Développement mobile avancé, IoT et embarqué

Rapport de Projet

Auteur :

SAR Alexandre

Décembre 2024

Lien GitHub du Projet : https://github.com/texao/Iot_projet_flutter

Table des matières

1 Introduction

Dans le cadre de ce projet, j'ai développé deux applications en utilisant Flutter : une carte de profil et un quizz interactif. L'objectif principal était de me familiariser avec les widgets `StatelessWidget` et `StatefulWidget`, tout en apprenant à structurer une interface utilisateur moderne et fonctionnelle.

La carte de profil m'a permis d'explorer la création de composants visuels statiques, alors que le quizz m'a aidé à comprendre la gestion de l'état et des interactions utilisateur. Ce TP a été une excellente opportunité pour approfondir mes compétences en développement mobile avec Flutter, tout en découvrant des outils et des concepts essentiels, comme les `Scaffold` ou `AlertDialog`.

2 Profile Card

2.1 Objectifs

L'objectif de cet exercice est de créer une interface graphique en Flutter représentant le profil d'une personne. L'application doit inclure un avatar et des informations personnelles, telles que le nom, l'email, et l'adresse.

Il faut comprendre et d'utiliser des widgets `Stateless` et des éléments d'interface pour concevoir une page de profil fonctionnelle.

2.2 Conception

L'application utilise un widget `ProfileCard` héritant de la classe `StatelessWidget`. Ce widget représente la page d'accueil et est composé de plusieurs éléments distincts permettant d'afficher les informations du profil.

L'interface affiche une carte de profil contenant les éléments suivants :
- Un avatar circulaire affichant une photo de profil.
- Les informations personnelles : nom, email, adresse, et ville, compte Twitter.
- Deux boutons permettant de partager ou modifier le profil.

L'application repose sur des composants Flutter comme :
- `Card` pour structurer l'ensemble du contenu.
- `CircleAvatar` pour afficher une photo de profil circulaire.
- `Column` et `Row` pour organiser les différents éléments de manière verticale et horizontale.

Chaque détail du profil est affiché avec une icône descriptive et un texte stylisé pour avoir une bonne lisibilité. Les boutons interactifs ajoutent une fonctionnalité permettant de manipuler ou partager le profil utilisateur.

Style personnalisés :

L'interface utilise des styles personnalisés pour améliorer l'apparence et l'expérience utilisateur. Par exemple, une police élégante, `Google Fonts Poppins`, est utilisée pour le texte, avec un style en italique. De plus, il y a un effet d'ombre qui est appliqué au texte, avec un décalage et un flou pour créer un effet qui renforce la lisibilité, notamment sur des arrière-plans clairs. Enfin, il y a des couleurs cohérentes, comme que le bleu pour les icônes et le noir pour le texte qui sont utilisées pour maintenir avoir un meilleur visuel dans l'interface.

Affichage :

J'ai également créé le widget `ProfileDetailRow` pour afficher les informations. Ce widget permet de présenter une ligne contenant une icône, une étiquette et une valeur associée.

Le widget utilise un `Row` pour aligner horizontalement l'icône et le texte, avec un espace défini entre les deux. Le texte est stylisé à l'aide de la police Google `Poppins`, avec une taille de police spécifique, un effet d'ombre subtil, et un léger espacement entre les lettres pour améliorer la lisibilité.

Ce widget est conçu pour être utilisé dans des cartes de profil ou toute interface nécessitant l'affichage d'informations structurées de manière claire et esthétique.

```dart
class ProfileDetailRow extends StatelessWidget {
  final IconData icon;
  final String label;
  final String value;

  const ProfileDetailRow({
    super.key,
    required this.icon,
    required this.label,
    required this.value,
  });

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 10.0),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Icon(icon, color: Colors.blue.shade700, size: 30),
          const SizedBox(width: 18),
          Expanded(
            child: Text(
              "$label: $value",
              style: GoogleFonts.poppins(
                textStyle: TextStyle(
                    color: Colors.black87,
                    fontSize: 19,
                    fontWeight: FontWeight.w600, // Semi-gras
```

FIGURE 1 – Affiche des informations

Sur la figure ci-dessus, nous pouvons voir le code permettant l'affichage des informations personnelles comme que l'email, le lien Twitter, l'adresse, et d'autres détails dans un profil utilisateur. L'utilisation de `Row` pour organiser l'icône et le texte, ainsi que l'application de styles personnalisés (comme l'ombre et l'espacement des lettres), contribue à rendre l'interface élégante et facile à lire.

Interface :

L'interface affichée sur la figure ci-dessous montre un profil utilisateur. Elle présente différentes informations personnelles avec un format structuré. L'interface est organisée à l'aide de composants comme des icônes et du texte, ce qui permet une lecture facile. Le tout est disposé avec des effets comme des ombres et des espacements pour rendre l'affichage plus agréable. Cette organisation permet à l'utilisateur de rapidement repérer les informations importantes du profil.

