TP3 - Application multifenêtres et gestion des évènements

el mehdi aboulouafa
25 novembre 2025

1. Introduction

Ce TP porte sur la création d'une application Flutter de gestion de tâches (Todo List). Nous allons apprendre à structurer un projet, créer des interfaces utilisateur et gérer les événements.

2. Étape 1 : Création du projet

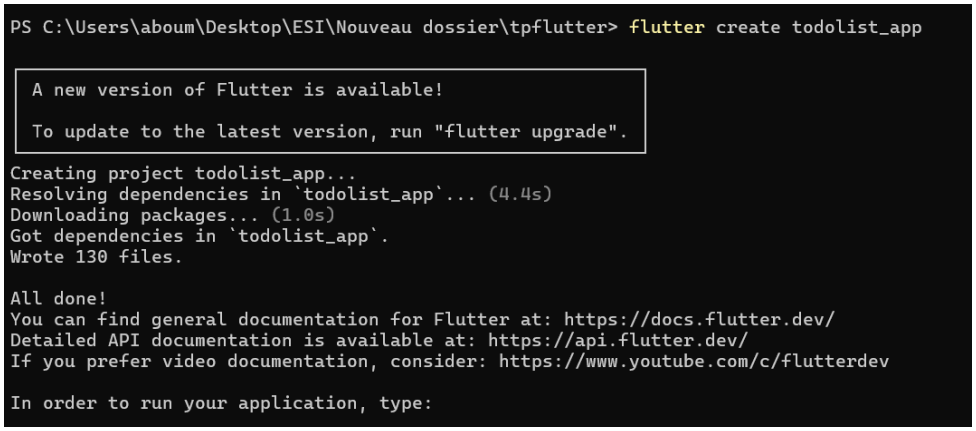

FIGURE 1 – a commande flutter create

Questions et Réponses

Quelle commande crée un projet Flutter ?
Réponse : `flutter create todolist_app`
Cette commande génère la structure de base.

Pourquoi initialiser Git ?
Réponse : Pour versionner le code et suivre les modifications.
Git permet de sauvegarder l'avancement.

```
PS C:\Users\aboum\Desktop\ESI\Nouveau dossier\tpflutter\todolist_app> git init
Initialized empty Git repository in C:/Users/aboum/Desktop/ESI/Nouveau dossier/tpflutter/todolist_app/.git/
PS C:\Users\aboum\Desktop\ESI\Nouveau dossier\tpflutter\todolist_app> git add .
warning: in the working copy of '.metadata', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'linux/flutter/generated_plugin_registrant.cc', LF will be replaced by CRLF the next tim
e Git touches it
warning: in the working copy of 'linux/flutter/generated_plugin_registrant.h', LF will be replaced by CRLF the next time
 Git touches it
warning: in the working copy of 'linux/flutter/generated_plugins.cmake', LF will be replaced by CRLF the next time Git t
ouches it
warning: in the working copy of 'macos/Flutter/GeneratedPluginRegistrant.swift', LF will be replaced by CRLF the next ti
me Git touches it
warning: in the working copy of 'pubspec.lock', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'windows/flutter/generated_plugin_registrant.cc', LF will be replaced by CRLF the next t
ime Git touches it
warning: in the working copy of 'windows/flutter/generated_plugin_registrant.h', LF will be replaced by CRLF the next ti
me Git touches it
warning: in the working copy of 'windows/flutter/generated_plugins.cmake', LF will be replaced by CRLF the next time Git
 touches it
PS C:\Users\aboum\Desktop\ESI\Nouveau dossier\tpflutter\todolist_app> git comiit -m "save adv mehdi"
git: 'comiit' is not a git command. See 'git --help'.

The most similar command is
        commit
PS C:\Users\aboum\Desktop\ESI\Nouveau dossier\tpflutter\todolist_app>
```

FIGURE 2 – git commands

3. Étape 2 : Fonction main() et MaterialApp

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

void main() {
  runApp(
    const MaterialApp(
      home: Tasks(),
    ),
  );
}
```

FIGURE 3 – Code de la fonction main() dans le fichier main.dart

