TP Listes et grilles : application MaBiblio avec Flutter

TP Listes et grilles : application MaBiblio avec Flutter

1. Objectifs

- Ce TP de **4 heures** clôt le chapitre *Listes et grilles*. En le réalisant, vous allez apprendre à :
  - afficher un élément de liste structuré avec `ListTile` (`leading`, `title`, `subtitle`, `trailing`, `onTap`) ;
  - gérer des choix multiples avec `CheckboxListTile` et `setState`, y compris une case « Tous » à trois états (`tristate`) ;
  - construire des listes statiques et dynamiques avec `ListView`, `ListView.builder` et `ListView.separated`, verticales et horizontales ;
  - mettre en forme un contenu avec `Card` (`elevation`, `shape`, `margin`, `clipBehavior`) ;
  - disposer des éléments en grille avec `GridView.count` et `GridView.extent`;
  - utiliser `GridView.builder` avec un `SliverGridDelegate` pour une grille dynamique et adaptée à la taille de l'écran ;
  - charger des données depuis un fichier **JSON** (`assets`, `jsonDecode`, `fromJson`), les afficher avec `FutureBuilder` et les sauvegarder dans un fichier local (`jsonEncode`, `toJson`) ;
  - versionner son travail avec **Git** et le publier sur **GitHub** dès le départ : dépôt, *commit* à chaque étape, *push*, `README.md` et version finale étiquetée.
- **Prérequis :** Flutter 3.10 ou plus récent, un émulateur ou un téléphone ; connaître `StatelessWidget` / `StatefulWidget`, `setState`, `Scaffold`, `Row` / `Column` et `Navigator.push`.
- **Avant la séance :** un compte **GitHub** gratuit et **Git** installé sur le poste (vérifier avec `git --version`).

1. Description

- Une petite bibliothèque de quartier souhaite une application mobile, **MaBiblio**, pour consulter son catalogue, filtrer les livres par genre et préparer une liste de lecture.
- L'application finale comporte **trois onglets** (`BottomNavigationBar`) et **deux écrans secondaires** ouverts par navigation :

apcpedagogie.com

Écran	Fichier Flutter	Widgets principaux du chapitre
Onglet « Catalogue »	`catalogue_screen.dart`	`ListView.builder`, `ListView` horizontal, `Card`, `ListTile`
Onglet « Galerie »	`galerie_screen.dart`	`GridView.count`, `GridView.extent`, `GridView.builder`, `Card`
Onglet « Ma liste »	`ma_liste_screen.dart`	`CheckboxListTile` (dont `tristate`), `ListView.separated`
Écran « Détail »	`detail_screen.dart`	`Card`, `ListTile`
Écran « Paramètres »	`parametres_screen.dart`	`ListView` statique, `ListTile`

- **Répartition du temps :**

Partie	Widget(s) travaillé(s)	Durée
Partie 00 – Mise en place du projet et du dépôt GitHub	Git, GitHub	20 min
Partie 01 – ListTile	`ListTile`	25 min
Partie 02 – ListView	`ListView`, `.builder`, `.separated`	30 min
Partie 03 – CheckboxListTile	`CheckboxListTile`	30 min
Partie 04 – Card	`Card`	30 min
Partie 05 – GridView	`GridView.count`, `GridView.extent`	25 min
Partie 06 – Utiliser GridView	`GridView.builder` + délégués	30 min
Partie 07 – Gestion des données en JSON	`jsonDecode`, `FutureBuilder`, `File`	30 min
Partie 08 – Intégration finale et version 1.0 sur GitHub	tous + `git tag`	20 min
Total		**4 h**

apcpedagogie.com

1. Travail à réaliser

- **Partie 00Mise en place du projet et du dépôt GitHub**

- **Q0.1.** Créer le projet avec `flutter create ma_biblio` et supprimer le code du compteur par défaut dans `main.dart`.
- **Q0.2.** Organiser le dossier `lib/` selon l'arborescence ci-dessous (à gauche).

