Tp9 Création dune application flutter de formulaire dinscription

TP8 : De la maquette Figma à l'application Flutter – Inscription multi-écran et validation de formulaires

1. Objectifs du TP

Ce travail pratique (durée estimée : **6 heures**, dont 1 h 30 de maquettage et 4 h 30 de développement) vous fait parcourir le cycle complet d'un écran d'inscription : conception de l'interface dans Figma, puis implémentation fidèle dans Flutter avec une validation rigoureuse des données.

À la fin du TP, vous serez capable de :

- réaliser **manuellement** des maquettes mobiles dans Figma (frames, grilles, styles, composants, variantes, prototype) ;
- traduire une maquette en widgets Flutter (`Form`, `TextFormField`, `AppBar`, `Card`, `ListTile`, `AlertDialog`) ;
- centraliser la validation dans une classe de validateurs réutilisables (expressions régulières, bornes numériques, champs optionnels, correspondance de mots de passe) ;
- contrôler la saisie avec `keyboardType` et `inputFormatters` ;
- créer un champ de formulaire personnalisé avec `FormField<bool>` (case à cocher obligatoire) ;
- naviguer entre plusieurs écrans en transmettant un objet modèle immuable (`copyWith`) ;
- écrire des tests unitaires pour les validateurs.

2. Prérequis

- Flutter 3.x installé (`flutter doctor` sans erreur bloquante) et un émulateur ou un téléphone.
- Un compte Figma gratuit (navigateur ou application de bureau).
- Avoir lu les cours : *Le widget TextFormField*, *Créer et valider un formulaire*, *Validation des e-mails*.

apcpedagogie.com

3. Règle fondamentale : une maquette = un écran

Le nombre d'écrans de l'application est égal au nombre de maquettes Figma. Chaque *frame* de la page « Maquettes » correspond à exactement un fichier dans `lib/screens/`. Ce TP comporte donc **3 maquettes** et **3 écrans**.

Les états d'un champ (normal, focus, erreur) et la boîte de dialogue de succès ne sont **pas** des écrans : ils sont dessinés comme *variantes de composants* et comme *overlay*, sans frame supplémentaire.

Maquette Figma (frame)	Écran Flutter	Contenu
`01_Etape1_InfosPerso`	`step_one_screen.dart`	Nom, prénom, e-mail, mot de passe, confirmation
`02_Etape2_ProfilPro`	`step_two_screen.dart`	Profession, expérience, TJM, biographie
`03_Recapitulatif`	`summary_screen.dart`	Résumé en cartes, acceptation des conditions, confirmation

4. Partie A – Réalisation des maquettes dans Figma (1 h 30)

Maquettes réalisées à la main, sans IA. Les fonctionnalités de génération de Figma (First Draft, Figma Make, « Make designs »…) et les plugins qui génèrent des écrans ou du code sont **interdits**. Sont autorisés : les outils de dessin, l'Auto Layout, les composants, et un plugin d'icônes (Material Symbols) qui ne fait qu'insérer des icônes existantes. L'historique des versions du fichier peut être consulté lors de l'évaluation.

A.1 Préparer le fichier

1. Créez un nouveau fichier Design nommé `TP8 – Inscription – NomPrenom`.
2. Créez deux pages : **Composants** et **Maquettes**.

A.2 Définir les styles (page Composants)

Créez des *styles locaux* (panneau de droite, rien de sélectionné → *Local styles* → **+**). Ils seront reproduits à l'identique dans le `ThemeData` de Flutter.

Style Figma	Valeur suggérée	Équivalent Flutter
Color / Primary	#00A8CC	`ColorScheme.fromSeed(seedColor: ...)`
Color / Error	#D32F2F	`colorScheme.error`

Style Figma	Valeur suggérée	Équivalent Flutter
Color / Surface	#FFFFFF	colorScheme.surface
Color / Field background	#F2F7F9	InputDecorationTheme.fillColor
Text / Title	Roboto 20, Medium	textTheme.titleLarge
Text / Label	Roboto 14, Regular	labelText du champ
Text / Helper & Error	Roboto 12, Regular	helperText / errorText

A.3 Créer les composants

1. **Champ de saisie**: rectangle 358 × 56, rayon 8, icône à gauche, libellé, texte d'aide en dessous. Mettez l'ensemble en *Auto Layout* (`Maj + A`), puis transformez-le en composant (`Ctrl`/`Cmd + Alt + K`).
2. Ajoutez une propriété de variante `State` avec 4 valeurs : **Default**, **Focus** (bordure 2 px Primary), **Error** (bordure et message en couleur Error), **Filled**. Ces variantes représentent exactement les états que Flutter affiche lors de la validation.
3. **Bouton**: variantes `Type = Primary / Secondary`, hauteur 48, rayon 24, icône optionnelle.
4. **AppBar**: hauteur 56 + barre de progression de 4 px en dessous ; propriété texte pour le titre, propriété booléenne « Bouton retour ».
5. **Carte récapitulative**: titre de section + lignes (icône, libellé, valeur).
6. **Dialogue de succès**: icône de validation, titre, message, bouton « Terminer ».