```dart
void main() {
  runApp(const MaterialApp(home: Tasks()));
}
```

Questions et Réponses

Quel est le rôle de main() ?
Réponse : Point d'entrée de l'application Dart.
Première fonction exécutée.

Que fait runApp() ?
Réponse : Initialise l'application Flutter.
Attache le widget racine à l'écran.

4. Étape 3 : Création du widget Tasks

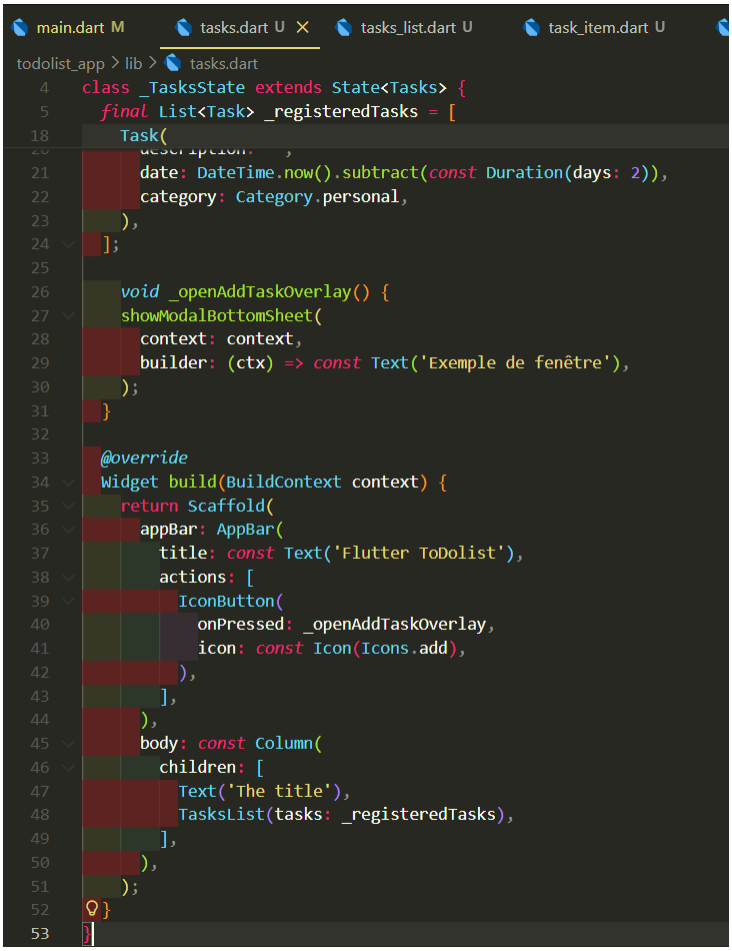

FIGURE 4 – Architecture d'un StatefulWidget avec sa classe d'état

Questions et Réponses

Pourquoi StatefulWidget ?
Réponse : Pour gérer les changements d'état.
La liste des tâches évolue.

Que fait createState() ?
Réponse : Crée l'état associé au widget.
Retourne une instance de la classe d'état.

5. Étape 4 : Création du modèle Task

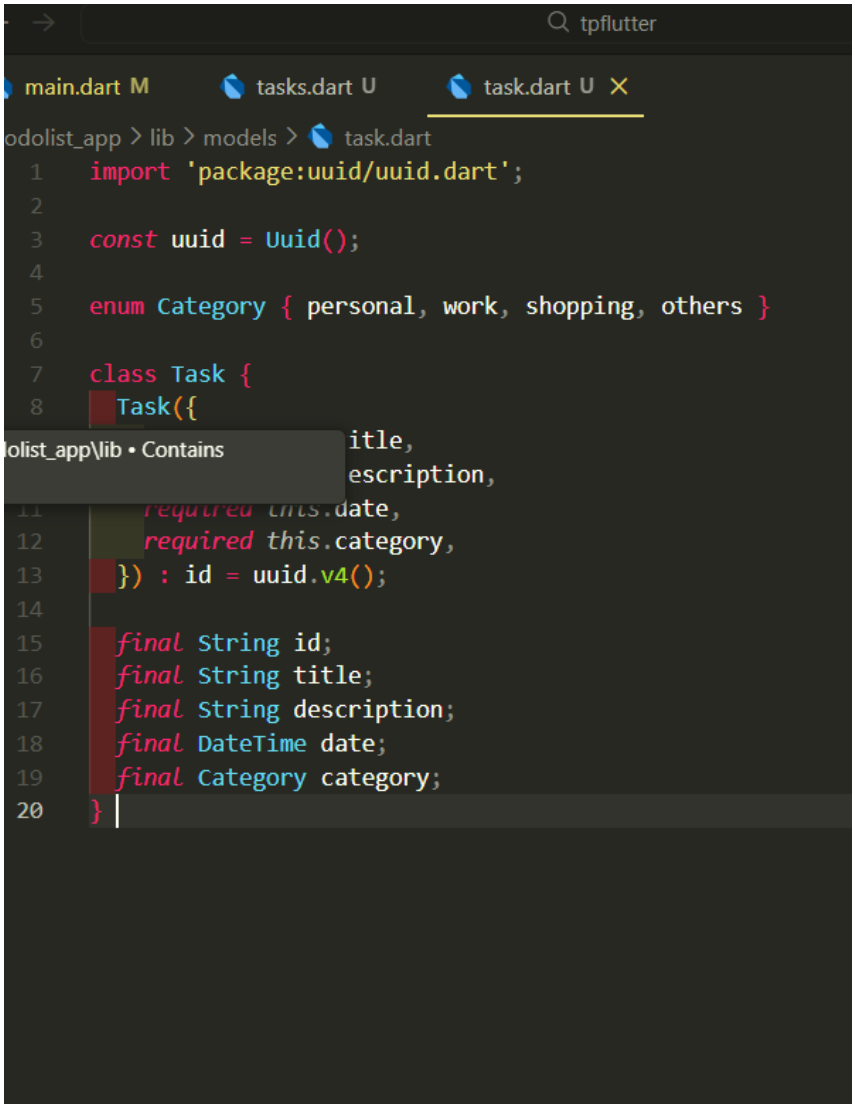

FIGURE 5 – Structure de la classe Task avec ses propriétés finales