- **Q0.3.** Dans `models/livre.dart`, créer la classe `Livre` (à droite). Les champs `lu` et `favori` ne sont pas `final`: ils seront modifiés par l'utilisateur.

```
ma_biblio/
├── .gitignore             (créé par Flutter)
├── README.md              (Partie 08)
├── reponses.md            (au fil du TP)
├── captures/              (Partie 08)
├── assets/
│   └── livres.json        (Partie 07)
lib/
├── models/
│   └── livre.dart
├── data/
│   └── livres_data.dart
├── services/
│   └── livre_service.dart (Partie 07)
├── widgets/
│   ├── livre_card.dart
│   └── livre_grid_item.dart
├── screens/
│   ├── catalogue_screen.dart
│   ├── galerie_screen.dart
│   ├── ma_liste_screen.dart
│   ├── detail_screen.dart
│   └── parametres_screen.dart
└── main.dart
```

```
import 'package:flutter/material.dart';

class Livre {
  final int id;
  final String titre;
  final String auteur;
  final String genre;   // Roman, Science, Histoire, Jeunesse, Informatique
  final int annee;
  final double note;    // sur 5
  final Color couleur;  // couleur de la « couverture »
  final bool nouveaute;
  bool lu;
```

apcpedagogie.com

```
  bool favori;

  Livre({
    required this.id,
    required this.titre,
    required this.auteur,
    required this.genre,
    required this.annee,
    required this.note,
    required this.couleur,
    this.nouveaute = false,
    this.lu = false,
    this.favori = false,
  });
}
```

- **Q0.4.** Dans `data/livres_data.dart`, déclarer la liste `livres` : **au moins 12 livres** répartis sur les 5 genres, dont au moins 4 marqués `nouveaute: true`. Vous pouvez partir des données suivantes :

```
import 'package:flutter/material.dart';
import '../models/livre.dart';

final List<Livre> livres = [
  Livre(id: 1, titre: 'Le Petit Prince', auteur: 'A. de Saint-Exupéry', genre: 'Jeunesse',
annee: 1943, note: 4.8, couleur: Color(0xFF0288D1)),
  Livre(id: 2, titre: "L'Étranger", auteur: 'Albert Camus', genre: 'Roman', annee: 1942, note:
4.3, couleur: Color(0xFFE64A19)),
  Livre(id: 3, titre: 'Une brève histoire du temps', auteur: 'Stephen Hawking', genre:
'Science', annee: 1988, note: 4.5, couleur: Color(0xFF283593), nouveaute: true),
  Livre(id: 4, titre: 'Sapiens', auteur: 'Yuval Noah Harari', genre: 'Histoire', annee: 2011,
note: 4.6, couleur: Color(0xFF6D4C41), nouveaute: true),
  Livre(id: 5, titre: 'Clean Code', auteur: 'Robert C. Martin', genre: 'Informatique', annee:
2008, note: 4.4, couleur: Color(0xFF2E7D32), nouveaute: true),
  Livre(id: 6, titre: 'Le Comte de Monte-Cristo', auteur: 'Alexandre Dumas', genre: 'Roman',
annee: 1844, note: 4.9, couleur: Color(0xFF5E35B1)),
  Livre(id: 7, titre: 'Cosmos', auteur: 'Carl Sagan', genre: 'Science', annee: 1980, note: 4.5,
couleur: Color(0xFF00838F)),
  Livre(id: 8, titre: 'La Muqaddima', auteur: 'Ibn Khaldoun', genre: 'Histoire', annee: 1377,
note: 4.7, couleur: Color(0xFFF9A825)),
```

apcpedagogie.com

```
  Livre(id: 9, titre: "Harry Potter à l'école des sorciers", auteur: 'J. K. Rowling', genre:
'Jeunesse', annee: 1997, note: 4.7, couleur: Color(0xFFC2185B), nouveaute: true),
  Livre(id: 10, titre: 'The Pragmatic Programmer', auteur: 'Hunt & Thomas', genre:
'Informatique', annee: 1999, note: 4.6, couleur: Color(0xFF455A64)),
  Livre(id: 11, titre: 'Les Misérables', auteur: 'Victor Hugo', genre: 'Roman', annee: 1862,
note: 4.8, couleur: Color(0xFFAD1457)),
  Livre(id: 12, titre: 'Le Gène égoïste', auteur: 'Richard Dawkins', genre: 'Science', annee:
1976, note: 4.2, couleur: Color(0xFF558B2F)),
];
```

- **Q0.5.** Dans `main.dart`, créer un `MaterialApp` (titre « MaBiblio », `useMaterial3: true`) dont la page d'accueil est une classe `HomePage` (`StatefulWidget`) contenant pour l'instant un simple `Scaffold` avec une `AppBar`.