A.4 Dessiner les 3 maquettes (page Maquettes)

1. Outil Frame (`F`) → préréglage **iPhone 14 (390 × 844)** ou **Android Compact (412 × 917)**. Gardez la même taille pour les 3 frames.
2. Ajoutez une grille de mise en page : 4 colonnes, marge 16, gouttière 16.
3. Nommez les frames exactement : `01_Etape1_InfosPerso`, `02_Etape2_ProfilPro`, `03_Recapitulatif`.
4. Construisez chaque écran **uniquement avec des instances** de vos composants, empilées en Auto Layout vertical (padding 16, espacement 16) :
    - **Maquette 1**: AppBar « Inscription – Étape 1/3 » (progression 1/3), 5 champs (Nom, Prénom, E-mail, Mot de passe avec icône œil, Confirmation), bouton Primary « Suivant ». Placez au moins un champ en variante *Error* pour montrer le message attendu.
    - **Maquette 2**: AppBar « Profil Professionnel – Étape 2/3 » avec bouton retour (progression 2/3), champs Profession, Années d'expérience, TJM (suffixe « € »), Biographie multiligne avec compteur `0/300`, bouton « Suivant ».
    - **Maquette 3**: AppBar « Récapitulatif de votre profil » avec icône de réinitialisation, deux cartes (Identité / Profil professionnel), case « J'accepte les conditions », bouton «

Confirmer et créer le compte ».

A.5 Prototype

1. Onglet *Prototype*: bouton « Suivant » de la maquette 1 → *Navigate to* maquette 2 (animation *Slide in*).
2. Bouton « Suivant » de la maquette 2 → maquette 3 ; bouton retour → *Back*.
3. Bouton « Confirmer » de la maquette 3 → *Open overlay*: dialogue de succès (centré, fond assombri).
4. Testez avec *Present* (`Ctrl`/`Cmd` + `Alt` + `Entrée`).

A.6 Livrables Figma

- Lien de partage du fichier (droit « peut voir »).
- Export PNG @2x des 3 frames + une capture de la page Composants montrant les variantes.

A.7 Correspondance composants Figma → widgets Flutter

Composant Figma	Widget Flutter
Champ de saisie (variantes Default / Focus / Error)	`TextFormField` + `InputDecorationTheme` (`border`, `focusedBorder`, `errorBorder`)
Bouton Primary / Secondary	`FilledButton` / `OutlinedButton`
AppBar + progression	`AppBar` + `bottom: LinearProgressIndicator`
Carte récapitulative	`Card` + `ListTile`
Case à cocher conditions	`FormField<bool>` + `CheckboxListTile`
Overlay de succès	`showDialog` + `AlertDialog`
Auto Layout vertical (padding 16, gap 16)	`ListView(padding: EdgeInsets.all(16))` + `SizedBox(height: 16)`

5. Partie B – Architecture du projet Flutter

Créez le projet (`flutter create tp_inscription`) et organisez le dossier `lib/` ainsi :

```
lib/
│
├── models/
│   └── user_profile.dart        # Modèle immuable + copyWith
│
```

```
├── utils/
│   └── validators.dart          # Toutes les règles de validation
│
├── screens/
│   ├── step_one_screen.dart     # Maquette 01
│   ├── step_two_screen.dart     # Maquette 02
│   └── summary_screen.dart      # Maquette 03
│
└── main.dart                    # Thème issu des styles Figma

test/
└── validators_test.dart         # Tests unitaires des validateurs
```

6. Étape 1 – Modèle de données `UserProfile`

Le modèle est **immuable**: chaque écran ne modifie pas l'objet reçu, il en crée une copie enrichie avec `copyWith`. C'est ce qui permet la « fusion » des données d'un écran à l'autre sans effet de bord.

lib/models/user_profile.dart

Afficher le code

```
class UserProfile {
  final String nom;
  final String prenom;
  final String email;
  final String password;
  final String profession;
  final int experience;
  final double tjm;
  final String biographie;

  const UserProfile({
    this.nom = '',
    this.prenom = '',
    this.email = '',
    this.password = '',
    this.profession = '',
    this.experience = 0,
    this.tjm = 0.0,
    this.biographie = '',
  });
```

apcpedagogie.com

```
  /// Crée une copie en remplaçant uniquement les champs fournis.
  UserProfile copyWith({
    String? nom,
    String? prenom,
    String? email,
    String? password,
    String? profession,
    int? experience,
    double? tjm,
    String? biographie,
  }) {
    return UserProfile(
      nom: nom ?? this.nom,
      prenom: prenom ?? this.prenom,
      email: email ?? this.email,
      password: password ?? this.password,
      profession: profession ?? this.profession,
      experience: experience ?? this.experience,
      tjm: tjm ?? this.tjm,
      biographie: biographie ?? this.biographie,
    );
  }

  String get nomComplet => '$prenom ${nom.toUpperCase()}';

  @override
  String toString() =>
      'UserProfile($nomComplet, $email, $profession, $experience an(s), $tjm €)';
}
```

7. Étape 2 – Centraliser la validation (`Validators`)

Plutôt que d'écrire les règles directement dans chaque `TextFormField`, on les regroupe dans une classe. Un validateur reçoit la valeur saisie et renvoie `null` si elle est valide, ou le **message d'erreur** à afficher sous le champ.

