**báo cáo mô tả các bước triển khai dự án**

Họ và tên: Trương Trọng Quý, Mã sv: 21IT164

**1. Tạo project và kết nối Firebase**

Tạo firebase project tại firebase console, sau đó tạo flutter project. Mở terminal và nhập các lệnh:

- firebase login

- dart pub global activate flutterfire\_cli

- flutterfire configure (khúc này chọn firebase project mới tạo và chọn tất cả platform)

- flutter pub add firebase\_core

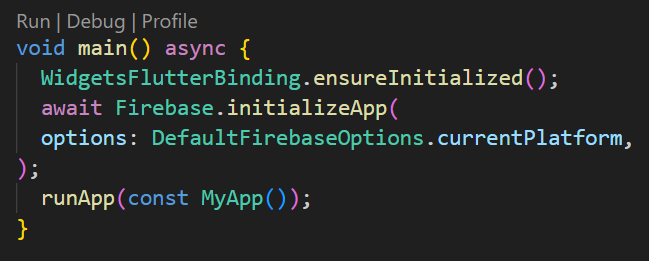
- flutterfire configure

Tiếp tục import tại file main.dart

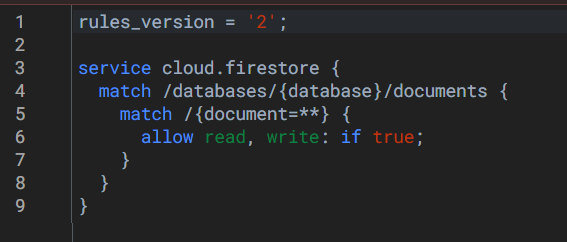
- import 'package:firebase\_core/firebase\_core.dart';

- import 'firebase\_options.dart';

Ngoài ra, trong tệp lib/main.dart, khởi tạo Firebase bằng cách sử dụng đối tượng DefaultFirebaseOptions.

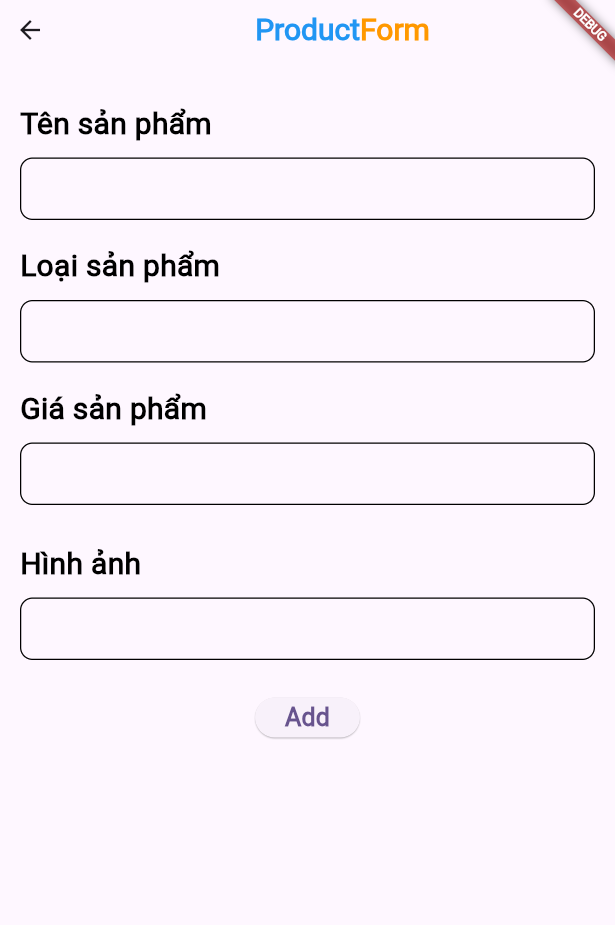


Trong firebase project, tại Rules của Firestore database sửa false thành true.

