Tienda online de bolsos - Práctica 3

Descripción general del proyecto

Esta aplicación permite a los usuarios explorar diferentes productos en una interfaz simple. En la aplicación se ha implementado el Navigator 2.0 además de modificar los apartados donde el sistema de navegación estándar era utilizado. A continuación se explicarán las funciones implementadas.

Explicación de funciones implementadas

1. **Creación de AppRouteInformationParser**

Se ha creado la clase de AppRouteInformationParser que hereda de RouteInformationParser cuya utilidad está en la conversión de información de rutas a un objeto de datos que representa el estado de la ruta en la aplicación.

```
Unset
class AppRouteInformationParser extends
RouteInformationParser<RouteSettings> {
  @override
  Future<RouteSettings> parseRouteInformation(RouteInformation
routeInformation) async {

    final uri = routeInformation.uri;
    //TODO Tasca 3 - Implemetar Navigator 2.0
    if(uri.pathSegments.isEmpty) return RouteSettings(name:
'/');
    if(uri.pathSegments.length == 1 && uri.pathSegments[0] ==
'login') return RouteSettings(name: '/login');
    if(uri.pathSegments.length == 1 && uri.pathSegments[0] ==
'products') return RouteSettings(name: '/products');
    if(uri.pathSegments.length == 2 && uri.pathSegments[0] ==
'productDetail'){
      final productJsonString = uri.pathSegments[1];
      final product =
Product.fromJsonString(productJsonString);
```

```
      return RouteSettings(name: '/productDetail', arguments:
product);
    }
    return RouteSettings(name: '/');
  }

  @override
  RouteInformation? restoreRouteInformation(RouteSettings
configuration)
  {
    //TODO Tasca 3 - Implemetar Navigator 2.0
    if (configuration.name == '/productDetail') {
      final product = configuration.arguments as Product;
      final productJsonString = product.toJsonString();
      return RouteInformation(uri:
Uri.parse('/productDetail/$productJsonString'));
    }
    return RouteInformation(uri: Uri.parse(configuration.name
?? '/'));
  }
}
```

2. **Creación de AppRouterDelegate**

Se ha creado la clase de AppRouterDelegate que hereda de RouterDelegate cuya utilidad está en la construcción de la interfaz en base al estado actual de la aplicación y las rutas proporcionadas por el RouteInformationParser. A parte de eso se ha añadido una clase extra para poder crear transiciones personalizadas a cada página. En el caso del proyecto se ha decidido escoger una transición de Fade.

```
Unset
class AppRouterDelegate extends RouterDelegate<RouteSettings>
    with ChangeNotifier, PopNavigatorRouterDelegateMixin<RouteSettings> {
  final GlobalKey<NavigatorState> navigatorKey;
  RouteSettings? _currentRoute;

  AppRouterDelegate() : navigatorKey = GlobalKey<NavigatorState>();

  RouteSettings? get currentConfiguration => _currentRoute;
```

```
  void _setNewRoutePath(RouteSettings settings) {
    _currentRoute = settings;
    notifyListeners();
  }

  @override
  Future<void> setNewRoutePath(RouteSettings configuration) {
    _setNewRoutePath(configuration);
    return SynchronousFuture<void>(null);
  }

  @override
  Widget build(BuildContext context) {
    return Navigator(
      key: navigatorKey,
      pages: [
        //TODO Tasca 3 - Implemetar Navigator 2.0
        CustomTransitionPage(key: ValueKey('IntroScreen'),
            child: IntroScreen()),
        if(_currentRoute?.name == '/login')
          CustomTransitionPage(key: ValueKey('LoginScreen'),child:
LoginScreen()),
        if(_currentRoute?.name == '/products')
          CustomTransitionPage(key: ValueKey('ProductsScreen'),child:
ProductListScreen()),
        if(_currentRoute?.name == '/productDetail')
          CustomTransitionPage(key: ValueKey('ProductDetailScreen'), child:
ProductDetailScreen(_currentRoute!.arguments as Product)),
      ],
      onPopPage: (route, result) {
        if (!route.didPop(result)) {
          return false;
        }

        if(_currentRoute?.name == '/productDetail')
          {
            _setNewRoutePath(RouteSettings(name: '/products'));
          }
        else if(_currentRoute?.name == '/products')
          {
            _setNewRoutePath(RouteSettings(name: '/login'));
          }
        else
          {
            _setNewRoutePath(RouteSettings(name: '/'));
          }
```

```
        return true;
      },
    );
  }
}
```

```
Unset
class CustomTransitionPage extends Page {
  final Widget child;

  CustomTransitionPage({required LocalKey key,required  this.child})
  : super(key: key);
  @override
  Route createRoute(BuildContext context) {
    return PageRouteBuilder(
        settings: this,
        pageBuilder: (context, animation, secondaryAnimation) => child,
        transitionsBuilder: (context, animation, secondaryAnimation, child)
{
          return FadeTransition(opacity: animation, child: child,);
        }
    );
  }
}
```

