

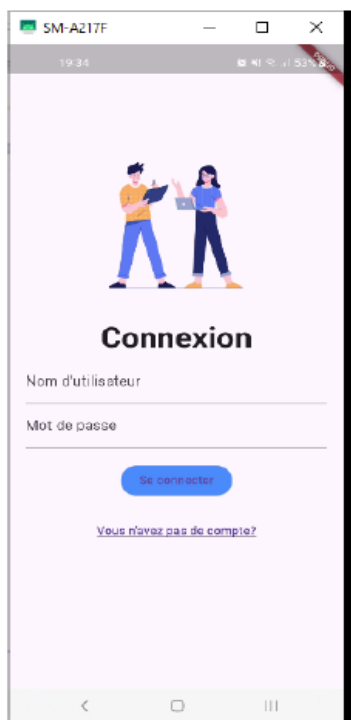
Atelier 5 : Interrogation d'une base de données avec Flutter

Objectif

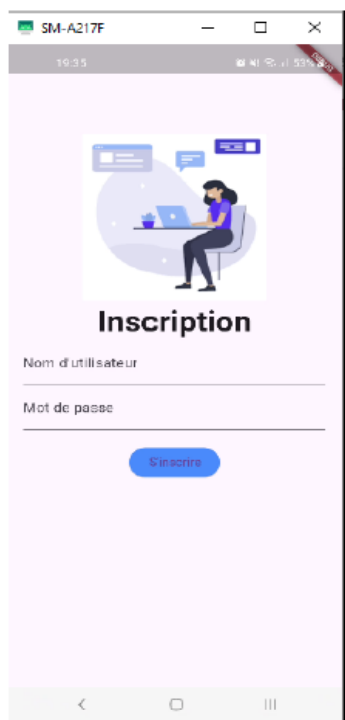
L'objectif de ce TP est de développer une application mobile qui permet à l'utilisateur de s'inscrire, de se connecter, et de gérer une liste d'articles (ajouter, modifier et supprimer).

Les articles sont stockés dans une base de données MySQL et le dialogue avec celle-ci se fait à travers des scripts PHP.

Les différents écrans de l'application sont donnés comme suit :



