Projet - Création de votre première App Flutter

Table des matières

I. Initialisation du projet

Durée : 1 H 30

Environnement de travail : avoir installé sur son ordinateur VS Code, Dart sdk et Flutter SDK

Flutter[1]

Prérequis : avoir les bases de Dart nécessaires à la compréhension de Flutter, savoir créer un projet Flutter, connaître les widgets de bases du langage flutter

Contexte

Nous allons créer une application mobile en utilisant le *framework* Flutter.

Il s'agit d'une application qui analyse un mot pour déterminer le nombre de chacune des voyelles et le nombre de consonnes. Nous créerons nos propres widgets.

1 https://docs.flutter.dev/get-started/install

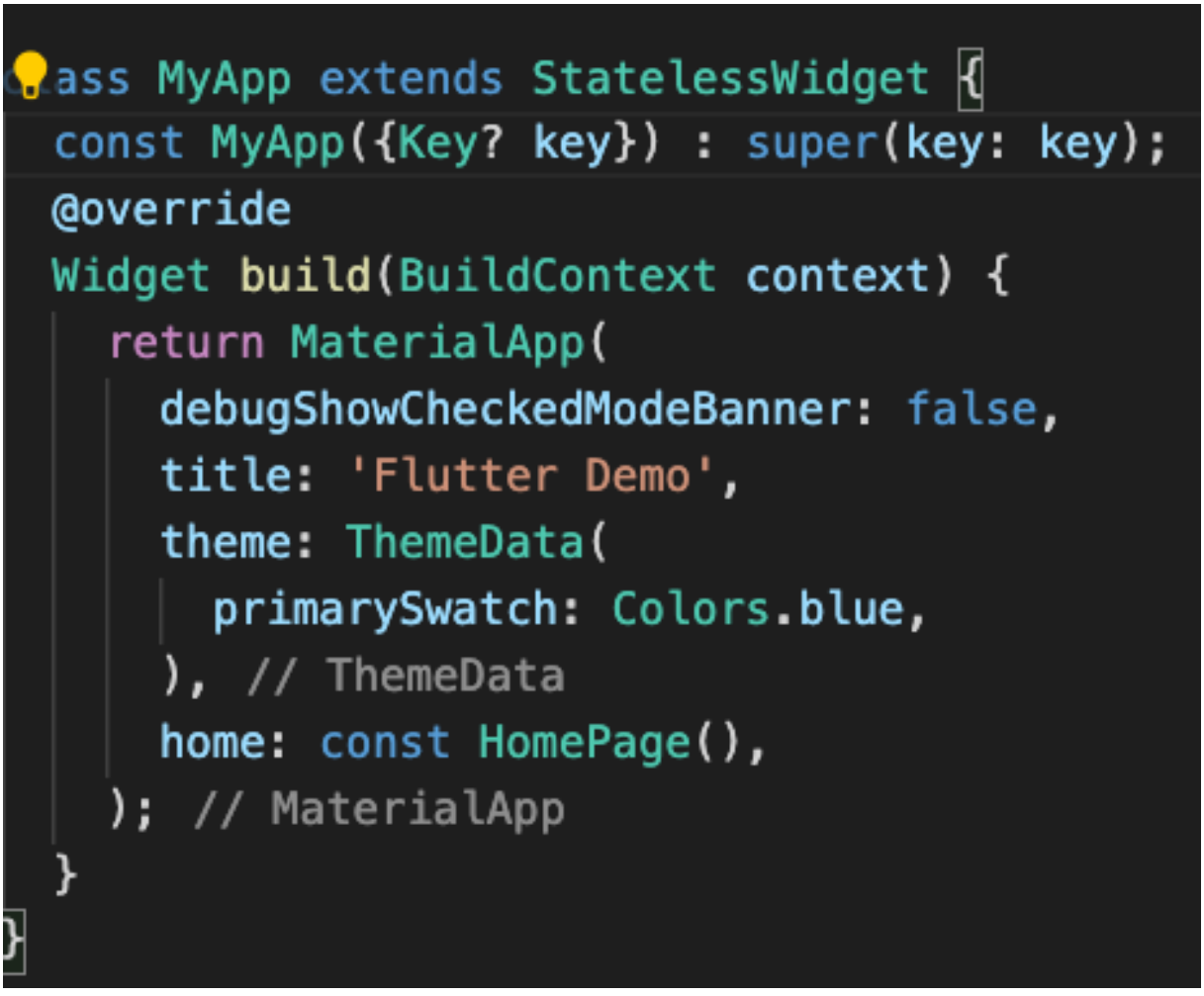
Objectifs

- Initialiser un projet avec vScode
- Faire le ménage dans le code
- Création d'un widget pour l'écran

Contexte

Tout projet commence par son initialisation. C'est une étape qu'il ne faut pas négliger. Nous ferons le ménage dans le projet de base. Nous allons mettre en place la structure du projet.

Nous allons générer l'application de base. Ensuite nous effacerons tous les commentaires. Il faudra alors créer le widget qui gère l'écran principal dans un fichier séparé. Ce widget s'appellera `< HomePage >`. Il sera appelé dans le `< main.dart >`.



II. Mise en place d'une icône pour notre application

Objectifs

- Générer les fichiers correspondants aux différents formats pour Android et iOS
- Copier les fichiers dans le dossier pour Android
- Copier le fichier dans le dossier pour iOS

Contexte

Pour toute application, il est important d'avoir une icône. Celle-ci permet d'identifier notre application.

Pour générer l'ensemble des fichiers pour l'icône, il faut une image de 1024 * 1024. À partir de cette image, nous allons utiliser un site tiers qui va nous générer les fichiers pour tous les écrans.

Ensuite, nous allons glisser déposer les fichiers pour Android et ceux pour iOS.