```dart
class Task {
  final String id;
  final String title;
  final String description;
  final DateTime date;
  final Category category;
}
```

Questions et Réponses

Que fait flutter pub add uuid ?
Réponse : Ajoute le package uuid au projet.
Pour générer des identifiants uniques.

6. Étape 5 : Création de TasksList

main.dart M tasks.dart U tasks_list.dart U task_item.dart U

todolist_app > lib > widgets > tasks_list.dart

```dart
import 'package:flutter/material.dart';
import 'package:todolist_app/models/task.dart';
import 'package:todolist_app/widgets/task_item.dart';

class TasksList extends StatelessWidget {
  const TasksList({
    super.key,
    required this.tasks,
  });

  final List<Task> tasks;

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: tasks.length,
      itemBuilder: (ctx, index) => TaskItem(tasks[index]),
    );
  }
}
```

FIGURE 6 – Utilisation de ListView.builder pour les listes dynamiques

Questions et Réponses

Pourquoi StatelessWidget ?
Réponse : Car elle affiche des données fixes.
Ses propriétés ne changent pas.

Que signifie => en Dart ?
Réponse : Syntaxe pour fonctions fléchées.
Retourne une expression directement.

7. Étape 6 : Organisation des dossiers

Questions et Réponses

Pourquoi séparer models et widgets ?
Réponse : Pour une architecture claire.
Sépare données et interface.

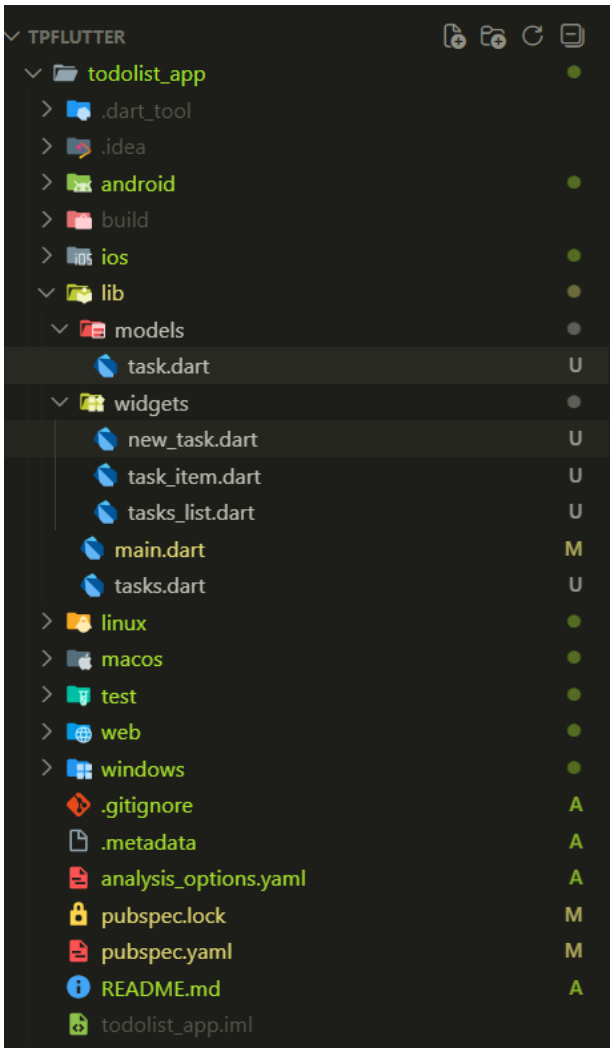

FIGURE 7 – Structure organisée des dossiers models et widgets

8. Étape 7 : Création de TaskItem