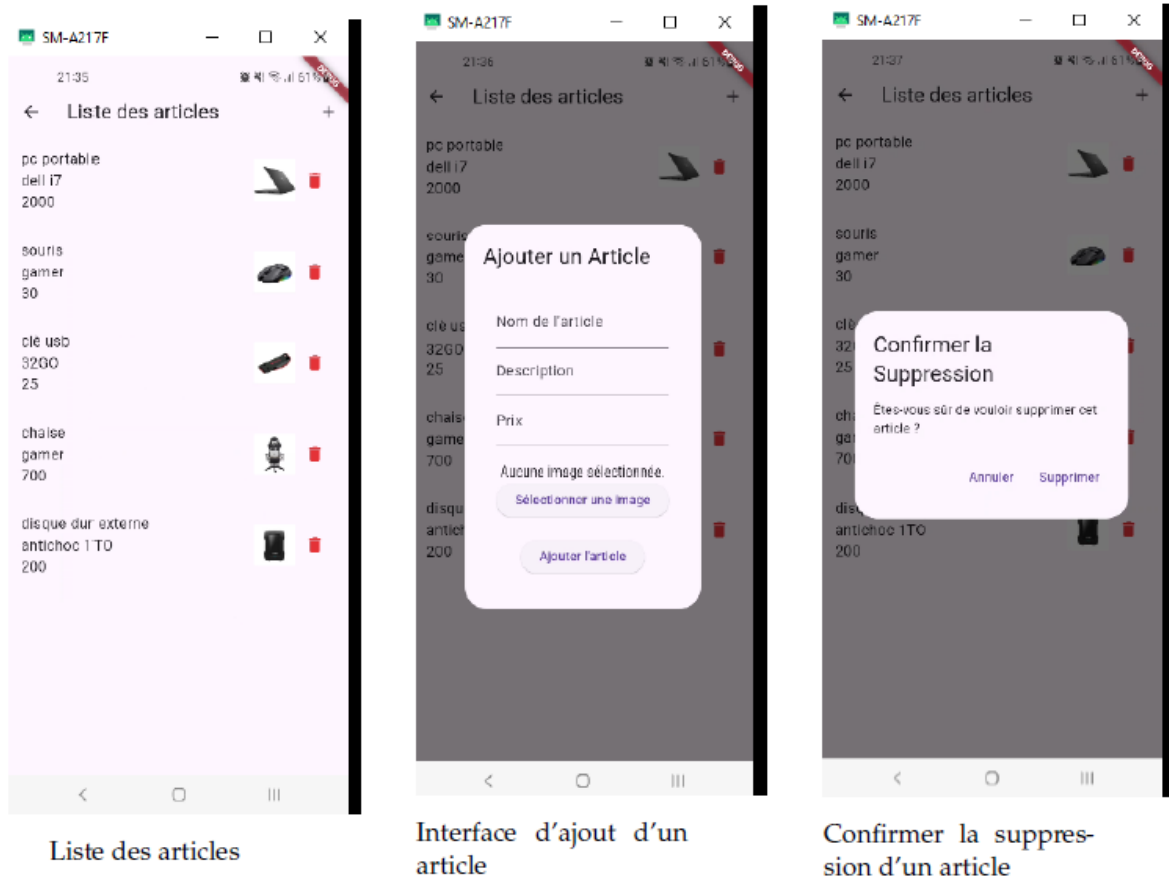
Interface de connexion



Interface d'inscription



Interface d'accueil



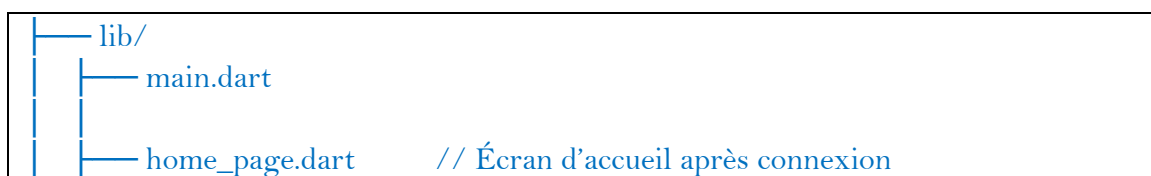
Instructions

1. Sous le répertoire : XAMPP "C:\xampp\htdocs",
 - a. Créer un nouveau dossier "my_store"
 - b. Placer les fichiers de script PHP dans le répertoire créé.
2. Créer la BD en important le fichier my_store.sql



Énoncé

1. Créez un nouveau projet Flutter nommé `mystore`.
Les dossiers lib et assets du projet sont donnés comme suit :



```
| | — list_data.dart    // Page qui affiche la liste des articles
| | — item_detail.dart // Page pour afficher les détails d'un article
| | — login_form.dart  // Page de formulaire de connexion
| — assets/
|   — images/          // images du projet
```

2. Dans le fichier **pubspec.yaml** :

- Faire les modifications nécessaires afin de prendre en considération les images à utiliser dans le projet
- Ajouter les package http(pour envoyer des requêtes vers le serveur xamp) et image_picker ou file_picker(pour choisir des images à partir du galerie) :

```
name: my_store
description: "A new Flutter project."
publish_to: 'none'
version: 0.1.0

environment:
  sdk: ^3.9.2

dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.0
  image_picker: ^0.8.4+8
  file_picker: ^10.3.3

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^5.0.0

flutter:
  uses-material-design: true
  assets:
    - images/
```

3. Terminer le code manquant du fichier **main.dart** suivant :

```
import 'package:flutter/material.dart';
//Terminer les importations nécessaires
```

```

void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title:.....// titre de l'application
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),

      initialRoute:....., // L'application démarre sur la page de connexion
      routes: {
        '/': (context) => .....//route nommée pour la page de connexion
        '/home': (context) => .....//route nommée pour la page d'accueil
        '/dataList': (context) => .....//route nommée pour la liste des articles
        '/itemDetail':(context) => .....//route nommée pour les détails d'un article
      },
    );
  }
}

```

4. Dans un fichier **globals.dart** ajouter la ligne de code permettant de définir une variable globale appelée **baseUrl** qui stocke l'adresse du serveur où se trouve la base de données.

Exemple

```
final String baseUrl = "http:// 192.168.1.10/my_store/";
```

5. Terminer le code manquant du fichier **login_form.dart** suivant :

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:my_store1/globals.dart' as globals;
import 'dart:convert';

class LoginForm extends StatefulWidget {
  const LoginForm({super.key});
  @override
  State<LoginForm> createState() => _LoginFormState();
}

```

```

class _LoginFormState extends State<LoginForm> {
  /* deux controleurs controllerusername et controllerpassword permettant de
  récupérer les textes saisis par l'utilisateur*/

  .....