Publier le projet sur GitHub dès le départ

- **Q0.6.** Sur **github.com/new**, créer un dépôt nommé `ma-biblio`, **sans** cocher « Add a README », « .gitignore » ni « license » (le projet existe déjà en local). Si le dépôt est privé, inviter l'enseignant comme collaborateur (*Settings* → *Collaborators*).
- **Q0.7.** Configurer Git une seule fois sur le poste avec votre nom et l'adresse e-mail de votre compte GitHub (commandes 1 et 2 ci-dessous).
- **Q0.8.** Ouvrir `.gitignore` à la racine du projet : vérifier qu'il exclut bien `build/` et `.dart_tool/`. Pourquoi ne faut-il jamais envoyer ces dossiers sur GitHub ?
- **Q0.9.** Dans le dossier `ma_biblio`, initialiser le dépôt local, faire le premier *commit* et l'envoyer sur GitHub (commandes 3 à 8). Remplacer `VOTRE-COMPTE` par votre nom d'utilisateur GitHub.

```
# 1-2. Identité (une seule fois par poste)
git config --global user.name "Prénom Nom"
git config --global user.email "adresse@exemple.com"

# 3-8. Premier envoi du projet
cd ma_biblio
git init
git add .
git commit -m "Partie 00 : création du projet MaBiblio"
git branch -M main
git remote add origin https://github.com/VOTRE-COMPTE/ma-biblio.git
git push -u origin main
```

apcpedagogie.com

```
# Vérifications utiles
git status          # fichiers modifiés
git log --oneline   # historique des commits
git remote -v       # adresse du dépôt GitHub
```

- **Q0.10.** Actualiser la page du dépôt sur GitHub et vérifier que les dossiers `lib/`, `android/` et le fichier `pubspec.yaml` apparaissent. Envoyer le lien du dépôt à l'enseignant **avant de commencer la partie 01**.

Authentification : lors du premier `git push`, GitHub n'accepte plus le mot de passe du compte. Connectez-vous par le navigateur quand Git le propose (Git Credential Manager, VS Code ou GitHub Desktop), ou utilisez un *Personal Access Token* (*Settings → Developer settings → Personal access tokens*) à la place du mot de passe.
Règle du TP : un commit et un push à la fin de chaque partie.
Chaque partie se termine par un encadré « Git » qui donne le message de *commit* attendu. L'historique des commits fait partie de la note : un dépôt qui ne contient qu'un seul commit déposé en fin de séance n'obtient aucun point sur le critère Git.
Question de réflexion : pourquoi est-il préférable de faire plusieurs petits commits plutôt qu'un seul gros commit à la fin ?

- **Partie 01Le widget ListTile**

- **Objectif :** afficher un livre sous forme de ligne structurée, puis construire un menu de paramètres.
- **Q1.1.** Dans `screens/parametres_screen.dart`, créer un écran `ParametresScreen` dont le corps est une `Column` contenant **un seul** `ListTile` avec :
  - `leading`: l'icône `Icons.person`;
  - `title`: « Mon profil » ;
  - `subtitle`: « Nom, e-mail, mot de passe » ;
  - `trailing`: l'icône `Icons.chevron_right`.
- **Q1.2.** Ajouter une propriété `onTap` qui affiche un `SnackBar` « Profil sélectionné ». Que se passe-t-il visuellement au clic (effet d'encre) ? Et si vous supprimez `onTap`?
- **Q1.3.** Ajouter trois autres `ListTile`: « Notifications » (`Icons.notifications`), « Langue » (`Icons.language`, subtitle « Français ») et « À propos » (`Icons.info`).
- **Q1.4.** Sur le `ListTile` « À propos », tester successivement `dense: true`, `isThreeLine: true` (avec un subtitle sur deux lignes) et `enabled: false`. Noter en commentaire l'effet de chacune.

apcpedagogie.com

- **Q1.5.** Personnaliser le `ListTile` « Notifications » avec `tileColor`, `iconColor`, `textColor` et une forme arrondie : `shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12))`.
- **Q1.6.** Dans `HomePage`, ajouter dans l'`AppBar` un `IconButton` (`Icons.settings`) qui ouvre `ParametresScreen` avec `Navigator.push`.
- **Q1.7.** Créer une fonction `Widget livreTile(Livre livre)` qui retourne un `ListTile` représentant un livre :
  - `leading`: un `CircleAvatar` de couleur `livre.couleur` contenant la première lettre du titre ;
  - `title`: le titre ; `subtitle`: « auteur – année » ;
  - `trailing`: la note suivie d'une icône `Icons.star` jaune.
  - Afficher temporairement les 3 premiers livres dans le corps de `HomePage` pour vérifier le rendu.

