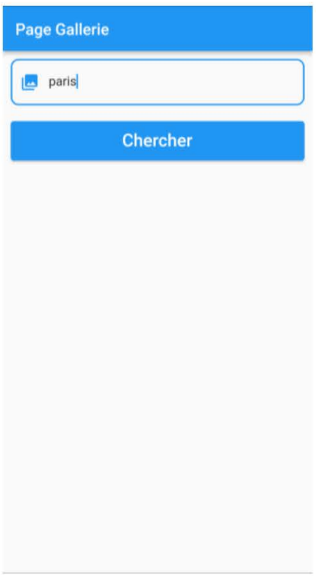
Atelier 6

Pages Gallerie & Gallerie-Details

Matière : ATELIER FRAMEWORK CROSS-PLATFORM	DSI3
Enseignants : M. Hadiji & S. Hadhri	

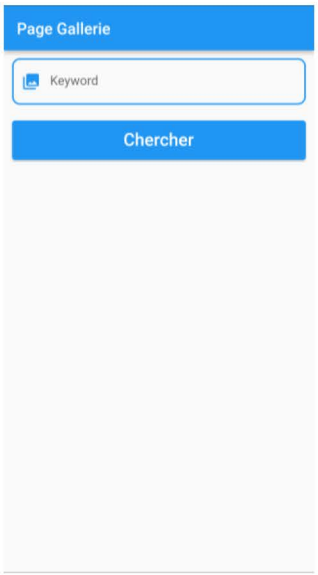
L'objectif de cet atelier est la réalisation de 2 pages Gallerie et Gallerie-Details :



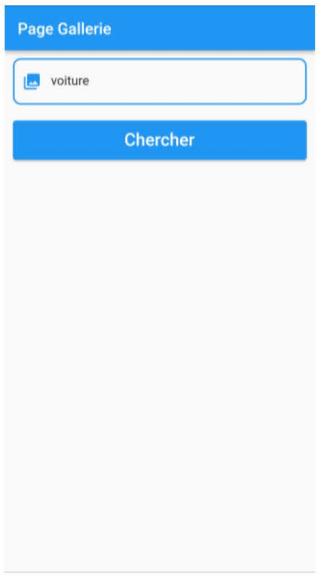
Page Gallerie

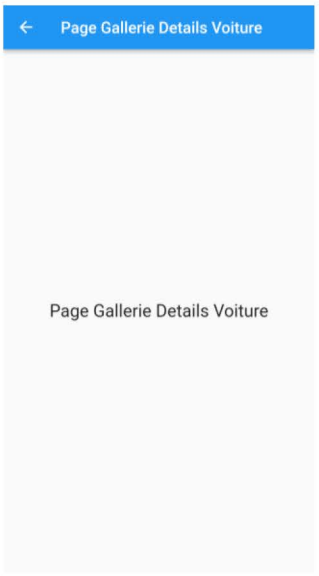
Page Gallerie-Details

1. Créer dans la page Gallerie un formulaire formé par une zone de texte et un bouton.



2. Sous le dossier « lib/*pages* », créer une nouvelle page « *gallerie-details.page.dart* » affichant le message 'Page Gallerie-Details' au centre de la page.
3. Développer la méthode privée _onGetGallerieDetails() appelée au clic sur le bouton « Chercher » qui permet de basculer vers la page GallerieDetails en lui passant la clé saisie. Vérifier que le passage s'est bien passé en affichant la clé saisie dans la page Gallerie-Details.





4. Convertir la page Gallerie-Details en StatefulWidget puis redéfinir la méthode initState() pour qu'elle appelle une fonction getGalleryData(keyword).
5. **Présentation du service Pixabay**

Pixabay.com est un site web de partage d'images libres de droits : photos, illustrations, images vectorielles et clips vidéo.
Pixabay fournit une API pour rechercher et récupérer des images et des vidéos gratuites publiées. Elle renvoie des objets JSON.

Remarque :
La page de documentation de l'API est accessible ici https://pixabay.com/service/about/api/

Utiliser pixabay nécessite de s'enregistrer afin d'obtenir une clé API.

➔ Les paramètres de l'API sont :

- **Key** : clé obligatoire
- **q** : Un terme de recherche. Si omis, toutes les images sont retournées. Cette valeur ne doit pas dépasser 100 caractères.
  Exemple : "jaune + fleur"
- **per_page** : un entier qui représente le nombre de résultats par page (par défaut: 20)
- **Page :** les résultats de recherche renvoyés étant paginés, ce paramètre représente le numéro de la page.

La réponse JSON de cette requête https://pixabay.com/api/?key={ KEY }&q=fleurs est la suivante :


```
{
  "total": 4692,
    "totalHits": 500,
    "hits": [
  {
    "id": 195893,
    "pageURL": "https://pixabay.com/en/blossom-bloom-flower-195893/",
    "type": "photo",
    "tags": "blossom, bloom, flower",
    "previewURL": "https://cdn.pixabay.com/photo/2013/10/15/09/12/flower.jpg"
    "previewWidth": 150,
```

