TP 2 Flutter - Rapport

Kemmoun Ramzy / Abadli Baddredine M2 IV

Exercice 1

Nous avons d'abord importé le package material.dart pour utiliser les composants Flutter. Ensuite, nous avons créé un répertoire nommé images dans le projet et ajouté des images dedans. Pour rendre ces images disponibles, nous les avons déclarées dans le fichier pubspec.yaml sous assets. Enfin, dans notre code Flutter, nous avons créé un widget Stateful et utilisé Image.asset('images/lt.png') pour afficher l'image directement.

nous avons créé aussi une fonction rotate qui gère la rotation d'une image en incrémentant un angle à chaque clic sur un bouton. L'angle est mis à jour dans la méthode setState, permettant à l'interface de se rafraîchir. Un FloatingActionButton a été ajouté avec une étiquette "Rotate" qui appelle cette fonction lorsqu'il est pressé. L'image est affichée à l'aide d'un widget Transform.rotate, qui applique la rotation basée sur l'angle en radians, permettant ainsi à l'utilisateur de faire pivoter l'image en continu.

Image Rotation

Exercice 2

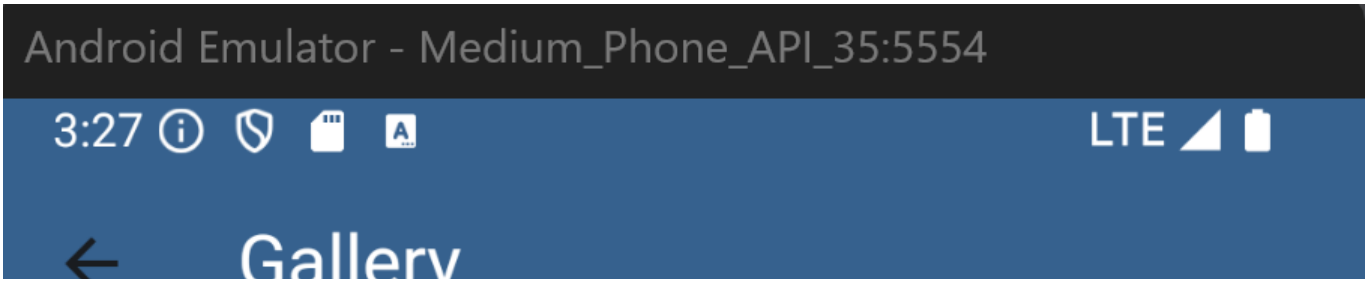
Dans cet exercice, nous avons implémenté la navigation, pour cela nous avons créé une seconde fenêtre en ajoutant un widget StatelessWidget pour la nouvelle page, et nous avons ajouté une instruction Navigator.push pour permettre de passer de la première à la deuxième fenêtre. et un button dans la deuxième fenêtre pour retourner à la première page.