Question de réflexion :
Quelle est la différence entre `leading` et `trailing`? Pourquoi un `ListTile` doit-il être placé dans un parent à largeur contrainte ?
Git – fin de la partie 01 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 01 : ListTile — écran Paramètres et livreTile()"
git push
```

- **Partie 02 Le widget ListView**

- **Objectif :** rendre les listes défilables, statiques puis dynamiques, verticales puis horizontales.
- **Q2.1.** Lancer l'écran Paramètres en mode paysage ou ajouter 10 `ListTile` supplémentaires. Quel message d'erreur apparaît (bande jaune et noire) ? Expliquer pourquoi.
- **Q2.2.** Corriger le problème en remplaçant la `Column` de `ParametresScreen` par un `ListView` (constructeur par défaut avec `children`). Ajouter un `padding` de 8 pixels.
- **Q2.3.** Insérer des `Divider()` entre les `ListTile`, puis remplacer cette solution par `ListTile.divideTiles(context: context, tiles: [...]).toList()`. Quel est l'avantage ?
- **Q2.4.** Dans `screens/catalogue_screen.dart`, créer `CatalogueScreen` qui affiche **tous** les livres avec `ListView.builder` (`itemCount`, `itemBuilder`) en réutilisant `livreTile()`.
- **Q2.5.** Expliquer en commentaire pourquoi `ListView.builder` est préférable à `ListView(children: …)` pour une liste de 1 000 livres.

apcpedagogie.com

- **Q2.6.** Remplacer `ListView.builder` par `ListView.separated` avec un `separatorBuilder` qui affiche un `Divider` indenté (`indent: 72`).
- **Q2.7.** Au-dessus de la liste, ajouter une section « Nouveautés » : un **ListView horizontal** (`scrollDirection: Axis.horizontal`) de 160 px de haut qui n'affiche que les livres où `nouveaute == true`. Chaque élément est un `Container` de 110 px de large, de la couleur du livre, avec le titre centré en blanc.
- **Q2.8.** Organiser `CatalogueScreen` avec une `Column`: titre « Nouveautés », le ListView horizontal (dans un `SizedBox`), titre « Tout le catalogue », puis le ListView vertical. Quelle erreur obtenez-vous si le ListView vertical n'est pas dans un `Expanded`? La corriger.
- **Q2.9 (approfondissement).** Tester `shrinkWrap: true` avec `physics: NeverScrollableScrollPhysics()` comme alternative à `Expanded`. Dans quel cas l'utiliseriez-vous ?

Git – fin de la partie 02 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 02 : ListView — catalogue, separated et nouveautés"
git push
```

- **Partie 03 Le widget CheckboxListTile**

- **Objectif :** permettre à l'utilisateur de cocher des choix et réagir à leur état.
- **Q3.1.** Dans `screens/ma_liste_screen.dart`, créer `MaListeScreen` en **StatefulWidget**. Pourquoi un `StatelessWidget` ne suffit-il pas ici ?
- **Q3.2.** Ajouter un seul `CheckboxListTile` « Afficher uniquement les nouveautés » lié à une variable `bool _nouveautesSeulement = false`. Implémenter `onChanged` avec `setState`. Que se passe-t-il si vous oubliez `setState`?
- **Q3.3.** Créer une `Map<String, bool> _genres` contenant les 5 genres (tous à `true` au départ) et les afficher avec des `CheckboxListTile` générés à partir de la map (`_genres.keys.map(...)`).
- **Q3.4.** Pour chaque `CheckboxListTile` de genre, ajouter :
  - `secondary`: une icône adaptée au genre (ex. `Icons.science` pour Science, `Icons.computer` pour Informatique) ;
  - `subtitle`: le nombre de livres de ce genre, calculé à partir de la liste `livres`;
  - `activeColor`: une couleur de votre choix.
- **Q3.5.** Tester `controlAffinity` avec `ListTileControlAffinity.leading`, `.trailing` et `.platform`. Laquelle place la case à gauche ? La conserver.

apcpedagogie.com

- **Q3.6.** Ajouter en tête de liste un `CheckboxListTile` « Tous les genres » avec `tristate: true`: coché si tous les genres le sont, décoché si aucun, `null` (tiret) sinon. Un clic dessus coche ou décoche tous les genres.
- **Q3.7.** Sous les filtres, afficher avec `ListView.separated` la liste des livres **filtrée** par les genres cochés. Chaque livre est un `CheckboxListTile` dont la valeur est `livre.lu`; le titre est barré si le livre est lu (`TextDecoration.lineThrough`).
- **Q3.8.** Dans l'`AppBar` de `MaListeScreen`, afficher dynamiquement « Lus : X / Y » (X = livres cochés, Y = livres affichés).

