Instituto Tecnológico y de Estudios Superiores de Occidente.

ITESO

Universidad Jesuita de Guadalajara

Tarea 2.p | Code Labs MDC 101-104

Materia: PROGRAMACION DE DISPOSITIVOS MOVILES (O2020_ESI3185B)

Profesor: Camacho Gil, Francisco Javier

Fecha: 30/08/2020

Autor(es): Barragán Alvarez, Daniel

Introducción

Reporte utilizando plantilla de tareas con el desarrollo de los Code Labs MDC101-104.

Desarrollo

El procedimiento para realizar estas guías fue seguir cada sección en los Code Labs. Las primeras secciones suelen ser de configuraciones y descripción de lo que se va a realizar en dicho Code Lab, en la última sección se incluye un resumen de lo que se vio y los siguientes pasos.

MDC-101 Flutter [1]

En este Code Lab se agrega crea una aplicación llamada Shrine, con su página de inicio de sesión. Dentro de esta vista se agrega una imagen del logo, el nombre de la app, dos TextField para usuario y contraseña, además de un botón para cancelar y otro para acceder. El primer paso fue descargar el código de Git indicado en la página y configurarlo para poder correrlo en el emulador.

Primero se crearon los TextField, y dentro de este se agrego un campo de decoration, que tiene widget de InputDecoration. Dentro de este widget, se añade el campo de filled como true para cambiar el color de fondo del TextField para que el usuario presione este.

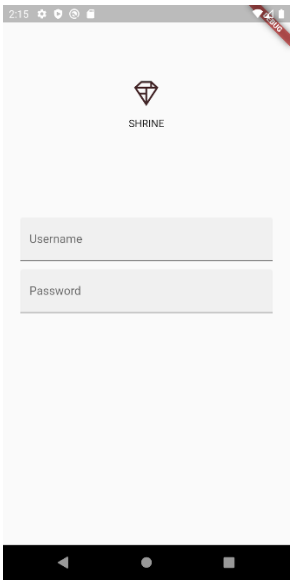


Ilustración 1 Página de inicio con los TextField para usuario y contraseña

Seguido de esto se añade un ButtonBar para contener los botones de cancelar y de ingresar. Para el botón cancelar se usa un FlatButton, mientras que para el de ingresar se usa un RaisedButton, esto es para resaltar la importancia del segundo botón. Por ahora las funciones de onPressed() se dejan en blanco, pero se dejan los paréntesis para que no se deshabiliten.

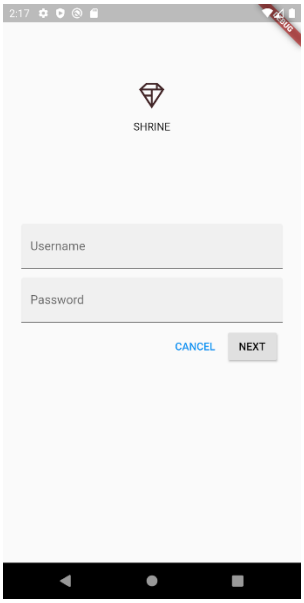


Ilustración 2 Vista de inicio de sesión con botones para cancelar e iniciar sesión

El siguiente paso consiste en añadir un TextEditingController por cada TextField para controlar su contenido. Estos se añaden dentro del campo controller de cada TextField. Dentro de la función onPressed() del botón de cancelar se añade dicho controller, y se usar .clear() para limpiar el contenido de ambos TextField.

Para la función onPressed() del botón Next se utiliza el Navigator (que mantiene una pila de las rutas) junto con la función de pop(context) para salir de la vista actual. Después de esto dentro de home.dart se añade el campo de resizeToAvoidBottomInset como falso, para evitar que el teclado no altere el tamaño de la página ni de los widgets.

MDC-102 Flutter [2]

El primer paso consiste en añadir un widget de AppBar. Esta se añade dentro del Scaffold en el campo de AppBar. Dentro de este widget, se añade el campo leading (izquierda) donde se añade un IconButton que funcionará como un botón de menú, además se añade un título con el campo title. Seguido del título se añaden dos IconButton para la función de buscar y de filtrar.