  Future login() async {
    //Construction de l'URL du script login.php
    var url = Uri.parse("${globals.baseUrl}login.php");
    try {
      var response = await http.post(
        url,
        headers: {'Content-Type': 'application/json'},
        //Envoie un JSON au serveur avec le username et le password saisis

        body: json.encode({
          "username": .....,
          "password": .....
        })),
      );
    //convertir la réponse JSON du serveur en un objet Dart
    var data = .....(response.body);
    /*Affiche un message et se redirige vers /home en passant le nom d'utilisateur comme argument*/
    if (data['success'] == true) {
      .....(
        SnackBar(content: Text("Login reussie !! ")),
      );
      Navigator.pushNamed(context, .....,
        arguments: .....);
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text(data['message'] ?? "Erreur de connexion")),
      );
    }
  }
  //Erreur serveur ou réseau
} catch (e) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text("Serveur non disponible !! ")),
  );
}

}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Form(
      child: SingleChildScrollView(
        child: Padding(

```

```

padding: EdgeInsets.all(20.0),
child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    SizedBox(height: 100),
    //Insérer l'image login.png
    Image.asset(.....),

    Text(
      "Connexion",
      style: TextStyle(fontSize: 35, fontWeight: FontWeight.bold),
    ),
    TextFormField(
      controller: controllerusername,
      decoration: InputDecoration(
        hintText: "Nom d'utilisateur",
        labelText: "Nom d'utilisateur"),
    ),

    TextFormField(
      controller: controllerpassword,
      decoration: InputDecoration(
        hintText: "Mot de passe",
        labelText: "Mot de passe"), // InputDecoration
      obscureText: true,
    ), // TextFormField
    Padding(padding: EdgeInsets.only(bottom: 20.0)),
    // Bouton "Se connecter"
    ElevatedButton(
      onPressed: .....,
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.blueAccent),
      child: Text("Se connecter")), // ElevatedButton

  ],
),
)),
),
);
}
}

```

6. Terminer le code manquant du fichier **home_page.dart** suivant :

```

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {

```

```

//Stocker le nom de l'utilisateur
final String user = ..... ;
return Scaffold(
  appBar: AppBar(title: Text("Page d'accueil")),
  body: Container(
    padding: EdgeInsets.all(20.0),
    child: Column(
      children: [
        Text("Bienvenue ${user}"),
        //Insérer l'image dashboard.png
        Image.asset(.....),
        SizedBox(height: 100),
        ElevatedButton(
          //Se déplacer vers la page '/dataList'
          onPressed: () {
            .....(
              context,
              .....,
            );
          },
          child: Text("Voir les données"),
          style:
            ElevatedButton.styleFrom(backgroundColor: Colors.pink)),
        ElevatedButton(
          //Se déplacer vers la page '/'
          onPressed: () {
            .....(
              context,
              .....,
            );
          },
          child: Text("Se déconnecter"),
          style:
            ElevatedButton.styleFrom(backgroundColor: Colors.blue)),
      ],
    ),
  ));
}

```

7. Terminer le code manquant du fichier **list_data.dart** suivant :

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:my_store/add_item_form.dart';
import 'package:my_store/globals.dart' as globals;

```

```

import 'dart:convert';
class ListData extends StatefulWidget {
  const ListData({super.key});
  @override
  State<ListData> createState() => _ListDataState();
}

class _ListDataState extends State<ListData> {
  /* déclarer une variable _items initialement vide qui va contenir les articles récupérés depuis
  le serveur*/
  .....
  @override
  // initState() est appelée une seule fois lorsque le widget est créé
  //Elle appelle _fetchItems() pour récupérer la liste des articles depuis le serveur.

  void initState() {
    super.initState();
    _fetchItems();
  }

  // fetchItems() récupère les données depuis le serveur dès le lancement de la page
  Future<void> _fetchItems() async {
    //Créer l'URL complète pour accéder au script PHP qui renvoie les articles
    var url = Uri.parse(.....);
    try {
    //Envoyer une requête GET
    var result = await http.get(.....);
    //Décoder la réponse JSON
    var data =.....;
    setState(() {
    //Mettre à jour _items avec les données reçues
    ..... ;
    });
    //En cas d'erreur, afficher un message via SnackBar
  } catch (exp) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text(exp.toString()),
    ));
  }
}

//Afficher une boîte de dialogue pour ajouter un article.
void _showAddItemDialog() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(