Champ	Règles	Technique
Nom, Prénom	Obligatoire, ≥ 2 caractères, lettres (accents, tiret, apostrophe, espace)	`trim()` + RegExp
E-mail	Obligatoire, format `xxx@domaine.ext`	RegExp

Champ	Règles	Technique
Mot de passe	≥ 8 caractères, 1 majuscule, 1 chiffre	Plusieurs RegExp, message précis par règle
Confirmation	Identique au mot de passe	Comparaison avec un autre contrôleur
Profession	Obligatoire, ≥ 2 caractères	`trim()`
Expérience	Entier entre 0 et 50	`int.tryParse` + bornes
TJM	Nombre entre 50 et 3000 €, virgule acceptée	`double.tryParse` + bornes
Biographie	**Facultative**, mais ≥ 20 caractères si remplie	Validation conditionnelle

lib/utils/validators.dart

Afficher le code

```
/// Validateurs réutilisables : renvoient null si la valeur est valide,
/// sinon le message d'erreur à afficher sous le champ.
class Validators {
  Validators._(); // classe utilitaire, non instanciable

  static final RegExp _nomRegExp = RegExp(r"^[A-Za-zÀ-ÖØ-öø-ÿ' -]+$");
  static final RegExp _emailRegExp =
      RegExp(r'^[\w.+-]+@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*\.[a-zA-Z]{2,}$');
  static final RegExp _majuscule = RegExp(r'[A-Z]');
  static final RegExp _chiffre = RegExp(r'[0-9]');

  static String? nom(String? value, {String champ = 'Ce champ'}) {
    final v = value?.trim() ?? '';
    if (v.isEmpty) return '$champ est obligatoire';
    if (v.length < 2) return '$champ doit contenir au moins 2 caractères';
    if (!_nomRegExp.hasMatch(v)) return '$champ ne doit contenir que des lettres';
    return null;
  }

  static String? email(String? value) {
    final v = value?.trim() ?? '';
    if (v.isEmpty) return "L'e-mail est obligatoire";
    if (!_emailRegExp.hasMatch(v)) return 'Format attendu : nom@domaine.com';
    return null;
  }
```

apcpedagogie.com

```
  static String? motDePasse(String? value) {
    final v = value ?? '';
    if (v.isEmpty) return 'Le mot de passe est obligatoire';
    if (v.length < 8) return 'Au moins 8 caractères';
    if (!_majuscule.hasMatch(v)) return 'Au moins une lettre majuscule';
    if (!_chiffre.hasMatch(v)) return 'Au moins un chiffre';
    return null;
  }

  static String? confirmation(String? value, String original) {
    if (value == null || value.isEmpty) return 'Veuillez confirmer le mot de passe';
    if (value != original) return 'Les mots de passe ne correspondent pas';
    return null;
  }

  static String? profession(String? value) {
    final v = value?.trim() ?? '';
    if (v.isEmpty) return 'La profession est obligatoire';
    if (v.length < 2) return 'Profession trop courte';
    return null;
  }

  static String? experience(String? value) {
    final v = value?.trim() ?? '';
    if (v.isEmpty) return "Indiquez vos années d'expérience";
    final n = int.tryParse(v);
    if (n == null) return 'Nombre entier attendu';
    if (n < 0 || n > 50) return 'Valeur comprise entre 0 et 50';
    return null;
  }

  static String? tjm(String? value) {
    final v = (value ?? '').trim().replaceAll(',', '.');
    if (v.isEmpty) return 'Le TJM est obligatoire';
    final n = double.tryParse(v);
    if (n == null) return 'Montant invalide';
    if (n < 50 || n > 3000) return 'Le TJM doit être compris entre 50 € et 3000 €';
    return null;
  }

  /// Champ facultatif : vide = valide, sinon au moins 20 caractères.
  static String? biographie(String? value) {
    final v = value?.trim() ?? '';
```

apcpedagogie.com

```
    if (v.isEmpty) return null;
    if (v.length < 20) {
      return 'Au moins 20 caractères (${v.length}/20) ou laissez vide';
    }
    return null;
  }
}
```

8. Étape 3 – Écran 1 : Informations personnelles (maquette 01)