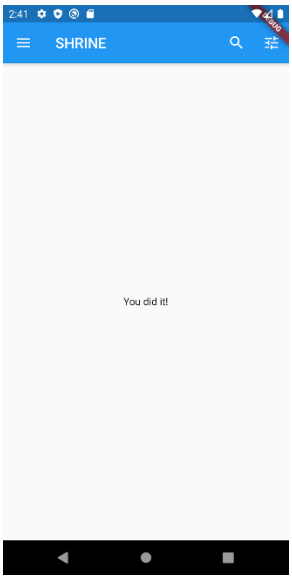


Ilustración 3 Home screen con app bar integrada

Ahora en el body de Scaffold cambiamos en widget de Center por un GridView, dentro del cual se agrega el campo de crossAxisCountque indica cuantos elementos se muestran. Los children del GridView se componen de un Widget Card, dentro del cual añadiremos un child tipo Column, para acomodar los widgets de forma vertical. Finalmente se anida una imagen dentro de un AspectRatio para acomodar la imagen, junto con un Padding que a su vez tiene dos Text .

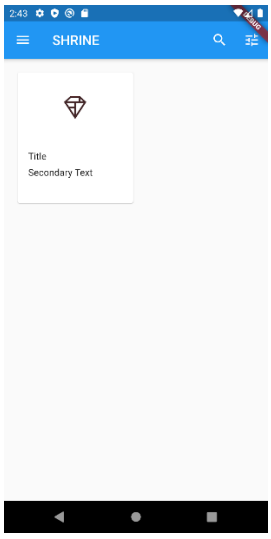


Ilustración 4 Vista del primer Card creado

El siguiente paso consiste en hacer una colección de Card. Para esto se crea una función que crea una lista de cartas, la cuál se añade como el children del GridView.

El código anterior muestra una colección de cartas en la vista de home, cada una con una imagen, un título y un texto secundario. Ahora generaremos estas tarjetas utilizando las imágenes e información de cada producto que tenemos en nuestro proyecto. Antes de eso es importante importar dichos elementos. A continuación, se muestra como importarlos y el código de la función:

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

import 'model/products_repository.dart';
import 'model/product.dart';
```

```
List<Card> _buildGridCards(BuildContext context) {
  List<Product> products = ProductsRepository.loadProducts(Category.all);

  if (products == null || products.isEmpty) {
    return const <Card>[];
  }

  final ThemeData theme = Theme.of(context);
  final NumberFormat formatter = NumberFormat.simpleCurrency(
      locale: Localizations.localeOf(context).toString());

  return products.map((product) {
    return Card(
      clipBehavior: Clip.antiAlias,
      // TODO: Adjust card heights (103)
      child: Column(
        // TODO: Center items on the card (103)
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          AspectRatio(
            aspectRatio: 18 / 11,
            child: Image.asset(
              product.assetName,
              package: product.assetPackage,
             // TODO: Adjust the box size (102)
            ),
          ),
          Expanded(
            child: Padding(
              padding: EdgeInsets.fromLTRB(16.0, 12.0, 16.0, 8.0),
              child: Column(
               // TODO: Align labels to the bottom and center (103)
                crossAxisAlignment: CrossAxisAlignment.start,
                // TODO: Change innermost Column (103)
                children: <Widget>[
                 // TODO: Handle overflowing labels (103)
                  Text(
                    product.name,
                    style: theme.textTheme.headline6,
                    maxLines: 1,
                  ),
                  SizedBox(height: 8.0),
                  Text(
```