```
class TaskItem extends StatelessWidget {
  const TaskItem(this.task, {super.key});
  final Task task;

  @override
  Widget build(BuildContext context) {
    return Card(child: Text(task.title));
  }
}
```

Questions et Réponses

À quoi sert le widget Card ?
Réponse : Fournit un conteneur avec elevation.
Effet visuel Material Design.

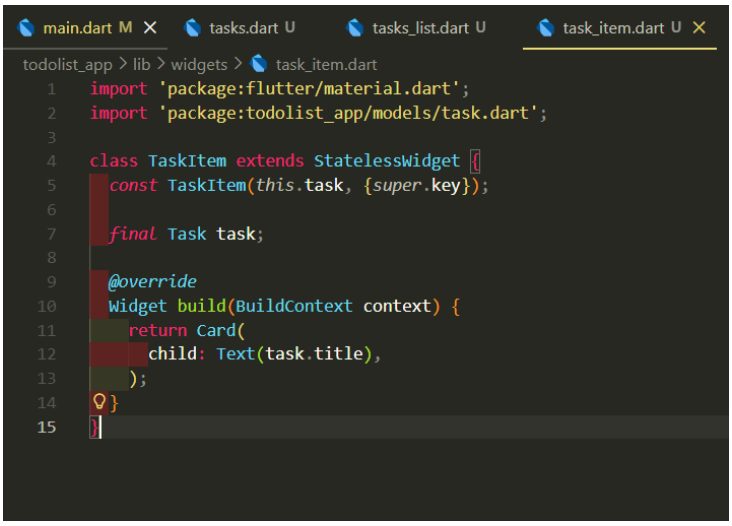

FIGURE 8 – Widget Card pour l'affichage des tâches

9. Étape 8 : Ajout du thème et AppBar

Questions et Réponses

Quel est ThemeData ?
Réponse : Configure le thème global de l'app.
Définit couleurs et styles.
Que fait onPressed() ?
Réponse : Gère l'action du bouton pressé.
Exécute du code au clic.

10. Étape 9 : Widget de saisie NewTask

Questions et Réponses

Pourquoi NewTask est StatefulWidget ?
Réponse : Pour gérer les champs de saisie.
Doit mémoriser le texte entré.

11. Étape 10 : Récupération de la saisie