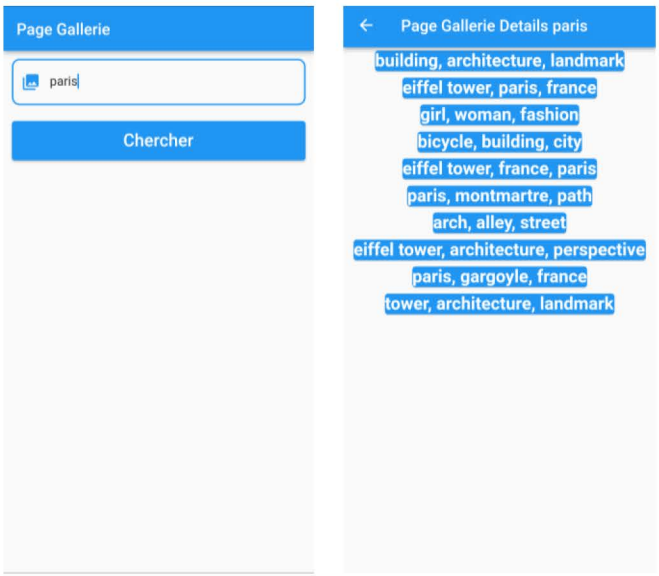
```
    "previewHeight": 84,
    "webformatURL": "https://pixabay.com/get/35bbf209e13e39d2_640.jpg",
    "webformatWidth": 640,
    "webformatHeight": 360,
    "largeImageURL": "https://pixabay.com/get/ed6a99fd0a76647_1280.jpg",
    "fullHDURL": "https://pixabay.com/get/ed6a9369fd0a76647_1920.jpg",
    "imageURL": "https://pixabay.com/get/ed6a9364a9fd0a76647.jpg",
    "imageWidth": 4000,
    "imageHeight": 2250,
    "imageSize": 4731420,
    "views": 7671,
    "downloads": 6439,
    "favorites": 1,
    "likes": 5,
    "comments": 2,
    "user_id": 48777,
    "user": "Josch13",
    "userImageURL": "https://cdn.pixabay.com/user/2013/11/05/02-10-23.jpg",
  },
  {
    "id": 73424,
    ...
  },
  ...
]
}
```

6. Coder la fonction getGalleryData(keyword) qui permet de lancer une requête http et récupérer l'état de la gallerie sous format JSON dans une variable galleryData. Vérifier la réception de la réponse en l'affichant dans la console.
7. Ajouter le script nécessaire pour afficher les tags dans un card

Fichier gallerie-details.page.dart

```
...
class _GallerieDetailsPageState extends State<GallerieDetailsPage> {
  int currentPage = 1;
  int size = 10;
  List<dynamic> hits = [];
...
    String url =
        "https://pixabay.com/api/?key=15646595-
375eb91b3408e352760ee72c8&q=${keyword}&page=${currentPage}&per_page=${size}";
    http.get(Uri.parse(url)).then((resp) {
      setState(() {
        this.galleryData = json.decode(resp.body);
        hits = galleryData['hits'];
        print(hits);
      });
...
        body: (galleryData == null
            ? CircularProgressIndicator()
            : ListView.builder(
                itemCount: (galleryData == null ? 0 : hits.length),
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Card(
                        child: Text(
                          ...,
                          style: TextStyle(
                              fontSize: 22,
```

```
                                color: Colors.white,
                                fontWeight: FontWeight.bold),
                            ),
                            color: Colors.blue,
                        ),
                    ],
                );
            }))));
    }
}
```



8. Améliorer cet affichage en :
    a. Rendant les tags occupant toute la largeur
    b. Ajoutant des padding aux différents éléments des tags
    c. Centrant le texte et en le mettant en gras

9. Utiliser un deuxième card pour afficher les images

Fichier gallerie-details.page.dart

```
...
   Widget build(BuildContext context) {
    return Scaffold(
        appBar: ...,
        body: (galleryData == null
```

```
            ? CircularProgressIndicator()
            : ListView.builder(
                itemCount: (galleryData == null ? 0 : hits.length),
                itemBuilder: (context, index) {
                  return Column(
                    children: [
                      Container(...),
                      Container(
                        child: Card(
                          //Afficher la photo
                          child:...,
                            fit: BoxFit.fitWidth,),
                        ),
                        padding: EdgeInsets.only(left: 10, right: 10),
                      )
...
```

10. Développer le système de pagination :
    a. Utiliser un objet ScrollController,
    b. Calculer le nombre de page
    c. Afficher aussi dans la barre de titre la page courante divisé par le nombre de pages
    d. Libérer le scroll
    e. Centrer CircularProgressIndicator

Fichier gallerie-details.page.dart

```
...
class _GallerieDetailsPageState extends State<GallerieDetailsPage> {
  int currentPage = 1;
  int size = 10;
  late int totalPages;
  ScrollController _scrollController= new ScrollController();
  List<dynamic> hits = [];
  var galleryData;
  @override
  void initState() {
    super.initState();
    getGalleryData(widget.keyword);
    _scrollController.addListener(() {
if(_scrollController.position.pixels==_scrollController.position.maxScrollExte
nt){
        if (currentPage<totalPages) {
          currentPage++;
          getGalleryData(widget.keyword);
```

```
        }
      }
    });
  }

  void getGalleryData(String keyword) {
...
      setState(() {
        this.galleryData = json.decode(resp.body);
        hits.addAll(galleryData['hits']);
        totalPages=(galleryData['totalHits']/size).ceil();
...
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: totalPages==0?Text('Pas de résultats'):
Text("${widget.keyword}, Page ${currentPage}/${totalPages}")
        ),
        body: (galleryData == null
            ? Center(child: CircularProgressIndicator())
            : ListView.builder(
                itemCount: (galleryData == null ? 0 : hits.length),
                controller: _scrollController,
...
  //liberer le scroll
  @override
  void dispose() {
    // TODO: implement dispose
    super.dispose();
    _scrollController.dispose();
  }
}
```