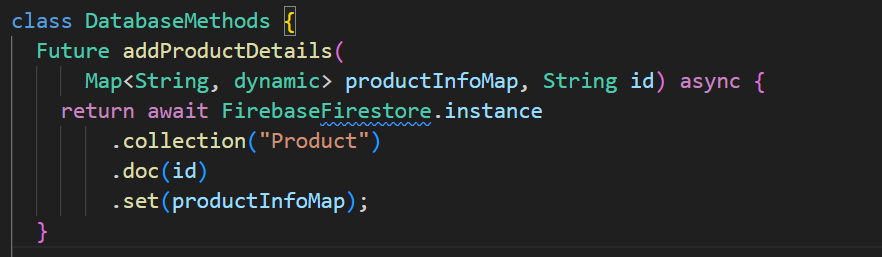


**2. Thêm sản phẩm**

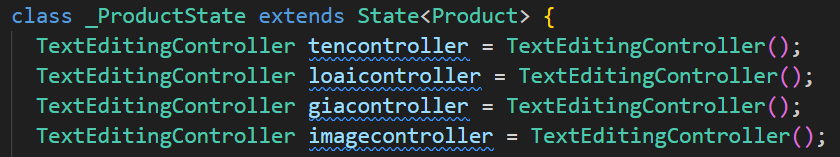
Tạo file product.dart, tạo lớp 1 product kế thừa StatefulWidget sau đó tạo 1 lớp \_ProductState. Trong lớp \_ProductState tạo 3 Textfield Tên sản phẩm, Loại sản phẩm, Giá sản phẩm và nút Button Add.



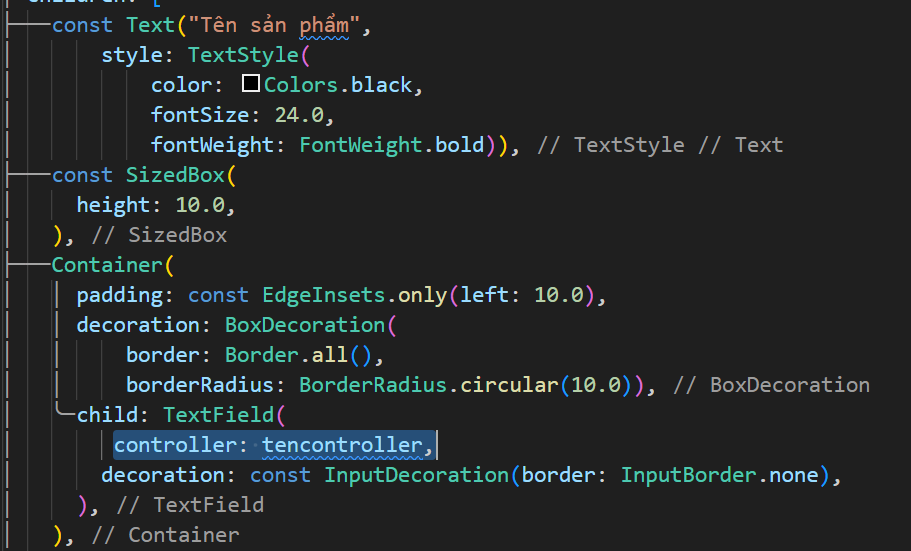
Tạo file database.dart, tạo lớp DatabaseMethods, sau đó tạo 1 hàm để thêm chi tiết sản phẩm vào Firestore, hàm có tên là addProductDetails.