```
// M thode 1 : Callback
onChanged: _saveTitleInput

// M thode 2 : Controller
controller: _titleController
```

Questions et Réponses

Qu'est-ce qu'un TextEditingController ?
Réponse : Gère programmatiquement les TextField.

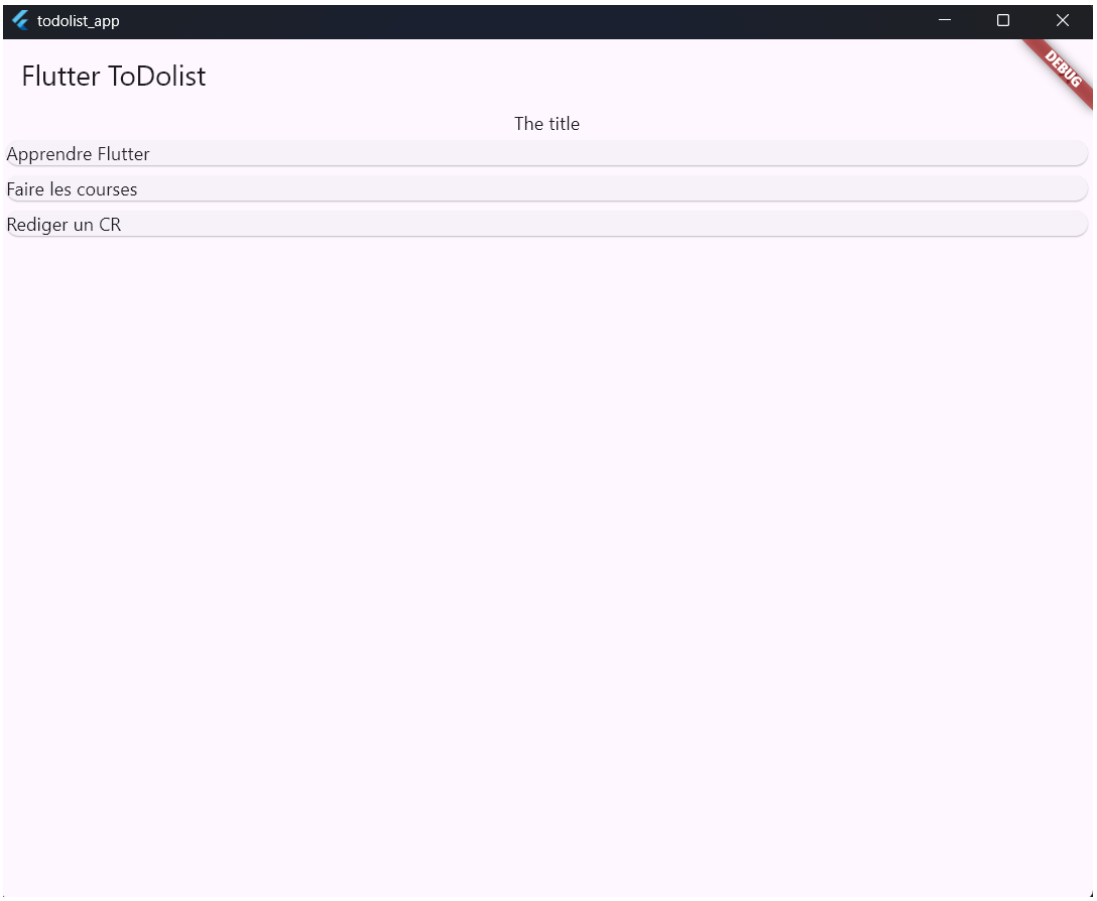

FIGURE 9 – AppBar avec icône d'ajout dans l'application