- **AppBar :** titre « Inscription – Étape 1/3 » et barre de progression (1/3), conformément à la maquette.
- **Validation en direct :** `autovalidateMode: AutovalidateMode.onUserInteraction` affiche l'erreur dès que l'utilisateur a touché un champ.
- **Revalidation croisée :** si le mot de passe change après la confirmation, on revalide la confirmation grâce à une clé `GlobalKey<FormFieldState<String>>`.
- **Ergonomie :** `textInputAction: TextInputAction.next` pour passer au champ suivant, icône œil pour afficher/masquer le mot de passe, `dispose()` des contrôleurs.
- **Navigation :** le bouton « Suivant » n'avance que si `_formKey.currentState!.validate()` renvoie `true`.

lib/screens/step_one_screen.dart

Afficher le code

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

import '../models/user_profile.dart';
import '../utils/validators.dart';
import 'step_two_screen.dart';

class StepOneScreen extends StatefulWidget {
  const StepOneScreen({super.key});

  @override
  State<StepOneScreen> createState() => _StepOneScreenState();
}

class _StepOneScreenState extends State<StepOneScreen> {
  final _formKey = GlobalKey<FormState>();
```

apcpedagogie.com

```
final _confirmKey = GlobalKey<FormFieldState<String>>();

final _nomCtrl = TextEditingController();
final _prenomCtrl = TextEditingController();
final _emailCtrl = TextEditingController();
final _passwordCtrl = TextEditingController();
final _confirmCtrl = TextEditingController();

bool _masquerMdp = true;
bool _masquerConfirm = true;

@override
void dispose() {
  _nomCtrl.dispose();
  _prenomCtrl.dispose();
  _emailCtrl.dispose();
  _passwordCtrl.dispose();
  _confirmCtrl.dispose();
  super.dispose();
}

void _suivant() {
  FocusScope.of(context).unfocus();
  if (!_formKey.currentState!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Veuillez corriger les champs en rouge')),
    );
    return;
  }
  final profil = UserProfile(
    nom: _nomCtrl.text.trim(),
    prenom: _prenomCtrl.text.trim(),
    email: _emailCtrl.text.trim(),
    password: _passwordCtrl.text,
  );
  Navigator.push(
    context,
    MaterialPageRoute(builder: (_) => StepTwoScreen(profil: profil)),
  );
}

@override
Widget build(BuildContext context) {
```

apcpedagogie.com

```
    return Scaffold(
      appBar: AppBar(
        title: const Text('Inscription - Étape 1/3'),
        bottom: const PreferredSize(
          preferredSize: Size.fromHeight(4),
          child: LinearProgressIndicator(value: 1 / 3),
        ),
      ),
      body: SafeArea(
        child: Form(
          key: _formKey,
          autovalidateMode: AutovalidateMode.onUserInteraction,
          child: ListView(
            padding: const EdgeInsets.all(16),
            children: [
              TextFormField(
                controller: _nomCtrl,
                decoration: const InputDecoration(
                  labelText: 'Nom',
                  prefixIcon: Icon(Icons.person_outline),
                ),
                textCapitalization: TextCapitalization.characters,
                textInputAction: TextInputAction.next,
                validator: (v) => Validators.nom(v, champ: 'Le nom'),
              ),
              const SizedBox(height: 16),
              TextFormField(
                controller: _prenomCtrl,
                decoration: const InputDecoration(
                  labelText: 'Prénom',
                  prefixIcon: Icon(Icons.badge_outlined),
                ),
                textCapitalization: TextCapitalization.words,
                textInputAction: TextInputAction.next,
                validator: (v) => Validators.nom(v, champ: 'Le prénom'),
              ),
              const SizedBox(height: 16),
              TextFormField(
                controller: _emailCtrl,
                decoration: const InputDecoration(
                  labelText: 'E-mail',
                  hintText: 'nom@domaine.com',
                  prefixIcon: Icon(Icons.email_outlined),
```

apcpedagogie.com

```
    ),
    keyboardType: TextInputType.emailAddress,
    autocorrect: false,
    textInputAction: TextInputAction.next,
    validator: Validators.email,
  ),
  const SizedBox(height: 16),
  TextFormField(
    controller: _passwordCtrl,
    obscureText: _masquerMdp,
    decoration: InputDecoration(
      labelText: 'Mot de passe',
      helperText: '8 caractères min., 1 majuscule, 1 chiffre',
      prefixIcon: const Icon(Icons.lock_outline),
      suffixIcon: IconButton(
        tooltip: _masquerMdp ? 'Afficher' : 'Masquer',
        icon: Icon(_masquerMdp ? Icons.visibility : Icons.visibility_off),
        onPressed: () => setState(() => _masquerMdp = !_masquerMdp),
      ),
    ),
    textInputAction: TextInputAction.next,
    validator: Validators.motDePasse,
    onChanged: (_) {
      // Revalide la confirmation si elle est déjà remplie
      if (_confirmCtrl.text.isNotEmpty) {
        _confirmKey.currentState?.validate();
      }
    },
  ),
  const SizedBox(height: 16),
  TextFormField(
    key: _confirmKey,
    controller: _confirmCtrl,
    obscureText: _masquerConfirm,
    decoration: InputDecoration(
      labelText: 'Confirmer le mot de passe',
      prefixIcon: const Icon(Icons.lock_reset),
      suffixIcon: IconButton(
        tooltip: _masquerConfirm ? 'Afficher' : 'Masquer',
        icon: Icon(_masquerConfirm ? Icons.visibility : Icons.visibility_off),
        onPressed: () => setState(() => _masquerConfirm = !_masquerConfirm),
      ),
    ),
```

apcpedagogie.com