```

```

        title: Text('Ajouter un Article'),
        /*AddItemForm est un formulaire permettant de saisir les détails du
nouvel article
Lorsque onItemAdded est appelé après avoir ajouté un nouvel article,
la méthode _fetchItems est appelée pour rafraîchir la liste des articles sur
la page.*/

        content: AddItemForm(onItemAdded: _fetchItems),
    );
},
);
}

//se déplacer vers '/itemDetail' en passant item comme argument
void _navigateToItemDetail(Map<String, dynamic> item) {
    Navigator.pushNamed(
        context,
        '.....',
        arguments: item,
    );
}

//Méthode pour supprimer un article
..... _deleteItem(String itemId) ..... {
    bool resp = ..... showDialog(
        context: context,
        builder: (BuildContext context) {
// Afficher une boîte de dialogue de confirmation
            return AlertDialog(
                title: Text("Suppression"),
                content: Text("Etes-Vous sûr de vouloir supprimer cet article?"),
                actions: [
                    TextButton(
// fermer la boîte de dialogue et renvoyer false
                        onPressed: () => Navigator.of(context).pop(false),
                        child: Text("Annuler")),
                    TextButton(
// fermer la boîte de dialogue et renvoyer true
                        onPressed: () => .....,
                        child: Text("Supprimer"))
                ],
            );
        });
//resp contiendra true ou false selon le choix de l'utilisateur
    if (resp) {
//envoyer la requête de suppression au serveur

```

```

        var url = Uri.parse(.....);
        try {
//envoyer une requête POST au serveur avec le corps contenant l'ID de l'article en JSON et attendre
//sa réponse
            var result = await http.post(url, body: .....);
            var data = json.decode(result.body);
//Si success est true : un SnackBar apparaît pour informer l'utilisateur que l'article a été
//supprimé.
            if (.....) {
                ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(content: Text("Article supprimé !! ")),
                );
//Sinon : affiche le message d'erreur envoyé par le serveur
            } else {
                ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(content: Text(data["message"])),
                );
            }
//Si une erreur survient lors de la requête HTTP ou du décodage JSON, elle est attrapée et un
//SnackBar affiche le message d'erreur.
        } catch (e) {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text(e.toString())),
            );
        }
    }
}

//Méthode pour modifier un article
void _editItem(int itemId) {
    // Récupérer l'article par son ID pour remplir le formulaire avec les
    données existantes
    var itemToEdit = _items.firstWhere((item) => item['id'] == itemId);

    // Afficher la boîte de dialogue avec le formulaire de modification
    showDialog(
        context: context,
        builder: (BuildContext context) {
            return AlertDialog(
                title: Text('Modifier l\'Article'),
                content: editData(
                    item: itemToEdit,
                    onItemUpdated: () {
                        ..... // Recharger les articles après modification
                    },
                ),
            );
        }
    );
}

```

```

    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Liste des articles"),
      actions: [
//bouton + pour ajouter un article
        IconButton(icon: Icon(Icons.add),
Onpressed : _showAddItemDialog,
        ),
      ],
    ),
    body: ListView.builder(
      itemCount: .....,
      itemBuilder: (context, index) {
//Récupérer les données de l'article
        var item = .....;
        var imgurl = .....;
        var full_imgurl = Uri.parse(.....);
        return Padding(
          padding: const EdgeInsets.symmetric(vertical: 4.0, horizontal: 8.0),
          child: ListTile(
//charger l'image de l'article depuis l'URL du serveur
            leading: Container(
              width: 50.0,
              height: 50.0,
              child: Image.network(full_imgurl.toString()),
            ),
//Row : disposition horizontale
            title: Row(
              crossAxisAlignment: CrossAxisAlignment.center,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
//Expanded prend tout l'espace horizontal disponible restant dans la Row(Cela garantit que le texte
//peut s'étendre sur la largeur disponible, et que les boutons restent à droite)
                Expanded(
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
//Nom de l'article
                        .....,
                        style: TextStyle(