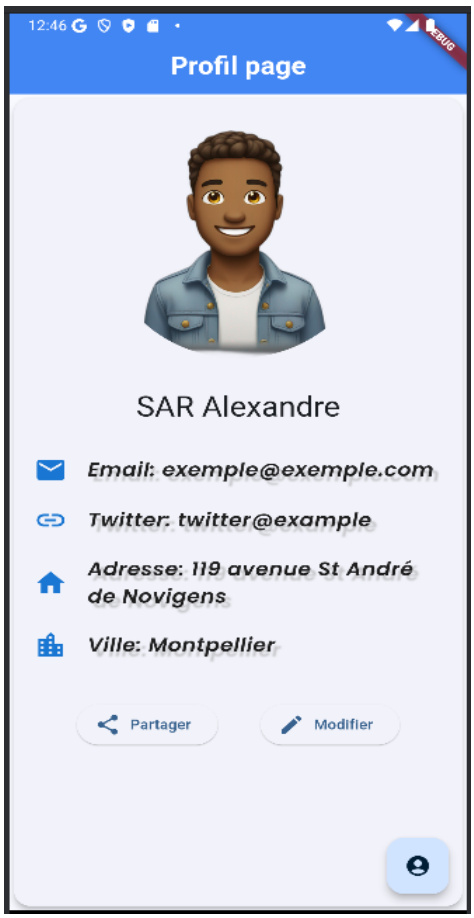

FIGURE 2 – Interface

3 Quizz

3.1 Objectifs

L'application de quizz repose sur un widget `StatefulWidget` nommé `QuizzPage`, qui gère l'état de l'application pour suivre la progression des questions et le score de l'utilisateur. Les données des questions sont définies dans une classe `Question`, qui encapsule les attributs nécessaires : le texte de la question, un booléen indiquant si la réponse est correcte, et une URL d'image qui est optionnelle pour certaines question. Ces choix de conception permettent de structurer clairement les données et de rendre l'application facilement extensible.

3.2 Conception

Architecture : L'architecture de l'application suit un modèle simple où le composant visuel et les données des questions sont distincts. Cela permet une gestion claire de l'état et de la logique de navigation. Voici un aperçu des principaux choix de conception :

- Utilisation de la classe `StatefulWidget` pour gérer dynamiquement les questions et le score.
- Séparation des données avec (classe `Question`) et de l'interface utilisateur (`QuizzPage`) pour une meilleure organisation.
- Utilisation d'un système de navigation intégré pour afficher un résumé des résultats dans une boîte de dialogue (`AlertDialog`).

Workflow : Voici le workflow de l'application :

1. Affichage d'une question avec son texte et, si disponible, une image associée.
2. L'utilisateur sélectionne une réponse en cliquant sur les boutons `Vrai` ou `Faux`.
3. Un message de confirmation apparaît (`SnackBar`), cela indique si la réponse est correcte ou non.
4. L'application passe automatiquement à la question suivante jusqu'à la fin du quizz.
5. Une fois toutes les questions répondues, un résumé des résultats s'affiche.

Composants principaux : Les principaux composants Flutter utilisés sont :

- `Scaffold` : fournit la structure principale de l'interface, qui inclue l'AppBar et le corps.
- `Column` et `Row` : permettent d'organiser verticalement et horizontalement les éléments visuels comme les boutons et le texte.
- `Image.network` : charge dynamiquement des images pour les questions qui en contiennent.
- `SnackBar` : affiche un retour visuel temporaire après chaque réponse.
- `AlertDialog` : présente le score final dans une boîte de dialogue interactive.

Choix de conception : Le choix d'utiliser des widgets standards de Flutter, comme `Scaffold` et `SnackBar`, permet de construire une application cohérente. De plus, la gestion des données par une classe dédiée (`Question`) améliore la lisibilité et la modularité du code. Ce modèle pourrait être étendu pour inclure plus de types de questions ou des fonctionnalités supplémentaires.

Vous pouvez regarder la vidéo présentant le quizz. Elle montre le déroulement du quizz, depuis l'affichage de la première question jusqu'au résumé final.

4 Conclusion

Ce TP m'a permis de réaliser deux applications complètes et interactives en Flutter, en appliquant des concepts clés comme la gestion de l'état et l'organisation des widgets. Grâce à ces exercices, j'ai appris à structurer mon code, à styliser des interfaces, et à gérer des interactions utilisateur dynamiques.

La création du quizz m'a particulièrement aidé à comprendre l'importance de la modularité. Je pense que ces bases me permettront de développer des applications mobiles plus complexes à l'avenir. Cette expérience a renforcé ma compréhension de Flutter.