3. Página de Inicio

En la página de inicio, para la implementación del Navigator 2.0 se ha tenido que hacer dos cambios al apartado de los botones. Al presionar el botón tiene que haber un cambio de página, cambiando la ruta adecuadamente. En este caso hay dos botones, uno que te lleva a la página de productos y otro que te lleva a la página de login.

```
Unset
FilledButton(
                  style: FilledButton.styleFrom(
                    minimumSize: const Size(double.infinity, 48.0),
                  ),
                  onPressed: ()
                  {
```

```
                    //TODO Tasca 3 - Implemetar Navigator 2.0
                    //Ir a la pantalla de login_screen.dart
                    final routerDelegate =
Router.of(context).routerDelegate;
                    routerDelegate.setNewRoutePath(RouteSettings(name:
'/login'));

                  },
                  child: Text(
                    'Login',
                    style: textTheme.titleMedium!.copyWith(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                )
```

Unset

```
FilledButton(
                  style: FilledButton.styleFrom(
                    minimumSize: const Size(double.infinity, 48.0),
                  ),
                  onPressed: ()
                  {

                    //TODO Tasca 3 - Implemetar Navigator 2.0
                    //Ir a la pantalla de product_list_screen.dart
                    final routerDelegate =
Router.of(context).routerDelegate;
                    routerDelegate.setNewRoutePath(RouteSettings(name:
'/products'));

                  },
                  child: Text(
                    'Shop as a Guest',
                    style: textTheme.titleMedium!.copyWith(
                      color: Colors.white,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                )
```

4. Página de Login

En la página de login hay un botón que al rellenar los datos de login te lleva a la página de productos (con los datos correctos) o aparece un popup informando de un error en los datos de login introducidos. En ambos casos hay que implementar lo creado en los apartados anteriores. En el caso del popup con realizar un pop será redirigido a la página en la que se estaba anteriormente.

```
Unset
if (username == 'user' && password == 'password') {

      //TODO Tasca 3 - Implemetar Navigator 2.0
      //Ir a la pantalla de ProductListScreen()
      final routerDelegate = Router.of(context).routerDelegate;
      routerDelegate.setNewRoutePath(RouteSettings(name: '/products'));

    } else {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: Text('Login Failed'),
            content: Text('Invalid username or password. Please try
again.'),
            actions: [
              TextButton(
                onPressed: ()
                {
                  //TODO Tasca 3 - Implemetar Navigator 2.0
                  //Quitar el AlertDialog de la pila
                  Navigator.pop(context);
                },
                child: Text('OK'),
              ),
            ],
          )
```

5. **Página de productos**

En la página de productos en la plantilla ya funcionaba, pero utilizaba el Navigator estándar. Por lo tanto se han tenido que realizar cambios para la implementación de Navigator 2.0. Como cada producto de la lista era un widget aparte se ha tenido que modificar el archivo de product_item.dart. Aparte de eso también se le pasaba a la siguiente página un argumento: el producto seleccionado.

```
Unset
onTap: () {
        final routerDelegate = Router.of(context).routerDelegate;
        routerDelegate.setNewRoutePath(RouteSettings(name: '/productDetail',
arguments: product));
      }
```

El producto era recogido en la página de detalle de producto de la siguiente manera:

```
Unset
const ProductDetailScreen(this.product, {super.key});
```

Aparte de esto en la página de productos también hay un botón para volver a la página de login.

```
Unset
onPressed: ()
              {
                //TODO Tasca 3 - Implemetar Navigator 2.0
                //Ir a la pantalla login_screen.dart

                final routerDelegate = Router.of(context).routerDelegate;
                routerDelegate.setNewRoutePath(RouteSettings(name:
'/login'));
              }
```

6. Página de detalle de producto

En este apartado, aparte de lo comentado anteriormente, también hay un botón que redirige al usuario a la página de productos. Para ello se ha utilizado el Navigator estándar utilizando pop como con el popup mencionado en apartados anteriores.

```
Unset
onPressed: ()
              {

                //TODO Tasca 3 - Implemetar Navigator 2.0
```

```
            //Volver a la pantalla de product_list_screen
            Navigator.pop(context);

          }
```