Flutter Forms and Lists

1. El primer requisito era crear una pantalla de inicio de sesión en la que los usuarios introdujeran un nombre de usuario y una contraseña mediante un formulario. En nuestra implementación, usamos el **widget Form** de Flutter para manejar la entrada del usuario.

```
// Login Card
Card(
  shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
  elevation: 5,
  color: Colors.white,
  child: Padding(
    padding: EdgeInsets.all(20),
    child: Form(
      key: _formKey,
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(
            "Welcome to Flutter Cities App",
            style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold, color: Colors.blue),
          ), // Text
          SizedBox(height: 20),
```

Creamos dos widgets **“TextFormField”** para el nombre de usuario y la contraseña, envueltos dentro de un widget Form con un **“GlobalKey<FormState>()”** para permitir la validación.

```
    ), // InputDecoration
    validator: (value) {
      if (value == null || value.isEmpty) return "Enter a username";
      return null;
    },
  ), // TextFormField
  SizedBox(height: 15),
```

```
    ), // InputDecoration
    validator: (value) {
      if (value == null || value.length < 7 || !RegExp(r'[0-9]').hasMatch(value)) {
        return "Password must be at least 7 characters & contain numbers";
      }
      return null;
```

Para validar la contraseña, nos aseguramos de que contiene al menos 7 caracteres. La lógica de validación está dentro del parámetro validator del **“TextFormField”**. Si la contraseña no cumple los criterios, se muestra un mensaje de error.

El formulario de acceso también incluye un botón de validación. Si la validación tiene éxito, la aplicación navega a la segunda pantalla **(CitiesScreen)** mientras pasa el nombre de usuario como parámetro.

```
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue,
          ),
        ), // ElevatedButtonThemeData
        inputDecorationTheme: InputDecorationTheme(
          filled: true,
          fillColor: Colors.white,
          focusedBorder: OutlineInputBorder(
            borderSide: BorderSide(color: Colors.blue, width: 2),
            borderRadius: BorderRadius.circular(12),
          ), // OutlineInputBorder
        ), // InputDecorationTheme
      ), // ThemeData
      home: LoginScreen(),
    ); // MaterialApp
```

2. El segundo requisito era crear una lista de ciudades que se mostraran en la segunda pantalla. Para ello utilizamos **ListView.builder**, que genera dinámicamente una lista de ciudades.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text("¡Hola $username!"), backgroundColor: Colors.blue),
    body: ListView.builder(
      itemCount: cities.length,
      itemBuilder: (context, index) {
        return Card(
          margin: EdgeInsets.all(10),
          child: ListTile(
            leading: Hero(
              tag: cities[index]['name']!,
              child: ClipRRect(
                borderRadius: BorderRadius.circular(8),
```

Cada ciudad se representa con una imagen, un nombre, un país y una población. Las imágenes se almacenan en la carpeta **assets/** y se cargan utilizando **Image.asset()**.

```
        child: Image.asset(
          cities[index]['image']!,
          width: 60,
          height: 60,
          fit: BoxFit.cover,
          errorBuilder: (context, error, stackTrace) {
            return Icon(Icons.image_not_supported, size: 60, color: Colors.grey);
          },
        ), // Image.asset
      ), // ClipRRect
    ), // Hero
```

3. El último requisito era navegar a una tercera pantalla al pulsar sobre una ciudad, utilizando el efecto de **animación Hero** y pasando los datos de la ciudad. Implementamos la animación Hero asegurándonos de que la imagen de la ciudad se envolviera con un **widget Hero**, tanto en la pantalla de lista como en la pantalla detallada.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text(cityName), backgroundColor: Colors.blue),
    body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Hero(
            tag: cityName,
            child: ClipRRect(
              borderRadius: BorderRadius.circular(10),
              child: Image.asset(
                imagePath,
                width: double.infinity,
                height: 250,
                fit: BoxFit.cover,
                errorBuilder: (context, error, stackTrace) {
                  return Icon(Icons.image_not_supported, size: 250, color:
                },
              ), // Image.asset
            ), // ClipRRect
          ), // Hero
```

El **widget Hero** crea un suave efecto de animación al pasar de una pantalla a otra, lo que garantiza una mejor experiencia de usuario.