```

```

        fontSize: 16,
        fontWeight: FontWeight.bold,
      ),
    ),
    SizedBox(height: 4),
    Text(
//Description de l'article
        .....,
        style: TextStyle(
          fontSize: 14,
          color: Colors.grey[600],
        ),
      ),
    SizedBox(height: 4),
    Text(
//Prix de l'article
        "\${item["price"].toString()}",
        style: TextStyle(
          fontSize: 14,
          fontWeight: FontWeight.w500,
          color: Colors.green,
        ),
      ),
    ],
  ),
),
// Icônes pour suppression et modification
Row(
  children: [
    IconButton(
      onPressed: () => .....,
      icon: Icon(
        Icons.edit,
        color: Colors.greenAccent,
      )),
    IconButton(
      onPressed: ()=>.....,
      icon: Icon(
        Icons.delete,
        color: Colors.redAccent,
      ))
  ],
),

])
),

```

```

onTap: () => _navigateToItemDetail(item),

    );

  }},
);
}
}

```

8. Terminer le code manquant du fichier **add_item_form.dart** suivant :

a. cas 1 : exécution sur mobile

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'dart:io';
import 'package:image_picker/image_picker.dart';
class AddItemForm extends StatefulWidget {
  //onItemAdded est un callback de type VoidCallback qui est passé
  //en tant qu'argument lors de l'instantiation de AddItemForm
  final VoidCallback onItemAdded;

  const AddItemForm({Key? key, required this.onItemAdded}) : super(key: key);

  @override
  _AddItemFormState createState() => _AddItemFormState();
}

class _AddItemFormState extends State<AddItemForm> {
  // 3 contrôleurs servent à récupérer les textes saisis

  final _itemNameController = TextEditingController();
  final _itemDescriptionController = TextEditingController();
  final _itemPriceController = TextEditingController();

  File? _selectedImage; // Stocker l'image sélectionnée
  // permettre à l'utilisateur de choisir une image depuis la galerie du téléphone
  Future<void> _pickImage() async {
    final pickedFile =
      await ImagePicker().pickImage(.....);
    if (.....) {
      setState(() {
        _selectedImage = File(pickedFile.path);
      });
    }
  }
}

```

```

}

Future<void> _addItem() async {
/*La méthode _addItem envoie une requête HTTP au serveur pour ajouter un
nouvel article.Si la requête est réussie, alors la méthode onItemAdded
est appelée. */

if (_selectedImage == null) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(content: Text("Veuillez sélectionner une image.")),
  );
  return;
}

var url = Uri.parse(".....");

try {
  var request = http.MultipartRequest('POST', url);

  // Ajouter les champs texte
  request.fields['item_name'] = .....;
  request.fields['item_description'] = .....;
  request.fields['item_price'] = .....;

  // Ajouter l'image sélectionnée
  request.files.add(await http.MultipartFile.fromPath(
    'item_image', _selectedImage!.path));

  var response = await request.send();
  if (response.statusCode == 200) {
    var responseData = await response.stream.bytesToString();
    var data = jsonDecode(responseData);

    if (.....) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Article ajouté avec succès!")),
      );
      widget.onItemAdded(); // Recharger les articles
      Navigator.of(context).pop(); // Retourner à la liste des articles
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(
            data['message'] ?? "Erreur lors de l'ajout de
l'article.")),
      );
    }
  }
}

```

```

    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content:
            Text("Erreur lors de la requête : ${response.statusCode}")),
        );
    }
  } catch (e) {
    print('Exception: $e');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Erreur réseau ou serveur indisponible.")),
    );
  }
}

@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [
          TextField(
            controller: .....,
            decoration: InputDecoration(labelText: 'Nom de l\'article'),
          ),
          TextField(
            controller: .....,
            decoration: InputDecoration(labelText: 'Description'),
          ),
          TextField(
            controller: .....,
            decoration: InputDecoration(labelText: 'Prix'),
            keyboardType: TextInputType.number,
          ),
          SizedBox(height: 20),
          _selectedImage != null
            ? Image.file(_selectedImage!, height: 100)
            : Text("Aucune image sélectionnée."),
          ElevatedButton(
            onPressed: .....,
            child: Text("Sélectionner une image"),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: .....,

```

```

        child: Text('Ajouter l'article'),
      ),
    ],
  ),
);
}
}

```

b. cas 2 :exécution sur web

```

import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
// import 'dart:io'; // Suppression de l'import dart:io
import 'package:file_picker/file_picker.dart';

// Ajout de l'importation de foundation pour les outils spécifiques à la
plateforme
import 'package:flutter/foundation.dart' show kIsWeb;
import 'package:my_store/globals.dart' as globals;

class AddItemForm extends StatefulWidget {
  final VoidCallback onItemAdded;

  const AddItemForm({super.key, required this.onItemAdded});