Permet de lire et écrire le texte.

12. Étape 11 : Contrôle de la saisie

Questions et Réponses

Que fait showDialog() ?
Réponse : Affiche une boîte de dialogue modale.
Pour les messages d'alerte.
Pourquoi trim().isEmpty ?
Réponse : Vérifie si le champ est vide.
Enlève les espaces avant.

13. Étapes 12-14 : Fonctionnalités avancées

Fonctionnalités implémentées

— Ajout de tâches avec tous les champs
— Validation des données saisies
— Affichage des catégories avec couleurs

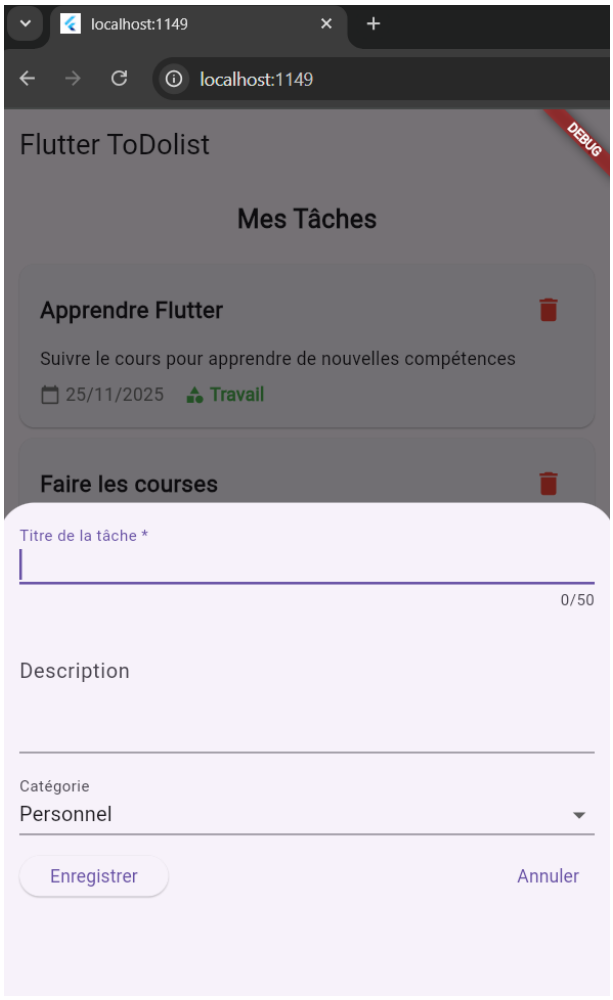

FIGURE 10 – Formulaire de saisie de nouvelle tâche

— Suppression de tâches
— Interface utilisateur complète

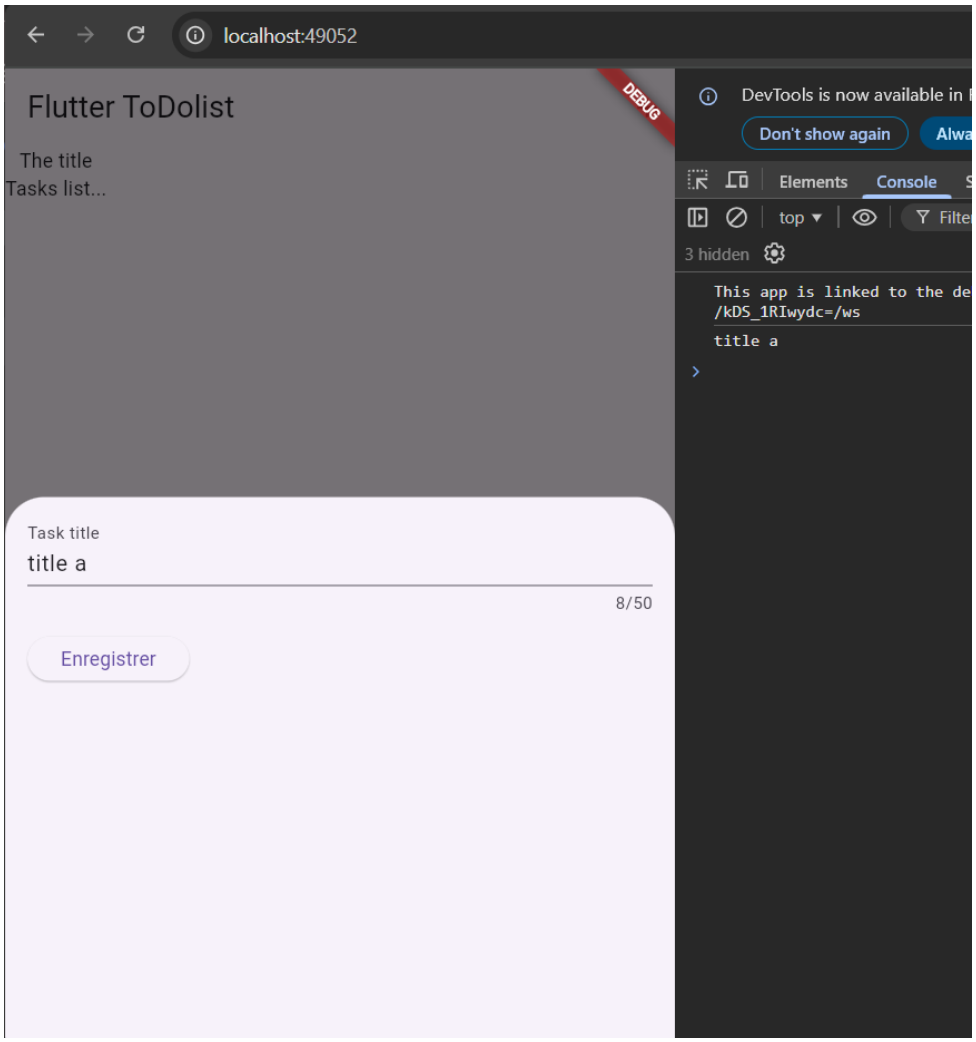