Complément

App Icon Generator[1]

III. Importer une police de caractère

Objectifs

- Choix d'une police de caractère sur Google Fonts
- Télécharger la police et la mettre en place
- Création d'un label pour tester la police de caractère mise en place

Contexte

Pour personnaliser une application, il est utile d'ajouter une police de caractère. Google Fonts fournit des polices de caractères.

Nous allons devoir, dans notre devoir créer un dossier « ***fonts*** » pour stocker nos polices de caractères. Nous allons copier la police télécharger sur Google Fonts dans le dossier « ***fonts*** ».

manières de gérer cela.

Méthode

Pour intégrer la police de caractères, nous allons devoir modifier le fichier `< pubspec.yaml >`. Il va falloir respecter les tabulations. À l'intérieur de l'écran principal, nous allons afficher un texte avec un style utilisant la police ajoutée.

IV. Créer une zone de couleur pour une voyelle

1 https://appicon.co/

Objectifs

- Création d'une Colonne qui va contenir une zone pour chaque voyelle et une pour les consonnes donc 7 rectangles au bord arrondi
- Créer un *container*, rectangle coloré
- Positionner les textes : le nom de la voyelle et le nombre d'occurrences

Contexte

Il convient de créer des zones pour afficher le nombre d'occurrences pour chaque voyelle. Pour commencer, nous allons directement intégrer un container de couleur dans le *widget* principale. Ce container va correspondre à une de nos zones d'affichage.

Il convient d'insérer un widget « ***column*** » à la propriété *body* du widget « ***Scafold*** ». Les enfants ou « ***children*** » du widget « ***column*** » vont contenir les différentes zones pour chaque voyelle. On va créer une première zone.

On lui définit une couleur de *background*. À l'intérieur du container on affiche le nom d'une voyelle et « ***0 occurrences*** ». Le texte est stylisé en utilisant la police de caractère que l'on a intégrée.