```
                    formatter.format(product.price),
                    style: theme.textTheme.subtitle2,
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }).toList();
}
```

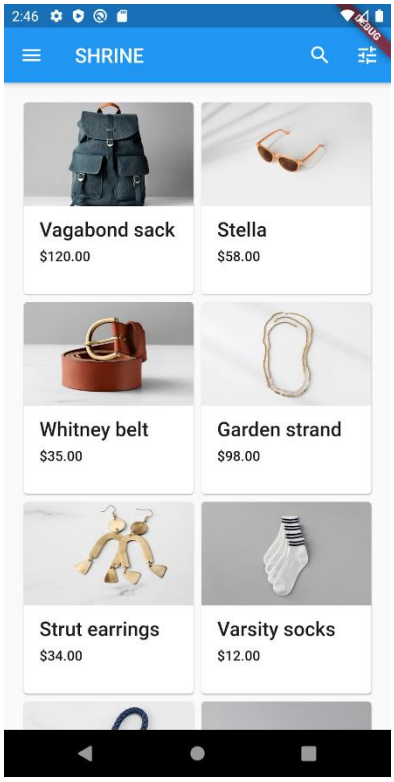


Ilustración 5 Home page con las cartas de cada elemento

MDC-103 Flutter [3]

El principal objetivo de este Code Lab es el manejo de colores y temas. Lo primero que realizamos fue crear un archivo colors.dart con los colores que usaremos para la aplicación. Después creamos una función dentro del archivo app.dart (pero fuera del scope de esta clase) utilizada para crear un tema que puede ser aplicado a diferentes elementos con el campo de theme. La función de copWith() se encarga de crear una instancia del widget que concuerde con la instancia que lo está llamando pero con los valores especificados modificados.

```
// TODO: Build a Shrine Theme (103)
final ThemeData _kShrineTheme = _buildShrineTheme();

ThemeData _buildShrineTheme() {
  final ThemeData base = ThemeData.light();
  return base.copyWith(
    accentColor: kShrineBrown900,
    primaryColor: kShrinePink100,
    buttonTheme: base.buttonTheme.copyWith(
      buttonColor: kShrinePink100,
      colorScheme: base.colorScheme.copyWith(
        secondary: kShrineBrown900,
      ),
    ),
    buttonBarTheme: base.buttonBarTheme.copyWith(
      buttonTextTheme: ButtonTextTheme.accent,
    ),
    scaffoldBackgroundColor: kShrineBackgroundWhite,
    cardColor: kShrineBackgroundWhite,
    textSelectionColor: kShrinePink100,
    errorColor: kShrineErrorRed,
    // TODO: Add the text themes (103)
    // TODO: Add the icon themes (103)
    // TODO: Decorate the inputs (103)
  );
}
```

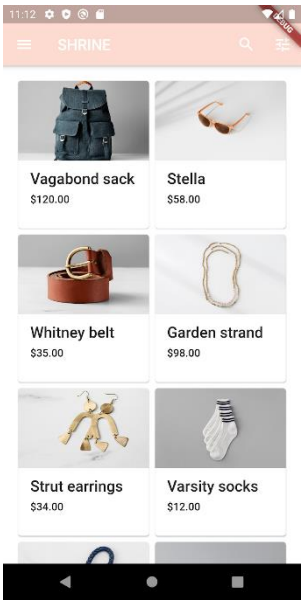


Ilustración 6 Home page con el tema aplicado

Se puede crear temas también para texto utilizando TextTheme de forma similar a ThemeData. TextTheme es útil para asegurarnos que todo el texto de la aplicación sea consistente y legible.