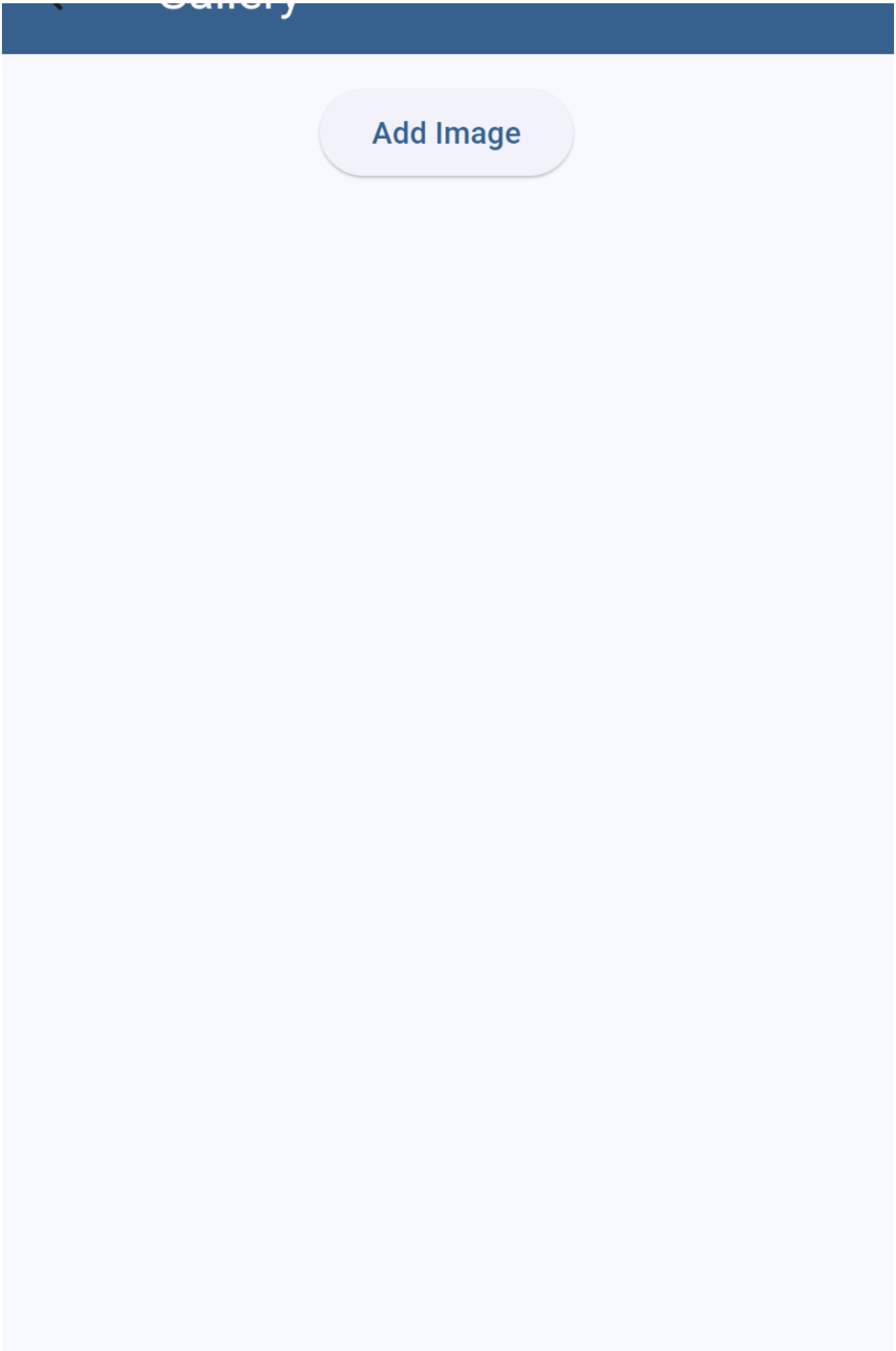
Navigation

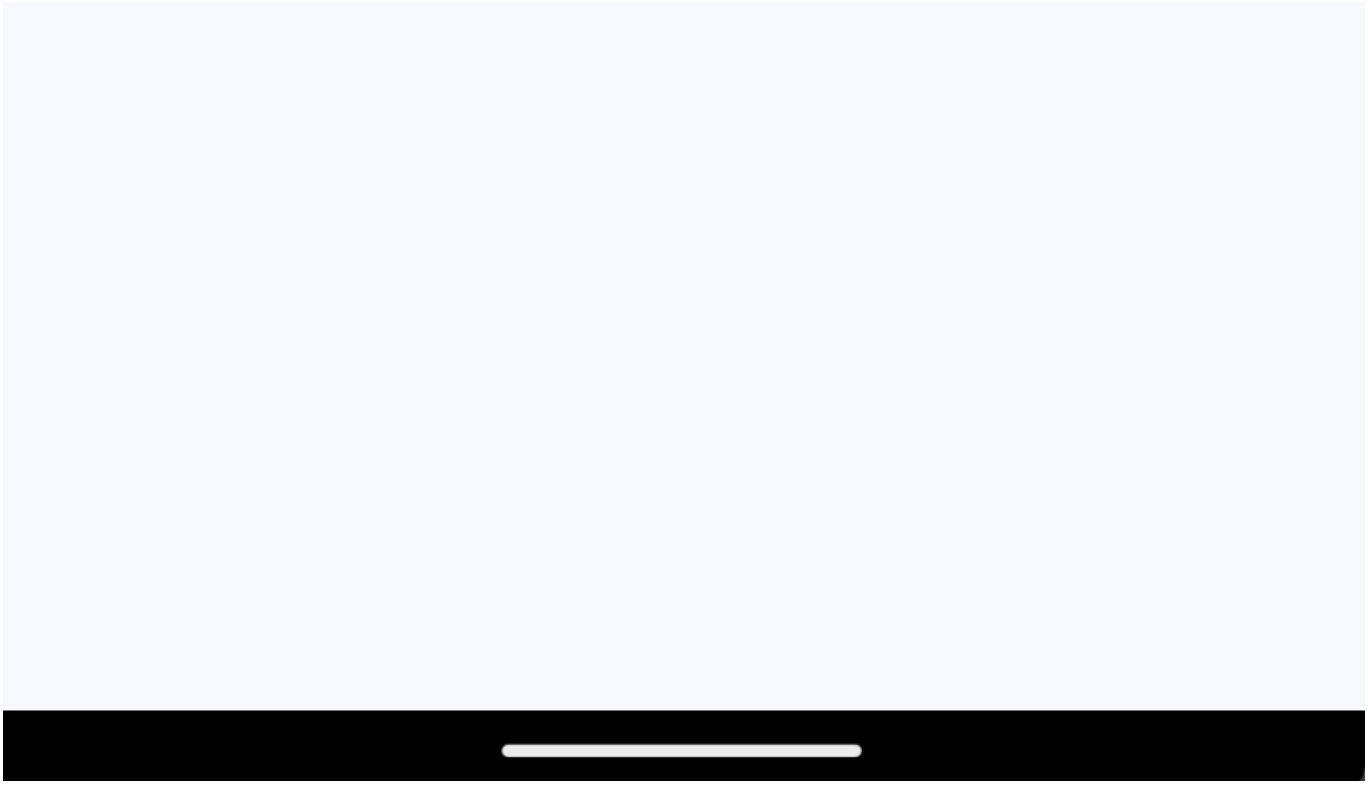
Exercice 3

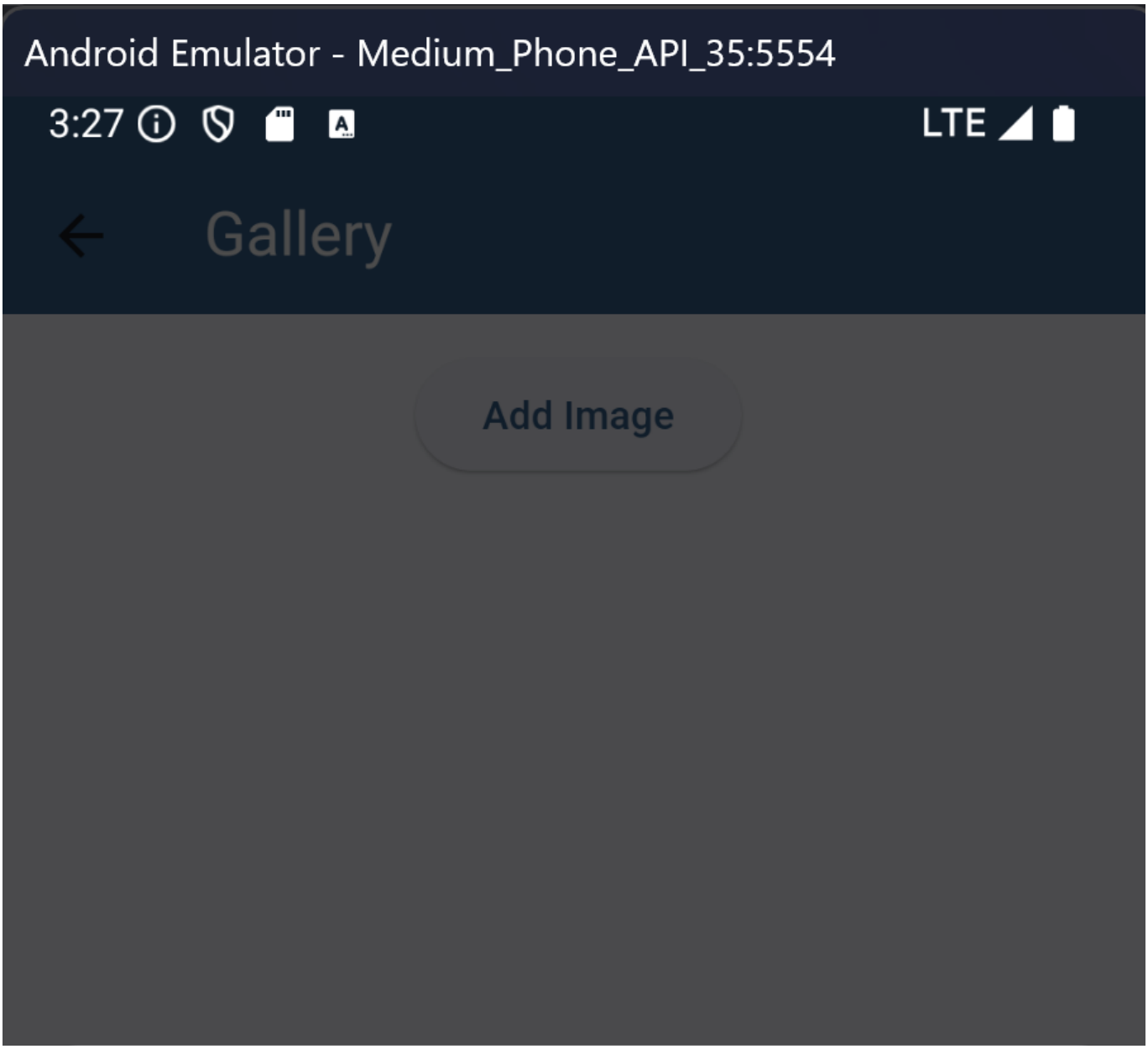
Dans le troisième exercice, nous avons mis en place une galerie d'images en utilisant le package image_picker, que j'ai installé pour permettre à l'utilisateur de sélectionner des images à partir de la galerie de son appareil. J'ai ensuite utilisé un GridView.count pour afficher ces images sous forme de grille. Chaque image est cliquable et, lorsque l'on appuie dessus, elle redirige vers un écran de détails de l'image, permettant d'afficher l'image sélectionnée en plus grande taille et avec plus d'informations.