V. Création d'un widget pour générer les zones de chaque lettre

Objectifs

- Créer le widget qui va créer la zone pour chaque voyelle et les consonnes
- Utilisation du widget pour générer les 7 zones à l'intérieur de notre colonne

Contexte

Pour gagner du temps et rendre notre projet plus lisible, il semble opportun de créer un widget pour générer les 7 zones. Le widget recevra en paramètres : la couleur de la zone, la lettre et le texte indiquant le nombre d'occurrences.

Nous allons créer notre propre widget : « ***RowLettre*** ». Ce widget sera de type « ***stateLess*** » car il ne sert à afficher que des données. Dans notre « ***column*** » du widget principal, nous appellerons 7 fois le widget « ***RowLettre*** » pour construire les 7 zones qui contiendront les informations sur la composition du mot.

VI. Insertion d'une zone pour saisir un mot et un bouton pour faire l'analyse

Objectifs

Mettre en place les composants pour permettre à l'utilisateur d'interagir avec l'application

Contexte

Il convient d'offrir la possibilité à l'utilisateur de saisir le mot dont il veut connaître la composition en voyelle et consonne. Il convient aussi d'ajouter un bouton pour lancer l'analyse.

Pour permettre à l'utilisateur de saisir un mot, nous allons utiliser un widget « ***TextField*** ». Pour lancer, l'analyse, nous allons utiliser un widget « ***elevation*** ». Si on a un message de débordement pour nous signaler que tout l'écran ne peut pas être afficher tout, nous intégrerons le tout dans une « ***singleScrollView*** ».

VII. Ajout d'une structure de données et des fonctions qui vont la gérer

Objectifs

- Intégrer une structure de données pour stocker la répartition des lettres
- Créer les fonctions qui vont incrémenter la structure de données
- Créer la fonction qui va intégrer une boucle pour parcourir le mot saisi

Contexte

Pour ne pas créer un compteur individuel pour chaque lettre, on va créer une structure de données élaborer. Des méthodes feront évoluer cette structure de données. Il est nécessaire d'implémenter.

Pour stocker les compteurs des lettres, nous allons utiliser une structure de données de type dictionnaire ou une clé et associé à un nombre. Ici la clé sera le nom de la lettre et la valeur sera le nombre d'occurrences.

Méthode

Pour créer le dictionnaire, en dart, il faut utiliser le mot clé < map >. Notre structure de données sera définie ainsi :

```
Map <String, int> voyelleDico = {'a':0,'e':0,'i':0,'o':0,'u':0,'y':0};
```

Étape 1 :

La structure de données est définie dans le « ***state*** » du widget.

Étape 2 :

Ensuite, on crée les méthodes qui vont incrémenter les valeurs à l'intérieur du dictionnaire. Celles- ci vont modifier le « ***state*** » de notre structure de données.

Par exemple, la fonction < `IncrementeA` > est la suivante :

```
void _incrementeA(){
  setState(() {
    int? value = voyelleDico['a'];
    value = value! + 1;
    voyelleDico['a'] = value;
  });
}
```

Étape 3 :

Enfin une fonction **< `analyse` >** va boucler sur le mot pour incrémenter les bonnes entrées du dictionnaire.

```
void _analyse(){

  _resetDico();
  for(var i=0; i < monMot.length; i++){
    switch(monMot[i]){
      case 'a':_incrementeA();
      break;
      case 'e':_incrementeE();
      break;
      case 'i':_incrementeI();
```

```
        break;
      case 'o':_incrementeO();
        break;
      case 'u':_incrementeU();
        break;
      case 'y':_incrementeY();
      break;
      default: {_incrementeConsonnes(); }
    break;

    }
  }
}
```

À chaque fois que le « ***state*** » est modifié, l'affichage va être rafraîchi.

VIII. Pour aller plus loin...

Ce module de projet guidé est maintenant terminé, mais vous pouvez ajouter des fonctionnalités. Vous pouvez enregistrer chaque résultat de composition dans une liste. Vous pourrez ajouter un *drawer* ou menu latéral à l'application.