```
TextTheme _buildShrineTextTheme(TextTheme base) {
  return base.copyWith(
    headline5: base.headline5.copyWith(
      fontWeight: FontWeight.w500,
    ),
    headline6: base.headline6.copyWith(
        fontSize: 18.0
    ),
    caption: base.caption.copyWith(
      fontWeight: FontWeight.w400,
      fontSize: 14.0,
    ),
    bodyText1: base.bodyText1.copyWith(
      fontWeight: FontWeight.w500,
      fontSize: 16.0,
    ),
  ).apply(
    fontFamily: 'Rubik',
    displayColor: kShrineBrown900,
    bodyColor: kShrineBrown900,
  );
}
```

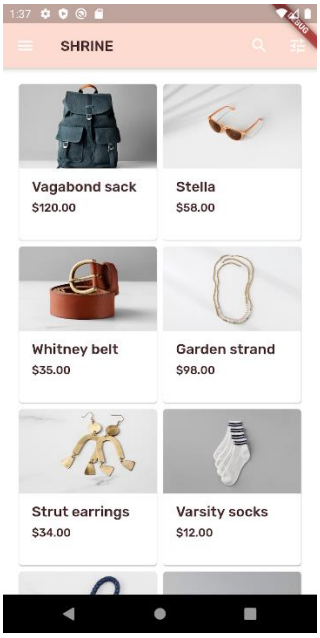


Ilustración 7 Home page con TextTheme implementado

Los siguientes pasos consisten en agregar el parámetro elevation para las tarjetas y el botón de Next, este campo cambia la altura de los elementos en la vista, al tener una elevación de 0 se elimina la sombra, pero si se va aumentando se añade la sombra del elemento.

En la pantalla de inicio de sesión se modifica la forma de los botones usando el campo de shape, añadiendo un BeveledRectangleBorder, dentro del cual en el campo de borderRadius se añade un borde circular para cambiar la forma de los botones.

El siguiente paso consisten en cambiar el layout de las cartas que tenemos en el home page. Primero cambiaremos el GridView a un AsymmetricView, para modificar como se despliegan las cartas. Ahora la lista se encuentra de forma horizontal en lugar de vertical.

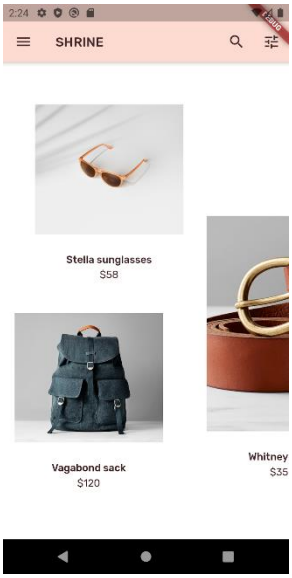


Ilustración 8 Home page con AsymmetricView en lugar de GridView

La última sección del Code Lab muestra como cambiar entre diferentes temas de la aplicación. Esto se puede hacer modificando el campo de brightness a dark.

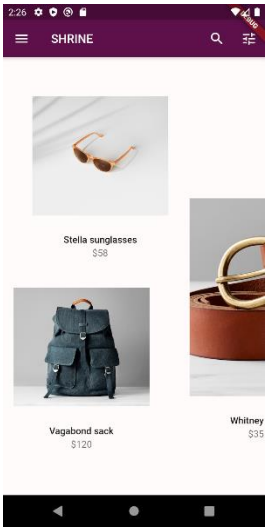


Ilustración 9 Ejemplo de la aplicación con dark theme

MDC-104 Flutter [4]

En este Code Lab se crea un menú de tipo backdrop. Lo primero es ir a home.dart y su función build, y regresar ahora una AsymmetricView. Seguido de esto se crea un archivo backdrop.dart que será utilizado para mostrar este menú. Es importante resaltar que la propiedad de required para evitar que el constructor deje valores nulos.

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

import 'model/product.dart';

// TODO: Add velocity constant (104)

class Backdrop extends StatefulWidget {
  final Category currentCategory;
  final Widget frontLayer;
  final Widget backLayer;
  final Widget frontTitle;
  final Widget backTitle;

  const Backdrop({
    @required this.currentCategory,
    @required this.frontLayer,
    @required this.backLayer,
    @required this.frontTitle,
    @required this.backTitle,
  })  : assert(currentCategory != null),
        assert(frontLayer != null),
        assert(backLayer != null),
        assert(frontTitle != null),
        assert(backTitle != null);

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

// TODO: Add _FrontLayer class (104)
// TODO: Add _BackdropTitle class (104)
// TODO: Add _BackdropState class (104)
```

Seguido de esto se importa el código de backdrop.dart dentro de app.dart para poder usarlo. Ahora cambiamos el campo de home con un widget de tipo Backdrop. Al correr la aplicación se muestra similar a la aplicación que teníamos anteriormente, pero ahora con el código para usar un menú de backdrop listo.