  @override
  _AddItemFormState createState() => _AddItemFormState();
}

class _AddItemFormState extends State<AddItemForm> {
  final _itemNameController = TextEditingController();
  final _itemDescriptionController = TextEditingController();
  final _itemPriceController = TextEditingController();

  Uint8List? _imageBytes; // Utilisé pour l'affichage et l'envoi (via
fromBytes)
  String? _fileName; // Pour stocker le nom du fichier pour l'envoi

  Future<void> _pickImage() async {
    // Utiliser FileType.image et ne pas limiter l'extension
    FilePickerResult? result = await FilePicker.platform.pickFiles(

```

```
        type: FileType.image,
        withData:
            true, // IMPORTANT: Garantit que les bytes sont retournés (essentiel
pour le Web)
    );

    if (result != null && result.files.single.bytes != null) {
        setState(() {
            _imageBytes = result.files.single.bytes;
            _fileName = result.files.single.name;
        });
    }
}

Future<void> _addItem() async {
    // VÉRIFICATION N°1 : Image
    if (_imageBytes == null) {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text("Veuillez sélectionner une image.")),
        );
        return;
    }

    // VÉRIFICATION N°2 : Prix
    double? price = double.tryParse(
        _itemPriceController.text.replaceAll(',', '.'),
    ); // Gérer la virgule décimale
    if (price == null) {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text("Le prix saisi n'est pas valide.")),
        );
        return;
    }

    var url = Uri.parse("${globals.baseUrl}add_item.php");

    try {
        var request = http.MultipartRequest('POST', url);

        // Ajouter les champs texte
        request.fields['item_name'] = _itemNameController.text;
        request.fields['item_description'] = _itemDescriptionController.text;
        request.fields['item_price'] = price.toString();

        // --- LOGIQUE D'ENVOI WEB-COMPATIBLE (fromBytes) ---

        // Toujours utiliser fromBytes pour l'environnement Web
    }
}
```

```
request.files.add(
  http.MultipartFile.fromBytes(
    'item_image',
    _imageBytes!, // Utilisez les octets
    filename:
      _fileName ??
      'upload.jpg', // Utiliser le nom de fichier ou un défaut
  ),
);

// --- FIN LOGIQUE D'ENVOI ---

var response = await request.send();
if (response.statusCode == 200) {
  var responseData = await response.stream.bytesToString();
  var data = jsonDecode(responseData);

  if (data['success'] == true) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Article ajouté avec succès!")),
    );
    widget.onItemAdded(); // Recharger les articles
    Navigator.of(context).pop(); // Retourner à la liste des articles
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(
          data['message'] ?? "Erreur lors de l'ajout de l'article.",
        ),
      ),
    );
  }
} else {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Erreur lors de la requête :
${response.statusCode}"),
    ),
  );
}
} catch (e) {
  debugPrint('Exception: $e');
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text("Erreur réseau ou serveur
indisponible.")),
  );
}
```

```

}

@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [
          TextField(
            controller: _itemNameController,
            decoration: const InputDecoration(labelText: 'Nom de
l\'article'),
          ),
          TextField(
            controller: _itemDescriptionController,
            decoration: const InputDecoration(labelText: 'Description'),
          ),
          TextField(
            controller: _itemPriceController,
            decoration: const InputDecoration(labelText: 'Prix'),
            keyboardType: TextInputType.number,
          ),
          const SizedBox(height: 20),
          // Utilisation de Image.memory() qui est compatible Web
          _imageBytes != null
            ? Image.memory(_imageBytes!, height: 100)
            : const Text("Aucune image sélectionnée."),
          ElevatedButton(
            onPressed: _pickImage,
            child: const Text("Sélectionner une image"),
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: _addItem,
            child: const Text('Ajouter l\'article'),
          ),
        ],
      ),
    ),
  );
}

```

9. Terminer le code manquant du fichier **item_detail.dart** suivant :

```

class ItemDetail extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Récupérer l'article passé depuis la page précédente
    final Map<String, dynamic> item =
ModalRoute.of(context)?.settings.arguments as .....;
    // Si item_image est un lien http
    String imageUrl = item['item_image'];
    // Si item_image est juste un nom de fichier, on ajoute l'adresse du serveur local pour obtenir une
    // URL complète.

