaire de création n'a donc pas été implémenté.

CHOIX EFFECTUÉS

Étant intéressé par la technologie mais n'ayant jamais vraiment développé de projet avec Flutter, j'ai d'abord souhaité trouver des sources d'exemples fiables pour apprendre les normes et bonnes pratiques de ce langage. J'ai donc utilisé Flutter Gallery pour apprendre à travailler avec des composants génériques.



À l'aide de cet outil et de la doc Flutter, j'ai d'abord souhaité travailler sur l'implémentation du router pour passer d'une section à l'autre. Pour aller un peu plus loin sur ce point, je me suis intéressé au fonctionnement des transitions et des animations pour rendre le passage d'une page à l'autre le plus agréable possible visuellement. J'ai donc choisi de considérer chaque page comme un widget qui viendra remplacer le précédent dans la vue.

```
final List<Widget> _pages = [
  const Home(),
  const Search(),
  const Notifications(),
  const Account(),
];

. . .

body: Center(
  child: PageTransitionSwitcher(
    transitionBuilder: (child, animation, secondaryAnimation) {
      return FadeThroughTransition(
        animation: animation,
        secondaryAnimation: secondaryAnimation,
        child: child,
      );
    },
    child: _pages[_currentIndex.value],
  ),
),
```

Pour ce qui est de l'UI, Flutter utilise un objet particulier qui permet de définir un ensemble de valeurs par défaut pour le thème de l'application. Cette méthode permet de définir dans un seul fichier toutes les couleurs, formes, bordures etc... et donc évite une grande quantité de répétition du code tout en permettant une modification rapide et efficace de l'interface.

```
return ThemeData(
  primaryColorDark: const Color(0xff1C2C48),
  backgroundColor: Colors.black,
  primaryColorLight: const Color(0xff47ABFF),
  fontFamily: 'Montserrat', //3
  textTheme: const TextTheme(
    bodyText1: TextStyle(
      color: Colors.white,
    ),
  ),
  buttonTheme: ButtonThemeData(
    shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(18.0)),
    buttonColor: Colors.purple[50],
  ),
);
```

PISTES D'AMÉLIORATION

Un problème majeur que j'ai rencontré en travaillant sur la liste des exercices a été de créer un lien vers une vidéo Youtube. Il s'avère que faire appel à une autre application demande de passer par des librairies qui impliquent d'interagir avec du code natif android et ios ce qui dans le cadre du projet allait au-delà de ce que je souhaitais faire.

D'autres difficultées ont été rencontrées lors du développement, majoritairement dues au nouveau paradigme apporté par Flutter par rapport aux technologies web auxquelles je suis habitué comme React. À titre d'exemple, la construction d'une interface en Flutter consiste en un arbre de widgets imbriqués dont les branches sont reconstruites en fonction du state de l'app.

Enfin, ne pouvant pas garantir l'accès au backend original du projet platine, j'ai du passer par une open api pour récupérer des posts aléatoires qui manquent donc de certaines données propres à MODELIA, ce qui n'à permi de couvrir qu'une partie de l'interface. Néanmoins, même en passant par des api ouvertes ou même des mocks, l'application finale fait tout de même un bon tour des fonctionnalités principales attendues : un système de routage, une gestion des états, des requêtes HTTP asynchrones, des transitions d'une section à l'autre, des

animations de chargement, la gestion de l'UI via un thème global ainsi que des champs de formulaire avec filtrage dynamique de données.