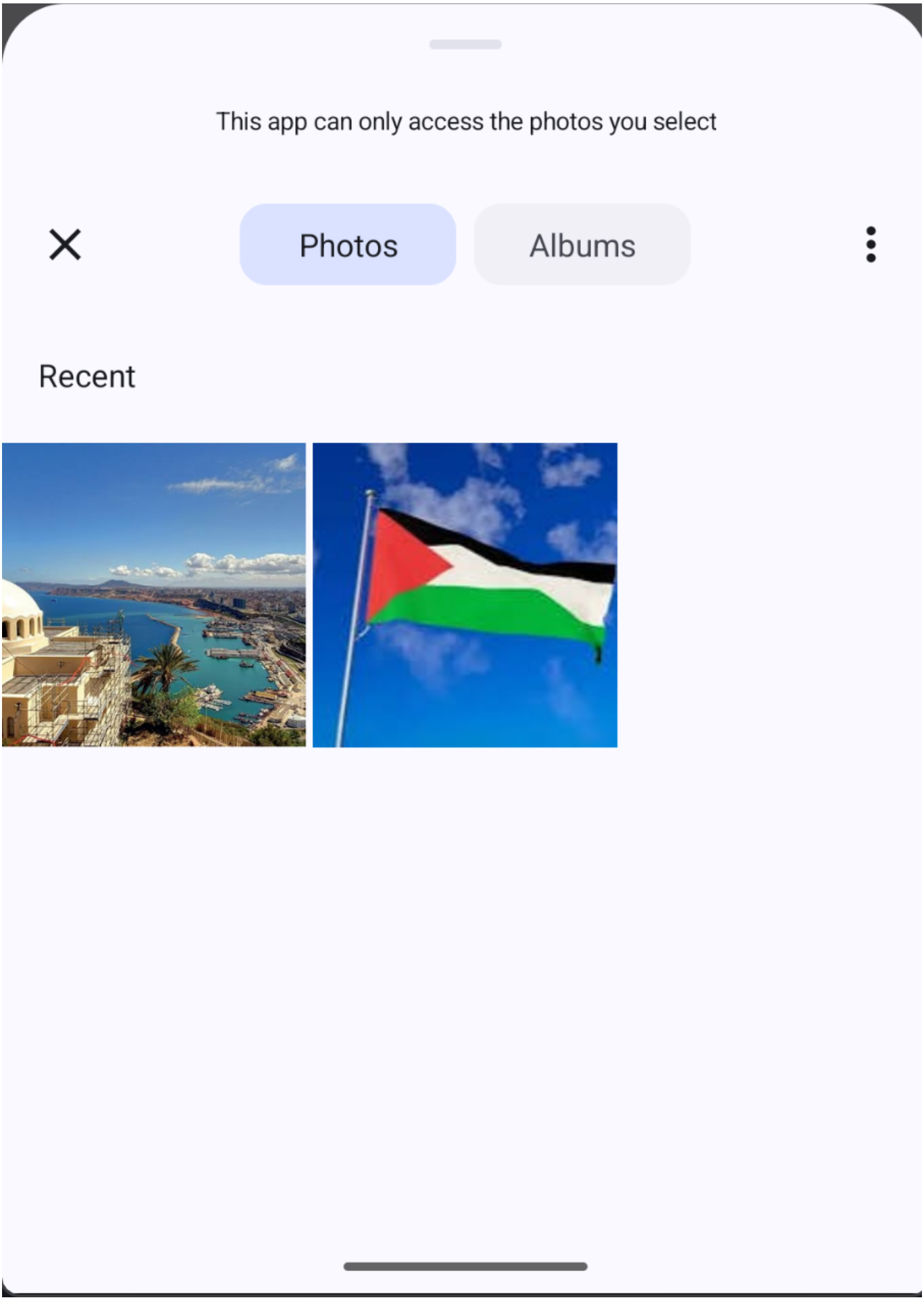
Dans l'utilisation de la bibliothèque image_picker, nous utilisons les mots-clés async et Future pour gérer les opérations asynchrones. Lorsqu'un utilisateur sélectionne une image, une fonction marquée async permet d'attendre le résultat du Future retourné par le image_picker. Cela garantit que notre application reste réactive pendant que l'image est chargée, sans bloquer l'interface utilisateur. Cette approche rend le processus de sélection d'image plus fluide et agréable pour l'utilisateur.