Git – fin de la partie 03 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 03 : CheckboxListTile — filtres par genre et livres lus"
git push
```

- **Partie 04Le widget Card**

- **Objectif :** présenter chaque livre dans une carte réutilisable, puis construire l'écran de détail.
- **Q4.1.** Dans `widgets/livre_card.dart`, créer un `StatelessWidget` `LivreCard` qui reçoit un `Livre` et retourne une `Card` contenant le `livreTile()` de la Q1.7.
- **Q4.2.** Régler la `Card`: `elevation: 4`, `margin: EdgeInsets.symmetric(horizontal: 12, vertical: 6)` et une forme `RoundedRectangleBorder` de rayon 16. Comparer visuellement `elevation` à 0, 4 et 12.
- **Q4.3.** Ajouter `shadowColor` et `color` (ou `surfaceTintColor` en Material 3). Quelle différence observez-vous entre `color` et `surfaceTintColor`?
- **Q4.4.** Dans `CatalogueScreen`, remplacer `livreTile()` par `LivreCard` dans le ListView vertical. Le séparateur devient inutile : quel constructeur de `ListView` reprenez-vous ?
- **Q4.5.** Rendre la carte cliquable avec un `InkWell` (ou le `onTap` du `ListTile`). Pourquoi faut-il `clipBehavior: Clip.antiAlias` sur la `Card` pour que l'effet d'encre respecte les coins arrondis ?
- **Q4.6.** Créer `DetailScreen` qui reçoit un `Livre` et affiche une **Card** contenant une `Column`:
  - en haut, un `Container` de 200 px de haut, de la couleur du livre, avec le titre en grand ;
  - puis trois `ListTile`: Auteur (`Icons.person`), Année (`Icons.calendar_today`), Genre (`Icons.category`) ;
  - en bas, une rangée de boutons (`OverflowBar`) : « Ajouter aux favoris » et « Marquer comme lu ».

apcpedagogie.com

- **Q4.7.** Au clic sur une `LivreCard` du catalogue, ouvrir `DetailScreen` avec `Navigator.push` en passant le livre sélectionné.
- **Q4.8.** Dans `DetailScreen`, le bouton « Ajouter aux favoris » bascule `livre.favori` et change l'icône (`Icons.favorite` / `Icons.favorite_border`). De quel type doit être `DetailScreen`?

Git – fin de la partie 04 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 04 : Card – LivreCard et écran Détail"
git push
```

- **Partie 05 Le widget GridView**

- **Objectif :** découvrir la disposition en grille avec les constructeurs simples.
- **Q5.1.** Dans `screens/galerie_screen.dart`, créer `GalerieScreen`. Afficher une grille de **6 conteneurs colorés numérotés de 1 à 6** avec `GridView.count` et `crossAxisCount: 2`.
- **Q5.2.** Ajouter `mainAxisSpacing: 10`, `crossAxisSpacing: 10` et `padding: EdgeInsets.all(10)`. Expliquer la différence entre l'axe principal (*main axis*) et l'axe secondaire (*cross axis*) d'une grille verticale.
- **Q5.3.** Passer `crossAxisCount` à 3 puis à 4, puis modifier `childAspectRatio` (0.7, 1.0, 1.5). Quel ratio donne des cases au format « couverture de livre » (plus hautes que larges) ?
- **Q5.4.** Ajouter `scrollDirection: Axis.horizontal`. Comment se comporte maintenant `crossAxisCount`? Remettre ensuite la direction verticale.
- **Q5.5.** Remplacer les 6 conteneurs par les livres en générant les enfants avec `livres.map((l) => ...).toList()`. Chaque case est une `Card` (couleur du livre, coins arrondis) contenant une `Column` centrée : icône `Icons.menu_book` blanche, titre (2 lignes max, `overflow: TextOverflow.ellipsis`) et auteur.
- **Q5.6.** Remplacer `GridView.count` par `GridView.extent` avec `maxCrossAxisExtent: 150`. Tourner l'émulateur ou redimensionner la fenêtre (web / desktop). Qu'observez-vous sur le nombre de colonnes ? Quelle est la différence fondamentale avec `GridView.count`?