    if (imageUrl.isNotEmpty && !imageUrl.startsWith('http')) {
      imageUrl = 'http://192.168.1.11/my_store/$imageUrl';
    }
    return Scaffold(
      appBar: AppBar(
        // On affiche le nom de l'article dans la barre d'en-tête
        // Si item['item_name'] est null → on affiche le texte par défaut
        title: Text(item['item_name'] ?? 'Détails de l'article'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Afficher l'image ou un icône par défaut
            item['item_image'] != null && imageUrl.isNotEmpty
              ? .....(
                  imageUrl,
                  fit: BoxFit.cover,
                  height: 200,
                  width: double.infinity,
                // Pendant que l'image se télécharge → afficher un cercle de chargement
                loadingBuilder: (context, child, loadingProgress) {
                  if (loadingProgress == null) {
                    return child;
                  } else {
                    return Center(
                      child: CircularProgressIndicator(
                        value: loadingProgress.expectedTotalBytes != null
                          ? loadingProgress.cumulativeBytesLoaded /
                            (loadingProgress.expectedTotalBytes ?? 1)
                          : null,
                      ),
                    );
                  }
                },
            ],
          ),
        ),
      ),
    );
  }
}

```

```

// Si l'image ne peut pas être chargée
    errorBuilder: (context, error, stackTrace) {
      return Center(
        child: Text('Erreur lors du chargement de l'image'),
      );
    },
  ),
)
// Afficher des informations de l'article
  : Icon(Icons.image, size: 100, color: Colors.grey),
  SizedBox(height: 16.0),
  Text(
    item['item_name'] ?? 'Nom non disponible',
    style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
  ),
  SizedBox(height: 8.0),
  Text(
    ..... ?? 'Description non disponible',
    style: TextStyle(fontSize: 16),
  ),
  SizedBox(height: 8.0),
  Text(
    'Prix: ${..... ?? 'Prix non disponible'}',
    style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
  ),
],
),
),
);
}
}

```

10. Terminer le code manquant du fichier **editData.dart** suivant :

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'listdata.dart';

class editData extends StatefulWidget {
  final Map<String, dynamic> item;
  final VoidCallback onItemUpdated;

```

```

const editData({
  Key? key,
  required this.item,
  required this.onItemUpdated,
}) : super(key: key);

@override
State<editData> createState() => _editDataState();
}

class _editDataState extends State<editData> {
  TextEditingController controllerCode = TextEditingController();
  TextEditingController controllerName = TextEditingController();
  TextEditingController controllerPrice = TextEditingController();
  TextEditingController controllerStock = TextEditingController();

  Future editData() async {
    var url = Uri.parse( .....);
    await http.post(url, body: {
      'id': widget.item['id'].toString(),
      'itemcode': controllerCode.text,
      'itemname': .....
      'price': .....
      'stock': .....
    });
  }

  @override
  void initState() {
    super.initState();
    // Vérification des données nulles et affectation des valeurs par défaut
    controllerCode.text = widget.item['item_code']?.toString() ??
      ''; // Utiliser une chaîne vide si null
    //controllerName.text = widget.item['item_name'] ?? ''; // Si item_name
    est null, utiliser ''
    controllerName.text = widget.item['item_name'];
    controllerPrice.text = widget.item['price']?.toString() ??
      ''; // Si price est null, utiliser ''
    controllerStock.text = widget.item['stock']?.toString() ??
      ''; // Si stock est null, utiliser ''
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Modifier produit")),
      body: Form(

```

```

        child: SingleChildScrollView(
          child: Padding(
            padding: EdgeInsets.all(20.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                TextFormField(
                  controller: controllerName,
                  decoration: InputDecoration(
                    hintText: "Name", //placeholder
                    labelText: "Name", //label
                  ),
                ),
                TextFormField(
                  controller: controllerPrice,
                  decoration: InputDecoration(
                    hintText: "Price", //placeholder
                    labelText: "Price", //label
                  ),
                ),
                TextFormField(
                  controller: controllerStock,
                  decoration: InputDecoration(
                    hintText: "Stock", //placeholder
                    labelText: "Stock", //label
                  ),
                ),
                Padding(padding: EdgeInsets.only(bottom: 20.0)),
                ElevatedButton(
                  onPressed: () {
                    .....;
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) =>.....));
                  },
                  child: Text("Modifier"),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blueAccent)),
                Padding(padding: EdgeInsets.only(bottom: 20.0)),
              ],
            ))));
      },
    );
  }
}

```