```
                textInputAction: TextInputAction.done,
                onFieldSubmitted: (_) => _suivant(),
                validator: (v) => Validators.confirmation(v, _passwordCtrl.text),
              ),
              const SizedBox(height: 24),
              FilledButton.icon(
                onPressed: _suivant,
                icon: const Icon(Icons.arrow_forward),
                label: const Text('Suivant'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

9. Étape 4 – Écran 2 : Informations professionnelles (maquette 02)

- **AppBar :** titre « Profil Professionnel – Étape 2/3 » avec bouton de retour explicite (`leading`) et progression 2/3.
- **Contrôle de la saisie :** clavier numérique et `inputFormatters` qui empêchent de taper des caractères interdits. Le validateur reste nécessaire (bornes, champ vide).
- **Biographie :** multiligne, compteur de 300 caractères, facultative mais ≥ 20 caractères si remplie.
- **Fusion des données :** `widget.profil.copyWith(...)` puis passage à l'écran 3.

lib/screens/step_two_screen.dart

Afficher le code

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

import '../models/user_profile.dart';
import '../utils/validators.dart';
import 'summary_screen.dart';

class StepTwoScreen extends StatefulWidget {
```

apcpedagogie.com

```
  final UserProfile profil;

  const StepTwoScreen({super.key, required this.profil});

  @override
  State<StepTwoScreen> createState() => _StepTwoScreenState();
}

class _StepTwoScreenState extends State<StepTwoScreen> {
  final _formKey = GlobalKey<FormState>();
  final _professionCtrl = TextEditingController();
  final _experienceCtrl = TextEditingController();
  final _tjmCtrl = TextEditingController();
  final _bioCtrl = TextEditingController();

  @override
  void dispose() {
    _professionCtrl.dispose();
    _experienceCtrl.dispose();
    _tjmCtrl.dispose();
    _bioCtrl.dispose();
    super.dispose();
  }

  void _suivant() {
    FocusScope.of(context).unfocus();
    if (!_formKey.currentState!.validate()) return;

    final profilComplet = widget.profil.copyWith(
      profession: _professionCtrl.text.trim(),
      experience: int.parse(_experienceCtrl.text.trim()),
      tjm: double.parse(_tjmCtrl.text.trim().replaceAll(',', '.')),
      biographie: _bioCtrl.text.trim(),
    );

    Navigator.push(
      context,
      MaterialPageRoute(builder: (_) => SummaryScreen(profil: profilComplet)),
    );
  }

  @override
  Widget build(BuildContext context) {
```

apcpedagogie.com

```
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          tooltip: 'Retour à l'étape 1',
          onPressed: () => Navigator.pop(context),
        ),
        title: const Text('Profil Professionnel - Étape 2/3'),
        bottom: const PreferredSize(
          preferredSize: Size.fromHeight(4),
          child: LinearProgressIndicator(value: 2 / 3),
        ),
      ),
      body: SafeArea(
        child: Form(
          key: _formKey,
          autovalidateMode: AutovalidateMode.onUserInteraction,
          child: ListView(
            padding: const EdgeInsets.all(16),
            children: [
              Text(
                'Bonjour ${widget.profil.prenom}, parlez-nous de votre activité.',
                style: Theme.of(context).textTheme.titleMedium,
              ),
              const SizedBox(height: 16),
              TextFormField(
                controller: _professionCtrl,
                decoration: const InputDecoration(
                  labelText: 'Profession',
                  hintText: 'Ex. : Développeur mobile',
                  prefixIcon: Icon(Icons.work_outline),
                ),
                textCapitalization: TextCapitalization.sentences,
                textInputAction: TextInputAction.next,
                validator: Validators.profession,
              ),
              const SizedBox(height: 16),
              TextFormField(
                controller: _experienceCtrl,
                decoration: const InputDecoration(
                  labelText: "Années d'expérience",
                  prefixIcon: Icon(Icons.timeline),
                ),
```

apcpedagogie.com