Question de réflexion :
Comme pour `ListView(children: …)`, quel est l'inconvénient de `GridView.count` et `GridView.extent` lorsque le nombre d'éléments est très grand ?

apcpedagogie.com

Git – fin de la partie 05 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 05 : GridView.count et GridView.extent"
git push
```

- **Partie 06Utiliser GridView dans une vraie interface**

- **Objectif :** construire une grille dynamique, performante et adaptée à la taille de l'écran.
- **Q6.1.** Réécrire la galerie avec `GridView.builder`: `itemCount: livres.length` et un `gridDelegate` de type `SliverGridDelegateWithFixedCrossAxisCount` (`crossAxisCount: 2`, `childAspectRatio: 0.7`, espacements de 12).
- **Q6.2.** Extraire le contenu d'une case dans un widget `LivreGridItem` (`widgets/livre_grid_item.dart`). Ajouter en haut à droite de la carte une icône cœur si `livre.favori` est vrai (`Stack` + `Positioned`).
- **Q6.3.** Remplacer le délégué par `SliverGridDelegateWithMaxCrossAxisExtent` (`maxCrossAxisExtent: 180`). Quel constructeur de la partie 05 ce délégué reproduit-il ?
- **Q6.4.** Rendre la grille **responsive**: avec `MediaQuery.of(context).size.width`, calculer `crossAxisCount` (2 colonnes en dessous de 600 px, 3 entre 600 et 900 px, 4 au-delà). Tester en mode paysage.
- **Q6.5.** Au clic sur une case (`InkWell` ou `GestureDetector`), ouvrir le `DetailScreen` de la partie 04 avec le livre correspondant.
- **Q6.6.** Ajouter dans l'`AppBar` de `GalerieScreen` un `IconButton` qui bascule entre l'affichage **grille** (`GridView.builder`) et l'affichage **liste** (`ListView.builder` de `LivreCard`). L'icône alterne entre `Icons.grid_view` et `Icons.view_list`.
- **Q6.7.** Ajouter un `IconButton` de tri par note décroissante (`livres.sort(...)`). Vérifier que la grille se met à jour.

Question de réflexion :

Compléter le tableau comparatif suivant dans `reponses.md`.

Constructeur	Nombre de colonnes (fixe / adaptatif)	Génération des éléments (tous / à la demande)	Cas d'usage conseillé
`GridView.count`	…	…	…
`GridView.extent`	…	…	…

apcpedagogie.com

Constructeur	Nombre de colonnes (fixe / adaptatif)	Génération des éléments (tous / à la demande)	Cas d'usage conseillé
`GridView.builder`	…	…	…

Git – fin de la partie 06 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 06 : GridView.builder responsive, bascule et tri"
git push
```

- **Partie 07 Gestion des données en JSON**

- **Objectif :** remplacer la liste écrite « en dur » dans `livres_data.dart` par un fichier JSON, puis sauvegarder les choix de l'utilisateur (livres lus, favoris) pour les retrouver au prochain lancement.

- **Lecture d'un fichier JSON fourni avec l'application**

- **Q7.1.** Créer le dossier `assets/` à la racine du projet et le fichier `assets/livres.json` (modèle ci-dessous). Reprendre les 12 livres de la partie 00. La couleur est stockée sous forme de texte hexadécimal.
- **Q7.2.** Déclarer le fichier dans `pubspec.yaml` (section `flutter:`), puis lancer `flutter pub get`. Que se passe-t-il si l'indentation de `assets:` est incorrecte ?

```
[
  {
    "id": 1,
    "titre": "Le Petit Prince",
    "auteur": "A. de Saint-Exupéry",
    "genre": "Jeunesse",
    "annee": 1943,
    "note": 4.8,
    "couleur": "#0288D1",
    "nouveaute": false,
    "lu": false,
    "favori": false
```

apcpedagogie.com

```
  },
  {
    "id": 2,
    "titre": "L'Étranger",
    "auteur": "Albert Camus",
    "genre": "Roman",
    "annee": 1942,
    "note": 4.3,
    "couleur": "#E64A19",
    "nouveaute": false,
    "lu": false,
    "favori": false
  }
]
```

```
# pubspec.yaml
flutter:
  uses-material-design: true
  assets:
    - assets/livres.json
```