FIGURE 11 – Méthode 1 : Avec une fonction callback (onChanged)

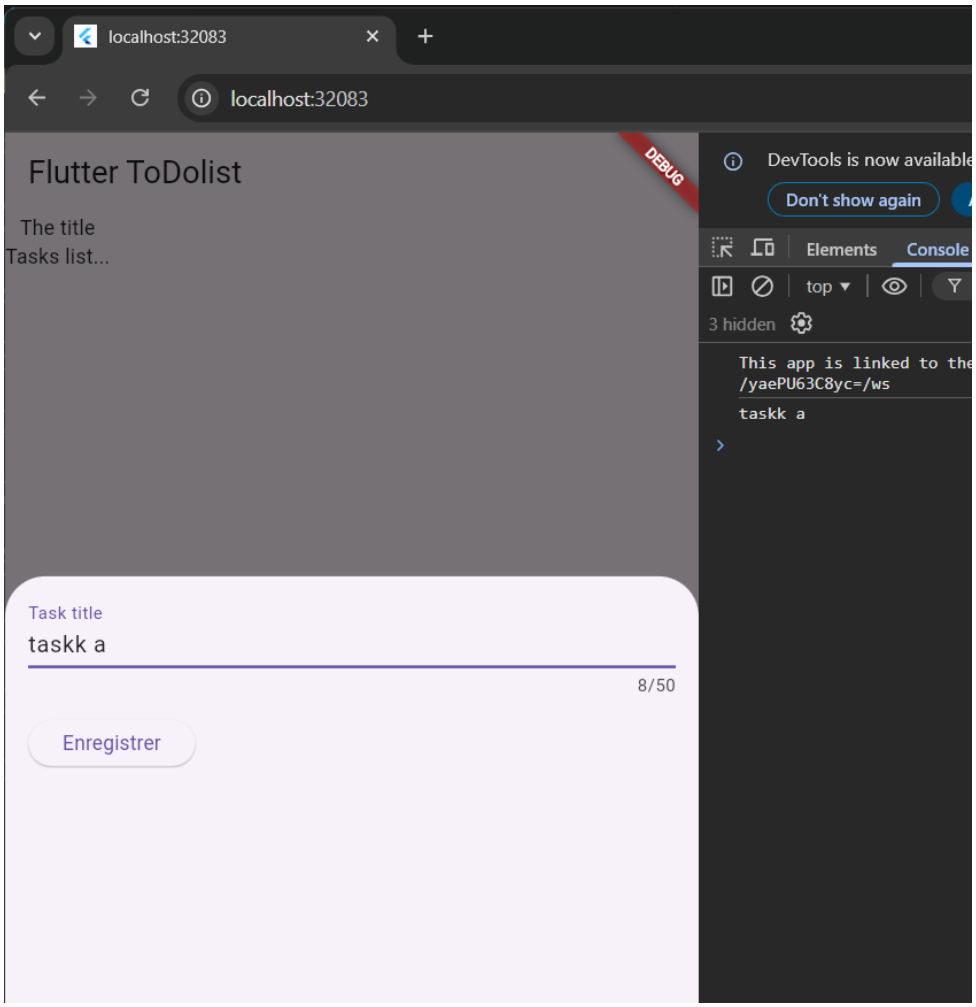

FIGURE 12 – method 2 : $\text{text}_e diting_c ontroller$

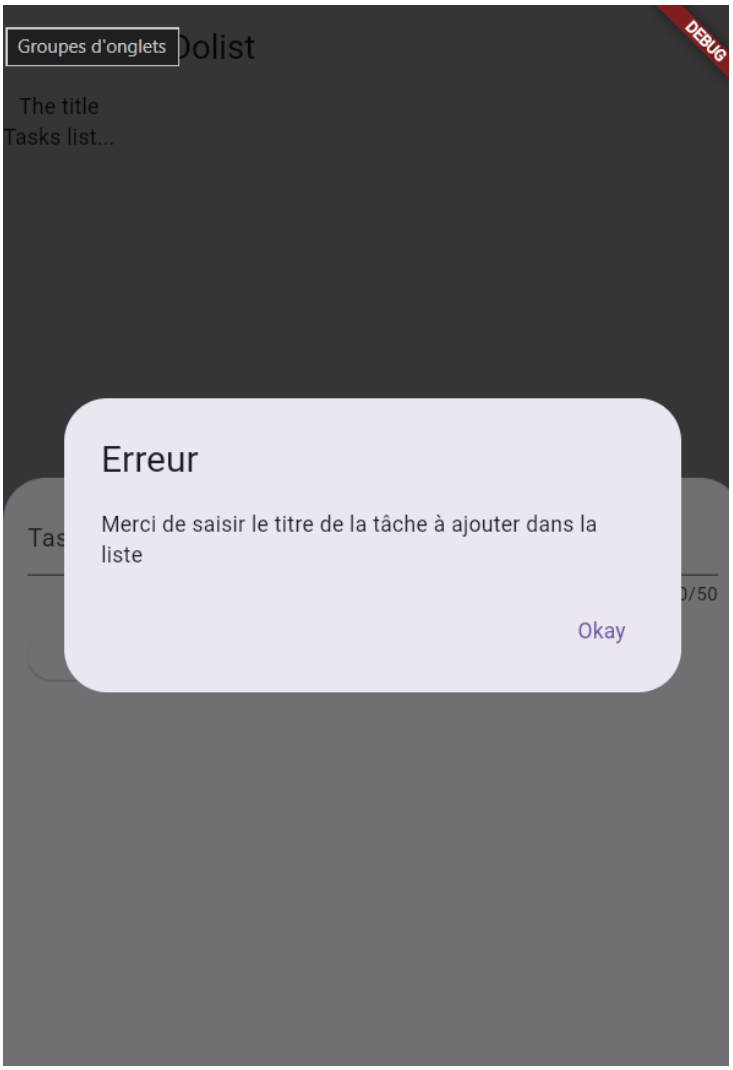

FIGURE 13 – Boîte de dialogue de validation des champs

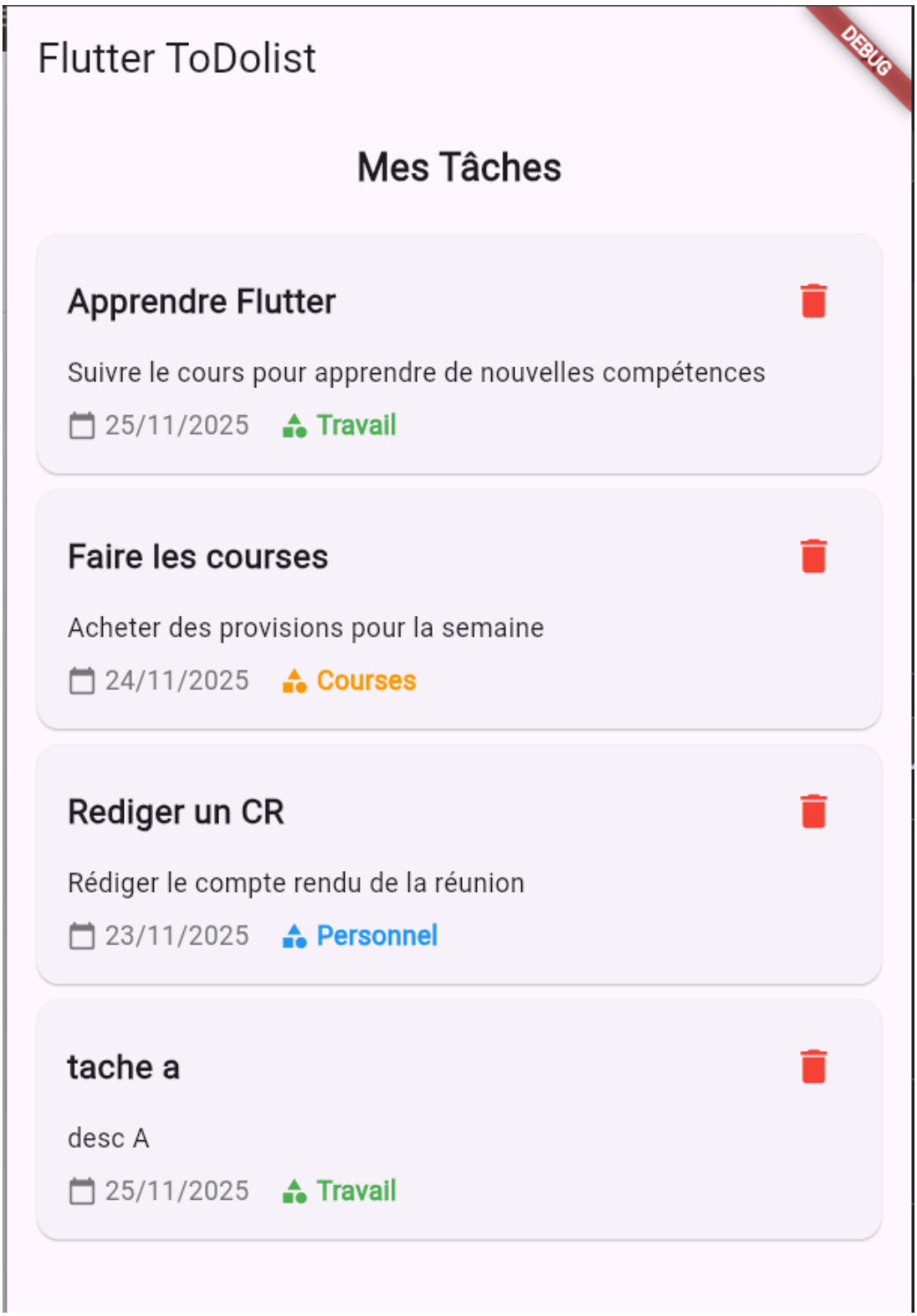

FIGURE 14 – Application TodoList complète et fonctionnelle