```
    keyboardType: TextInputType.number,
    inputFormatters: [
      FilteringTextInputFormatter.digitsOnly,
      LengthLimitingTextInputFormatter(2),
    ],
    textInputAction: TextInputAction.next,
    validator: Validators.experience,
  ),
  const SizedBox(height: 16),
  TextFormField(
    controller: _tjmCtrl,
    decoration: const InputDecoration(
      labelText: 'Tarif Journalier Moyen (TJM)',
      prefixIcon: Icon(Icons.euro),
      suffixText: '€',
    ),
    keyboardType: const TextInputType.numberWithOptions(decimal: true),
    inputFormatters: [
      // chiffres, un seul séparateur (. ou ,), 2 décimales max
      FilteringTextInputFormatter.allow(RegExp(r'^\d*[.,]?\d{0,2}')),
    ],
    textInputAction: TextInputAction.next,
    validator: Validators.tjm,
  ),
  const SizedBox(height: 16),
  TextFormField(
    controller: _bioCtrl,
    decoration: const InputDecoration(
      labelText: 'Biographie (facultative)',
      hintText: 'Présentez votre parcours en quelques lignes',
      alignLabelWithHint: true,
    ),
    minLines: 3,
    maxLines: 5,
    maxLength: 300,
    keyboardType: TextInputType.multiline,
    textCapitalization: TextCapitalization.sentences,
    validator: Validators.biographie,
  ),
  const SizedBox(height: 24),
  FilledButton.icon(
    onPressed: _suivant,
    icon: const Icon(Icons.arrow_forward),
```

apcpedagogie.com

```
                label: const Text('Suivant'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

10. Étape 5 – Écran 3 : Récapitulatif et validation finale (maquette 03)

- **AppBar :** titre « Récapitulatif de votre profil » et action de réinitialisation qui demande une confirmation avant de tout effacer.
- **Affichage :** deux `Card` contenant des `ListTile`. Le mot de passe n'est jamais affiché en clair.
- **Validation d'un champ non textuel :** la case « J'accepte les conditions » est un `FormField<bool>` avec son propre validateur.
- **Confirmation :** boîte de dialogue de succès (non fermable en touchant à côté), puis retour à l'écran 1 avec vidage de la pile de navigation.

lib/screens/summary_screen.dart

Afficher le code

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

import '../models/user_profile.dart';
import 'step_one_screen.dart';

class SummaryScreen extends StatefulWidget {
  final UserProfile profil;

  const SummaryScreen({super.key, required this.profil});

  @override
  State<SummaryScreen> createState() => _SummaryScreenState();
}

class _SummaryScreenState extends State<SummaryScreen> {
  final _formKey = GlobalKey<FormState>();
```

apcpedagogie.com

```
void _retourDebut() {
  Navigator.of(context).pushAndRemoveUntil(
    MaterialPageRoute(builder: (_) => const StepOneScreen()),
    (route) => false,
  );
}

Future<void> _reinitialiser() async {
  final confirme = await showDialog<bool>(
    context: context,
    builder: (ctx) => AlertDialog(
      title: const Text('Réinitialiser ?'),
      content: const Text('Toutes les données saisies seront perdues.'),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(ctx, false),
          child: const Text('Annuler'),
        ),
        FilledButton(
          onPressed: () => Navigator.pop(ctx, true),
          child: const Text('Réinitialiser'),
        ),
      ],
    ),
  );
  if (confirme == true && mounted) _retourDebut();
}

Future<void> _confirmer() async {
  if (!_formKey.currentState!.validate()) return;

  await showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (ctx) => AlertDialog(
      icon: const Icon(Icons.check_circle, color: Colors.green, size: 48),
      title: const Text('Compte créé !'),
      content: Text(
        'Bienvenue ${widget.profil.prenom}.\n'
        'Un e-mail de confirmation a été envoyé à ${widget.profil.email}.',
        textAlign: TextAlign.center,
      ),
```

apcpedagogie.com

```
        actions: [
          FilledButton(
            onPressed: () => Navigator.pop(ctx),
            child: const Text('Terminer'),
          ),
        ],
      ),
    );
    if (mounted) _retourDebut();
  }

  Widget _section(String titre, List<Widget> lignes) {
    return Card(
      margin: const EdgeInsets.only(bottom: 16),
      child: Padding(
        padding: const EdgeInsets.symmetric(vertical: 8),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Padding(
              padding: const EdgeInsets.fromLTRB(16, 8, 16, 0),
              child: Text(titre, style: Theme.of(context).textTheme.titleMedium),
            ),
            ...lignes,
          ],
        ),
      ),
    );
  }

  Widget _ligne(IconData icone, String libelle, String valeur) {
    return ListTile(
      leading: Icon(icone),
      title: Text(libelle),
      subtitle: Text(valeur.isEmpty ? 'Non renseigné' : valeur),
    );
  }