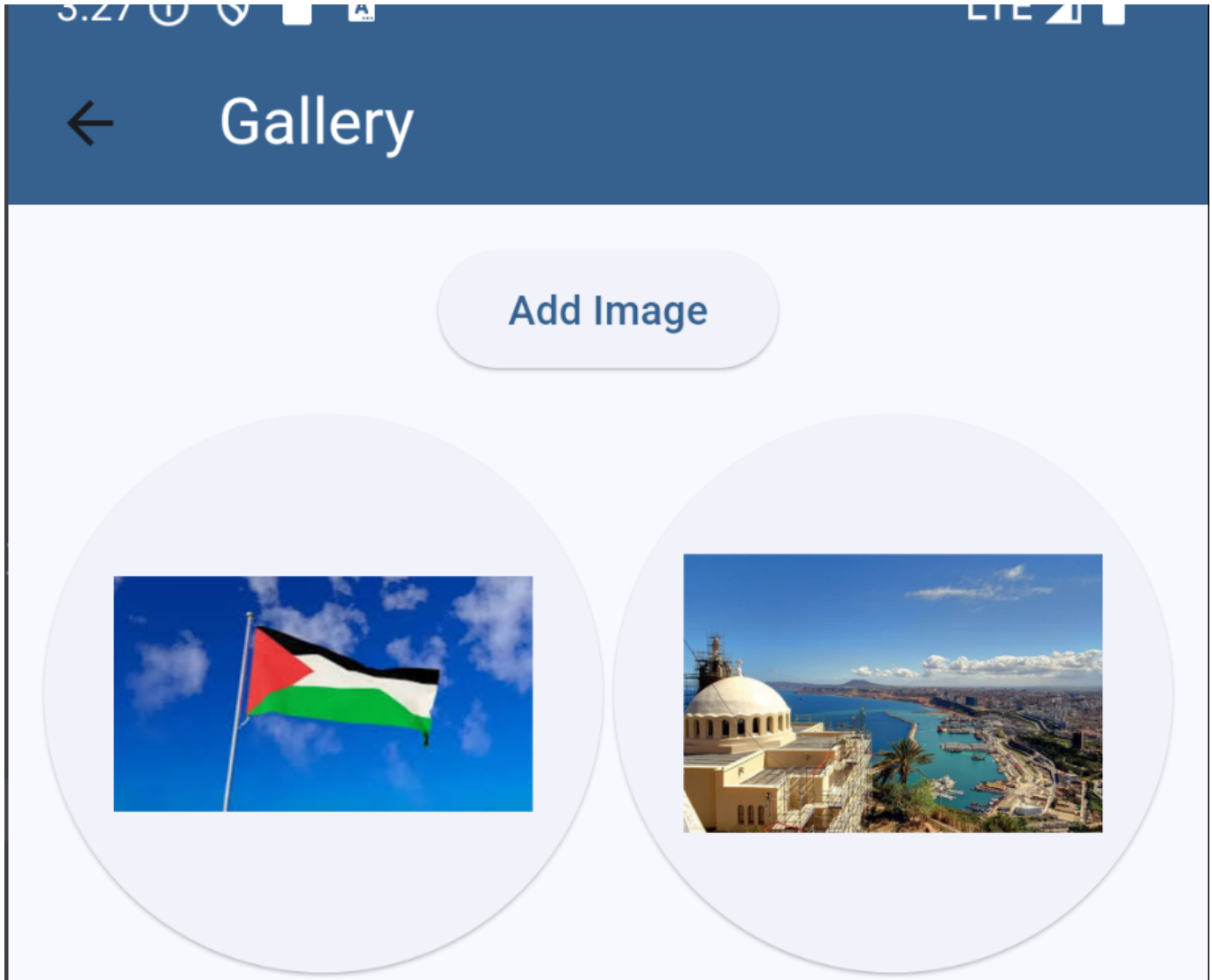


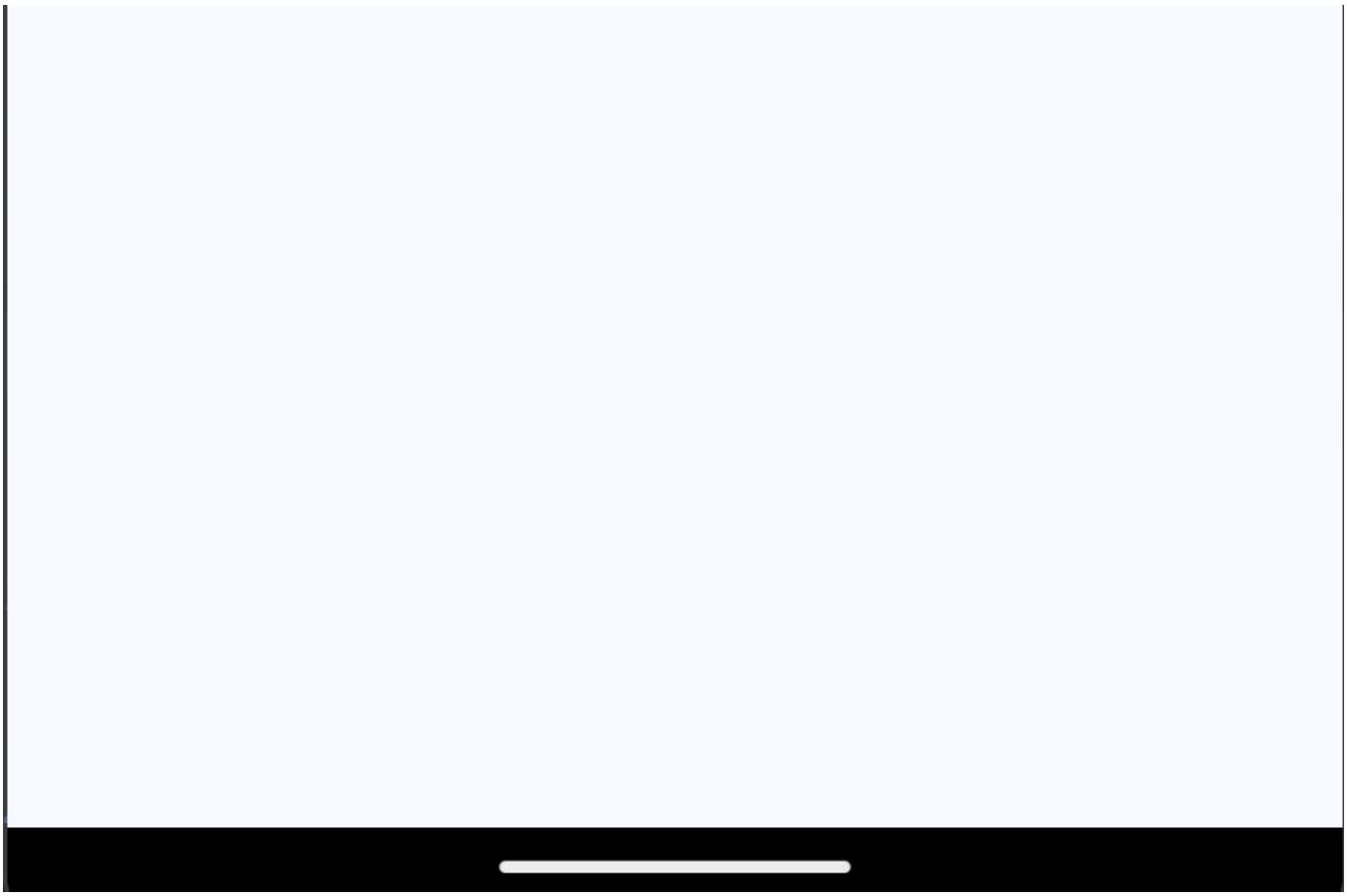

Add Image

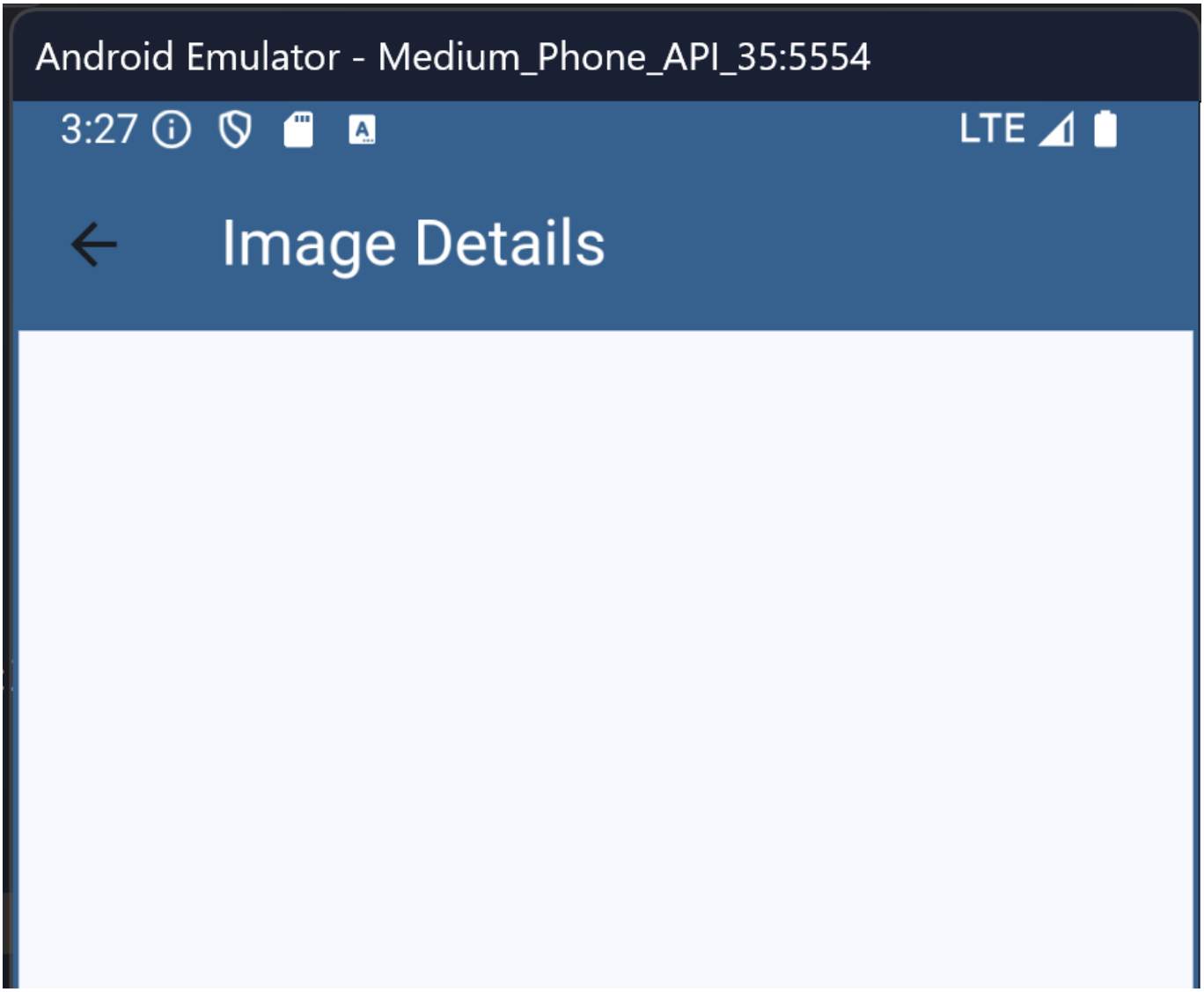



Android Emulator - Medium_Phone_API_35:5554
3:27
LTE
Gallery
Add Image


This app can only access the photos you select
Photos
Albums
Recent
Android Emulator - Medium_Phone_API_35:5554


Gallery
Add Image




Android Emulator - Medium_Phone_API_35:5554
3:27
LTE
Image Details

En conclusion, Flutter facilite grandement le développement d'applications mobiles grâce à sa large gamme de packages disponibles. Ces outils permettent d'intégrer rapidement des fonctionnalités complexes, comme la sélection d'images avec image_picker, tout en offrant une interface utilisateur intuitive. L'utilisation de GridView pour afficher des galeries d'images montre à quel point Flutter rend la création d'interfaces visuellement attrayantes et interactives simple et efficace. Ainsi, les développeurs peuvent se concentrer sur l'expérience utilisateur sans se soucier des détails techniques sous-jacents.