- **Q7.3.** Dans la classe `Livre`, ajouter :
  - un constructeur `factory Livre.fromJson(Map<String, dynamic> json)` qui convertit une entrée JSON en objet `Livre`; la couleur `"#0288D1"` devient `Color(0xFF0288D1)`;
  - une méthode `Map<String, dynamic> toJson()` qui fait l'opération inverse ;
  - attention : la note `4` écrite sans décimale est lue comme un `int`; utiliser `(json['note'] as num).toDouble()`.
- **Q7.4.** Créer `services/livre_service.dart` avec une fonction `Future<List<Livre>> chargerLivres()` qui lit le fichier avec `rootBundle.loadString('assets/livres.json')`, le décode avec `jsonDecode` (`dart:convert`) et retourne la liste de `Livre`. Pourquoi cette fonction est-elle `async`?
- **Q7.5.** Dans `HomePage`, charger les livres une seule fois dans `initState` (variable `late Future<List<Livre>> _futurLivres`) et afficher le contenu avec un `FutureBuilder`:
  - pendant le chargement : un `CircularProgressIndicator` centré ;
  - en cas d'erreur (testez en renommant le fichier) : un message rouge avec `snapshot.error`;
  - une fois les données reçues : les écrans des parties précédentes, qui reçoivent désormais la liste en paramètre au lieu d'utiliser la variable globale `livres`.
- **Q7.6.** Ajouter un livre directement dans `livres.json` (sans toucher au code Dart), relancer l'application (*hot restart*) et vérifier qu'il apparaît dans le Catalogue et la Galerie. Quel

apcpedagogie.com

avantage voyez-vous par rapport à `livres_data.dart`?

Sauvegarde des choix de l'utilisateur dans un fichier local

- **Q7.7.** Ajouter le package `path_provider` (`flutter pub add path_provider`). Dans `livre_service.dart`, écrire `Future<void> sauvegarderLivres(List<Livre> livres)` qui :
  - récupère le dossier de l'application avec `getApplicationDocumentsDirectory()`;
  - écrit dans le fichier `mabiblio.json` le résultat de `jsonEncode(livres.map((l) => l.toJson()).toList())` avec `File(...).writeAsString(...)` (`dart:io`).
- **Q7.8.** Modifier `chargerLivres()`: si le fichier `mabiblio.json` existe (`await fichier.exists()`), le lire en priorité ; sinon, lire `assets/livres.json`.
- **Q7.9.** Appeler `sauvegarderLivres()` à chaque fois qu'un livre est coché « lu » (partie 03) ou ajouté aux favoris (partie 04). Fermer complètement l'application, la relancer et vérifier que les choix sont conservés.
- **Q7.10.** Dans `ParametresScreen`, ajouter un `ListTile` « Réinitialiser ma bibliothèque » (`Icons.restart_alt`) qui supprime `mabiblio.json` après confirmation (`showDialog` + `AlertDialog`) et recharge les données d'origine.

Questions de réflexion :

1) Pourquoi ne peut-on pas écrire directement dans `assets/livres.json` depuis l'application ?

2) Quelle est la différence entre `jsonDecode` / `jsonEncode` et `fromJson` / `toJson`?

3) `path_provider` et `dart:io` ne fonctionnent pas sur Flutter Web : quelle alternative proposeriez-vous (indice : `shared_preferences`) ?

Git – fin de la partie 07 : enregistrer et publier votre travail, puis vérifier le nouveau commit sur GitHub.

```
git add .
git commit -m "Partie 07 : chargement et sauvegarde des livres en JSON"
git push
```

Partie 08 Intégration finale

- **Objectif :** assembler tous les écrans dans une application cohérente.
- **Q8.1.** Dans `HomePage`, ajouter une `BottomNavigationBar` à trois onglets : « Catalogue » (`Icons.list`), « Galerie » (`Icons.grid_on`), « Ma liste » (`Icons.checklist`). Gérer l'onglet actif avec une variable `int _index` et `setState`.

- **Q8.2.** Le corps du `Scaffold` affiche l'écran correspondant à l'onglet sélectionné (liste `List<Widget> _ecrans`), à l'intérieur du `FutureBuilder` de la partie 07.
- **Q8.3.** Vérifier la cohérence des données : un livre marqué « lu » dans Ma liste ou ajouté aux favoris dans le Détail doit apparaître comme tel dans le Catalogue et la Galerie au retour, et après un redémarrage de l'application. Sinon, expliquer pourquoi et proposer une correction (par exemple `Navigator.push(...).then((_) => setState(() {}))`).
- **Q8.4.** Tester l'application complète sur Android ou desktop et faire une capture de chaque écran : Catalogue, Galerie (grille et liste), Ma liste, Détail, Paramètres. Les enregistrer dans le dossier `captures/` du projet.