  @override
  Widget build(BuildContext context) {
    final p = widget.profil;

    return Scaffold(
```

apcpedagogie.com

```
      appBar: AppBar(
        title: const Text('Récapitulatif de votre profil'),
        actions: [
          IconButton(
            tooltip: 'Réinitialiser',
            icon: const Icon(Icons.restart_alt),
            onPressed: _reinitialiser,
          ),
        ],
        bottom: const PreferredSize(
          preferredSize: Size.fromHeight(4),
          child: LinearProgressIndicator(value: 1),
        ),
      ),
      body: SafeArea(
        child: Form(
          key: _formKey,
          child: ListView(
            padding: const EdgeInsets.all(16),
            children: [
              _section('Identité', [
                _ligne(Icons.person_outline, 'Nom complet', p.nomComplet),
                _ligne(Icons.email_outlined, 'E-mail', p.email),
                _ligne(Icons.lock_outline, 'Mot de passe', '•' * p.password.length),
              ]),
              _section('Profil professionnel', [
                _ligne(Icons.work_outline, 'Profession', p.profession),
                _ligne(Icons.timeline, 'Expérience', '${p.experience} an(s)'),
                _ligne(Icons.euro, 'TJM', '${p.tjm.toStringAsFixed(2)} €'),
                _ligne(Icons.notes, 'Biographie', p.biographie),
              ]),
              FormField<bool>(
                initialValue: false,
                validator: (v) =>
                    v == true ? null : "Vous devez accepter les conditions d'utilisation",
                builder: (field) => Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    CheckboxListTile(
                      value: field.value,
                      onChanged: field.didChange,
                      title: const Text("J'accepte les conditions d'utilisation"),
                      controlAffinity: ListTileControlAffinity.leading,
```

apcpedagogie.com

```
                    contentPadding: EdgeInsets.zero,
                  ),
                  if (field.hasError)
                    Padding(
                      padding: const EdgeInsets.only(left: 12),
                      child: Text(
                        field.errorText!,
                        style: TextStyle(
                          color: Theme.of(context).colorScheme.error,
                          fontSize: 12,
                        ),
                      ),
                    ),
                ],
              ),
            ),
            const SizedBox(height: 24),
            FilledButton.icon(
              onPressed: _confirmer,
              icon: const Icon(Icons.check),
              label: const Text('Confirmer et créer le compte'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

11. Étape 6 – Le thème : reproduire les styles Figma

Les couleurs, rayons et bordures définis dans Figma (A.2 et A.3) sont regroupés dans un seul `ThemeData`. Ainsi tous les champs prennent automatiquement l'apparence des variantes *Default*, *Focus* et *Error* de votre composant.

lib/main.dart

Afficher le code

apcpedagogie.com

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

import 'screens/step_one_screen.dart';

void main() => runApp(const InscriptionApp());

class InscriptionApp extends StatelessWidget {
  const InscriptionApp({super.key});

  // Valeurs reprises des styles Figma
  static const Color primary = Color(0xFF00A8CC);
  static const Color error = Color(0xFFD32F2F);
  static const Color fieldBackground = Color(0xFFF2F7F9);

  @override
  Widget build(BuildContext context) {
    final scheme = ColorScheme.fromSeed(seedColor: primary, error: error);
    final rayon = BorderRadius.circular(8);

    return MaterialApp(
      title: 'TP8 Inscription',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        useMaterial3: true,
        colorScheme: scheme,
        appBarTheme: AppBarTheme(
          backgroundColor: primary,
          foregroundColor: Colors.white,
          centerTitle: true,
        ),
        inputDecorationTheme: InputDecorationTheme(
          filled: true,
          fillColor: fieldBackground,
          border: OutlineInputBorder(borderRadius: rayon),
          enabledBorder: OutlineInputBorder(
            borderRadius: rayon,
            borderSide: BorderSide(color: Colors.grey.shade400),
          ),
          focusedBorder: OutlineInputBorder(
            borderRadius: rayon,
            borderSide: const BorderSide(color: primary, width: 2),
          ),
          errorBorder: OutlineInputBorder(
```

apcpedagogie.com