El siguiente paso es añadir una figura dentro de backdrop.dart se crea una nueva clase _FrontLayer, que crea un borde de forma como un rombo.

```
// TODO: Add _FrontLayer class (104)
class _FrontLayer extends StatelessWidget {
  // TODO: Add on-tap callback (104)
  const _FrontLayer({
    Key key,
    this.child,
  }) : super(key: key);

  final Widget child;
```

```
@override
Widget build(BuildContext context) {
  return Material(
    elevation: 16.0,
    shape: BeveledRectangleBorder(
      borderRadius: BorderRadius.only(topLeft: Radius.circular(46.0)),
    ),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.stretch,
      children: <Widget>[
        // TODO: Add a GestureDetector (104)
        Expanded(
          child: child,
        ),
      ],
    ),
  );
 }
}
```

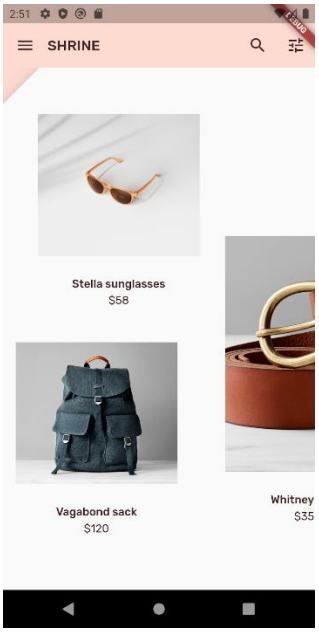


Ilustración 10 Figura para desplegar el menú de backdrop

Después se crea un AnimationController usado para animar el menú de backdrop. Este controlador se usa dentro de la función de initState() que se encuentra en backdrop.dart, también se usa en la función de dispose(). El AnimationController coordina las animaciones y ofrece una API para modificar la animación.

Seguido de estas modificaciones, ahora envolvemos el producto en una ListView, esto se logra modificando el archivo de product_columns.dart, cambiando el widget que regresa de Column a ListView. Una propiedad de la columna es MainAxisAlignment.end que hace que el layout empiece del final. ListView tiene un campo similar llamado reverse.

Lo siguiente es añadir el menú dentro del backdrop. Para esto creamos un nuevo archivo llamado category_menu_page.dart que es un GestureDetector envolviendo una columna.

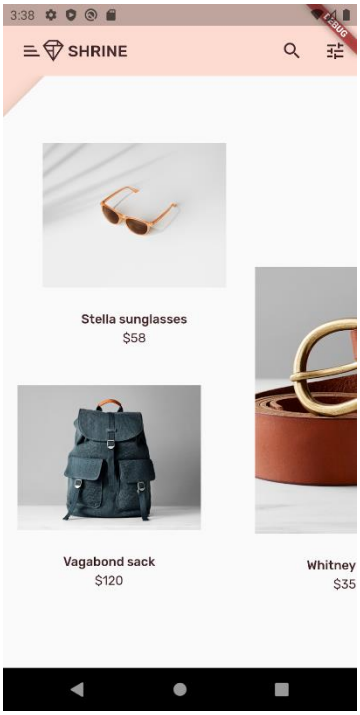
```dart
import 'package:flutter/material.dart';
import 'package:meta/meta.dart';

import 'colors.dart';
import 'model/product.dart';

class CategoryMenuPage extends StatelessWidget {
  final Category currentCategory;
  final ValueChanged<Category> onCategoryTap;
  final List<Category> _categories = Category.values;

  const CategoryMenuPage({
    Key key,
    @required this.currentCategory,
    @required this.onCategoryTap,
  })  : assert(currentCategory != null),
        assert(onCategoryTap != null);

  Widget _buildCategory(Category category, BuildContext context) {
    final categoryString =
        category.toString().replaceAll('Category.', '').toUpperCase();
    final ThemeData theme = Theme.of(context);

    return GestureDetector(
      onTap: () => onCategoryTap(category),
      child: category == currentCategory
          ? Column(
              children: <Widget>[
                SizedBox(height: 16.0),
                Text(
                  categoryString,
                  style: theme.textTheme.bodyText1,
                  textAlign: TextAlign.center,
                ),
                SizedBox(height: 14.0),
                Container(
                  width: 70.0,
                  height: 2.0,
                  color: kShrinePink400,
                ),
              ],
            )
          : Padding(
              padding: EdgeInsets.symmetric(vertical: 16.0),
              child: Text(
                categoryString,
                style: theme.textTheme.bodyText1.copyWith(
                    color: kShrineBrown900.withAlpha(153)
                ),
                textAlign: TextAlign.center,
              ),
            ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        padding: EdgeInsets.only(top: 40.0),
        color: kShrinePink100,
        child: ListView(
          children: _categories
              .map((Category c) => _buildCategory(c, context))
              .toList()),
      ),
    );
  }
}
```