Finaliser le dépôt GitHub

- **Q8.5.** Créer à la racine du projet un fichier `README.md` qui présente l'application (modèle ci-dessous) : description, fonctionnalités, captures d'écran et commandes pour lancer le projet.
- **Q8.6.** Vérifier que `reponses.md` est complet et présent à la racine du dépôt.
- **Q8.7.** Faire le dernier commit, puis marquer la version finale avec une étiquette `v1.0` et l'envoyer sur GitHub (commandes ci-dessous). Vérifier sur GitHub que le README s'affiche avec les captures et que l'étiquette apparaît dans *Tags*.

```
# MaBiblio

Application Flutter de catalogue de livres réalisée
dans le cadre du TP « Listes et grilles ».

## Fonctionnalités
- Catalogue en liste (ListView, Card, ListTile)
- Galerie en grille responsive (GridView.builder)
- Filtres par genre et livres lus (CheckboxListTile)
- Données chargées et sauvegardées en JSON

## Captures d'écran
![Catalogue](captures/catalogue.png)
![Galerie](captures/galerie.png)

## Lancer le projet
flutter pub get
flutter run

## Auteur
```

apcpedagogie.com

```
Prénom Nom — Groupe
```

```
git add .
git commit -m "Partie 08 : intégration finale, README et captures"
git push

git tag v1.0
git push origin v1.0
```

1. Exercices complémentaires (bonus)

- **Recherche :** ajouter un `TextField` au-dessus du catalogue qui filtre la liste par titre ou auteur.
- **Grille repliable :** remplacer la grille par un `SliverGrid` dans un `CustomScrollView` avec une `SliverAppBar` repliable.
- **Suppression par glissement :** ajouter un `Dismissible` sur les livres de Ma liste pour retirer un livre en le faisant glisser.
- **Intégration continue :** ajouter un fichier `.github/workflows/flutter.yml` qui lance `flutter analyze` à chaque `git push` (GitHub Actions), et afficher le badge de statut dans le `README.md`.

1. Compte rendu

- **À rendre :**
  - **le lien de votre dépôt GitHub** `ma-biblio`, envoyé dès la partie 00 : c'est ce dépôt qui est corrigé, dans l'état de l'étiquette `v1.0`;
  - dans le dépôt, le fichier `reponses.md`: réponses aux questions Q0.8, Q1.2, Q1.4, Q2.1, Q2.5, Q2.8, Q3.1, Q3.2, Q4.3, Q4.5, Q5.2, Q5.3, Q5.6, Q6.3, Q7.2, Q7.4, Q7.6, aux questions de réflexion et le tableau comparatif ;
  - dans le dépôt, le fichier `assets/livres.json` complet ;
  - dans le dépôt, le `README.md` et les captures d'écran du dossier `captures/`.
- **Barème indicatif (/20) :**

Critère	Points
Partie 00 – mise en place, arborescence et modèle de données	1
Partie 01 – `ListTile` (écran Paramètres, `livreTile()`)	2

apcpedagogie.com

Critère	Points
Partie 02 – `ListView`, `.builder`, `.separated`, liste horizontale	2,5
Partie 03 – `CheckboxListTile`, filtres et case « Tous » à 3 états	2,5
Partie 04 – `Card`, `LivreCard` et écran Détail	2,5
Partie 05 – `GridView.count` et `GridView.extent`	2
Partie 06 `GridView.builder` responsive, bascule grille / liste, tri	2,5
Partie 07 – lecture JSON (`fromJson`, `FutureBuilder`) et sauvegarde locale (`toJson`, `File`)	2,5
Partie 08 – intégration finale et cohérence des données	1
Git / GitHub – dépôt créé en partie 00, un commit par partie avec un message clair, `README.md`, étiquette `v1.0`	1,5
Total	**20**
Bonus	+2 max

Question de réflexion (à rédiger en 5 lignes) :

Pourquoi parle-t-on de « construction à la demande » (*lazy loading*) pour `ListView.builder` et `GridView.builder`, et quel est son intérêt sur un téléphone d'entrée de gamme ?

Source : https://apcpedagogie.com/tp-listes-et-grilles-application-mabiblio-avec-flutter · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com