```
            borderRadius: rayon,
            borderSide: const BorderSide(color: error),
          ),
          focusedErrorBorder: OutlineInputBorder(
            borderRadius: rayon,
            borderSide: const BorderSide(color: error, width: 2),
          ),
        ),
        filledButtonTheme: FilledButtonThemeData(
          style: FilledButton.styleFrom(
            minimumSize: const Size.fromHeight(48),
            shape: const StadiumBorder(),
          ),
        ),
      ),
      home: const StepOneScreen(),
    );
  }
}
```

12. Étape 7 – Tester la validation

12.1 Jeu de tests manuels

Saisissez les valeurs suivantes et vérifiez que le message obtenu correspond à celui attendu (et à la variante *Error* de votre maquette).

Champ	Saisie	Résultat attendu
Nom	`A`	Le nom doit contenir au moins 2 caractères
Nom	`Ben 3li`	Le nom ne doit contenir que des lettres
Prénom	(espaces)	Le prénom est obligatoire
E-mail	`ali@gmail`	Format attendu : nom@domaine.com
Mot de passe	`bonjour12`	Au moins une lettre majuscule
Confirmation	différente du mot de passe	Les mots de passe ne correspondent pas
Expérience	75	Valeur comprise entre 0 et 50
TJM	`abc`	Saisie impossible (bloquée par le formatter)
TJM	25	Le TJM doit être compris entre 50 € et 3000 €

apcpedagogie.com

Champ	Saisie	Résultat attendu
TJM	`350,50`	Valide (la virgule est acceptée)
Biographie	(vide)	Valide
Biographie	`Je code.`	Au moins 20 caractères (8/20) ou laissez vide
Conditions	non cochée	Vous devez accepter les conditions d’utilisation

12.2 Tests unitaires

Les validateurs étant de simples fonctions, ils se testent sans interface. Remplacez `tp_inscription` par le nom de votre projet (champ `name` du `pubspec.yaml`), puis lancez `flutter test`.

test/validators_test.dart

Afficher le code

```
import 'package:flutter_test/flutter_test.dart';
import 'package:tp_inscription/utils/validators.dart';

void main() {
  group('Validators.email', () {
    test('refuse une valeur vide', () {
      expect(Validators.email(''), isNotNull);
    });
    test('refuse une adresse sans extension', () {
      expect(Validators.email('ali@gmail'), isNotNull);
    });
    test('accepte une adresse valide', () {
      expect(Validators.email('ali.ben@gmail.com'), isNull);
    });
  });

  group('Validators.motDePasse', () {
    test('trop court', () {
      expect(Validators.motDePasse('Ab1'), 'Au moins 8 caractères');
    });
    test('sans majuscule', () {
      expect(Validators.motDePasse('bonjour12'), 'Au moins une lettre majuscule');
    });
    test('sans chiffre', () {
```

apcpedagogie.com

```
      expect(Validators.motDePasse('Bonjourrr'), 'Au moins un chiffre');
    });
    test('valide', () {
      expect(Validators.motDePasse('Bonjour12'), isNull);
    });
  });

  group('Validators.confirmation', () {
    test('différente', () {
      expect(Validators.confirmation('Abc12345', 'Abc12346'), isNotNull);
    });
    test('identique', () {
      expect(Validators.confirmation('Abc12345', 'Abc12345'), isNull);
    });
  });

  group('Validators numériques', () {
    test('expérience hors bornes', () {
      expect(Validators.experience('51'), isNotNull);
    });
    test('TJM avec virgule', () {
      expect(Validators.tjm('350,50'), isNull);
    });
    test('TJM trop bas', () {
      expect(Validators.tjm('20'), isNotNull);
    });
  });

  group('Validators.biographie (facultative)', () {
    test('vide = valide', () {
      expect(Validators.biographie(''), isNull);
    });
    test('trop courte', () {
      expect(Validators.biographie('Je code.'), isNotNull);
    });
  });
}
```

13. Exercices complémentaires (bonus)

1. **Indicateur de force**: sous le mot de passe, ajoutez une `LinearProgressIndicator` colorée (faible / moyen / fort) calculée dans `onChanged`. Ajoutez d’abord la variante correspondante dans Figma.

apcpedagogie.com

2. **Caractère spécial**: ajoutez une règle « au moins un caractère spécial » et le test unitaire associé.
3. **Profession par liste**: remplacez le champ Profession par un `DropdownButtonFormField<String>` avec validateur.
4. **Modifier depuis le récapitulatif**: ajoutez un bouton « Modifier » sur chaque carte qui ramène à l'étape concernée (`Navigator.popUntil`).
5. **Quitter sans perdre les données**: sur l'écran 2, utilisez `PopScope` pour demander confirmation si des champs sont déjà remplis.

14. Livrables et barème (/20)

Critère	Points
Figma – 3 frames nommées, même taille, grille appliquée	1,5
Figma – styles de couleurs et de texte utilisés partout	1
Figma – composants avec variantes (champ : 4 états, bouton : 2 types)	2
Figma – prototype navigable + overlay de succès, réalisé sans IA	1,5
Flutter – conformité visuelle aux maquettes (3 écrans = 3 maquettes)	3
Flutter – validateurs corrects et centralisés	4
Flutter – contrôle de saisie (claviers, formatters) et `FormField<bool>`	2
Flutter – navigation, `copyWith`, réinitialisation, dialogue	3
Tests – `flutter test` passe sans erreur	2
Total	**20**

À rendre : le lien Figma, les exports PNG, l'archive du dossier `lib/` et `test/`, et trois captures d'écran de l'application placées à côté des maquettes correspondantes.

Cours Flutter

Source : https://apcpedagogie.com/tp9-creation-dune-application-flutter-de-formulaire-dinscription · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com