Seguido de esto, cambiamos el widget de la aplicación de Stateless a Stateful, usando el comando de control + punto. Después importamos el código category_menu_page.dart y se modifica el campo de backLayer a un CategoryMenuPage:

```
home: Backdrop(
      // TODO: Make currentCategory field take _currentCategory (104)
      currentCategory: _currentCategory,
      // TODO: Pass _currentCategory for frontLayer (104)
      frontLayer: HomePage(),
      // TODO: Change backLayer field value to CategoryMenuPage (104)
      backLayer: CategoryMenuPage(
        currentCategory: _currentCategory,
        onCategoryTap: _onCategoryTap,
      ),
      frontTitle: Text('SHRINE'),
      backTitle: Text('MENU'),
    ),
```

Ilustración 11 Backdrop menú en home page sin errores

Ahora modificamos el archivo backdrop.dart para que la función de didUpdateWidget cambie la visibilidad del menú backdrop cuando sea seleccionado. La última parte de este Code Camp crea una nueva clase llamada _BackdropTitle, con lo que se añade un ícono con animación en la AppBar del home page.



El código que desarrollé para estos ejercicios se encuentra en mi proyecto de Git [5]. Los Code Lab 101 a 103 se encuentran en la carpeta mdc_100_series, mientras que el Code Lab 104 se encuentra en la carpeta mdc_104_series.

Conclusiones

La tarea me ayudó a comprender diferentes formas de utilizar los Material Components de Flutter. Aprendí una forma distinta a la vista en clase de como crear una lista, y como modificar la apariencia de la app bar, como definir temas tanto con los colores utilizados como con el estilo de la tipografía y el layout de los elementos.

El último Code Lab me enseñó a crear un backdrop, como animarlo y como filtrar los elementos a mostrar, al cambiar del tipo de categoría seleccionado. Algo que me llamó mucho la atención fue como animar el backdrop, para que se muestre como se despliega.

Observaciones

Lo que me costó más trabajo fue importar el código del primer Code Lab, ya que tuve problemas con como se compilaba el proyecto en Visual Studio Code. Para arreglar este error, bastó con abrir la carpeta en Visual Studio Code y editar el archivo pubspec.yaml (se agregó un espacio y se guardó el archivo), y de esta forma se volvieron a cargar las dependencias del proyecto de forma correcta. Como los siguientes Code Labs utilizaron el código del ejercicio 1, no volví a tener este problema.

Otra complicación que surgió fue al momento de implementar el cuarto Code Lab, ya que al final del Code Lab 3 hicimos algunos cambios al código para ver diferentes estilos y temas que podríamos aplicarle. Bastó con revisar las últimas secciones del Code Lab 3 y deshacer el código que modificaba el tema de la aplicación.

Referencias

[1] "MDC-101 Flutter: Material Components (MDC) Basics (Flutter)", Codelabs.developers.google.com, 2020. [Online]. Available: https://codelabs.developers.google.com/codelabs/mdc-101-flutter/#0. [Accessed: 30- Aug- 2020].

[2] "MDC-102 Flutter: Material Structure and Layout (Flutter)", Codelabs.developers.google.com, 2020. [Online]. Available: https://codelabs.developers.google.com/codelabs/mdc-102-flutter/#0. [Accessed: 30- Aug- 2020].

[3] "MDC-103 Flutter: Material Theming with Color, Shape, Elevation, and Type (Flutter)", Codelabs.developers.google.com, 2020. [Online]. Available: https://codelabs.developers.google.com/codelabs/mdc-103-flutter/#0. [Accessed: 30- Aug- 2020].

[4] "MDC-104 Flutter: Material Advanced Components (Flutter)", Codelabs.developers.google.com, 2020. [Online]. Available: https://codelabs.developers.google.com/codelabs/mdc-104-flutter/#0. [Accessed: 30- Aug- 2020].

[5] D. Barragán Alvarez, "DanielBarragan96/android_tarea2p_mdc", GitHub, 2020. [Online]. Available: https://github.com/DanielBarragan96/android_tarea2p_mdc. [Accessed: 30- Aug- 2020].