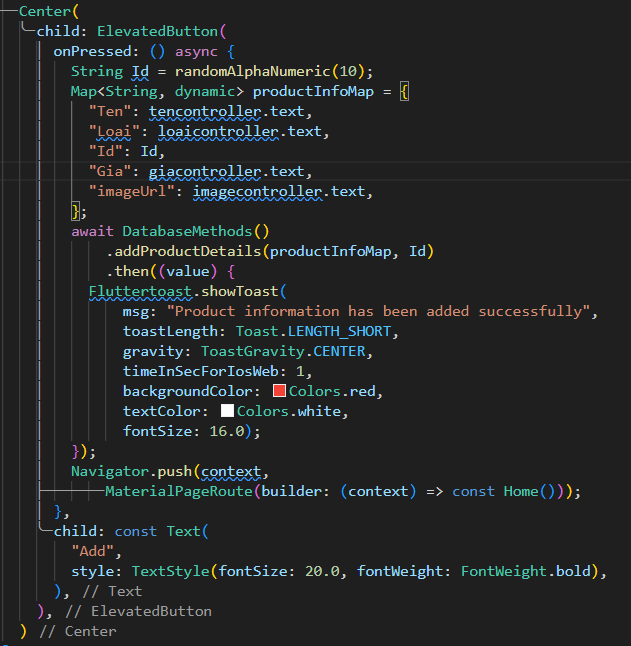
Tại file product.dart khai báo 3 đối tượng TextEditingController, được sử dụng để quản lý và theo dõi các giá trị nhập vào từ người dùng qua các text fields.



Ba biến này sẽ được gán cho các TextField tương ứng trong giao diện ứng dụng.

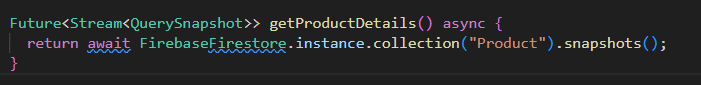


Tại button Add, khai báo biến Id được gán 10 số ngẫu nhiên mỗi khi tạo 1 sản phẩm. Dữ liệu nhập từ trường TextField được quản lý bởi controller và biến Id sẽ được lưu vào khóa “Ten”, “Loai”, “Id”, “Gia”. Sau đó được lưu trong productInfoMap, gọi phương thức addProductDetails từ đối tượng DatabaseMethods để thêm chi tiết sản phẩm vào Firestore. Sau khi sản phẩm được thêm thành công thì hiện thông báo bởi Fluttertoast.showToast()

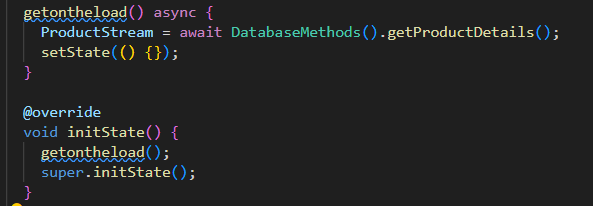


**3. Hiển thị sản phẩm**

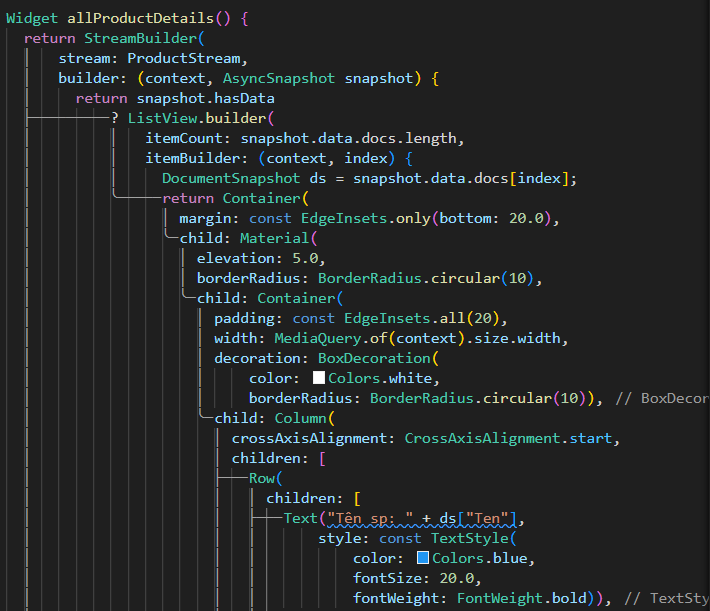
Tại file database.dart tạo hàm getProductDetails() để lấy thông tin sản phẩm từ Firestore.

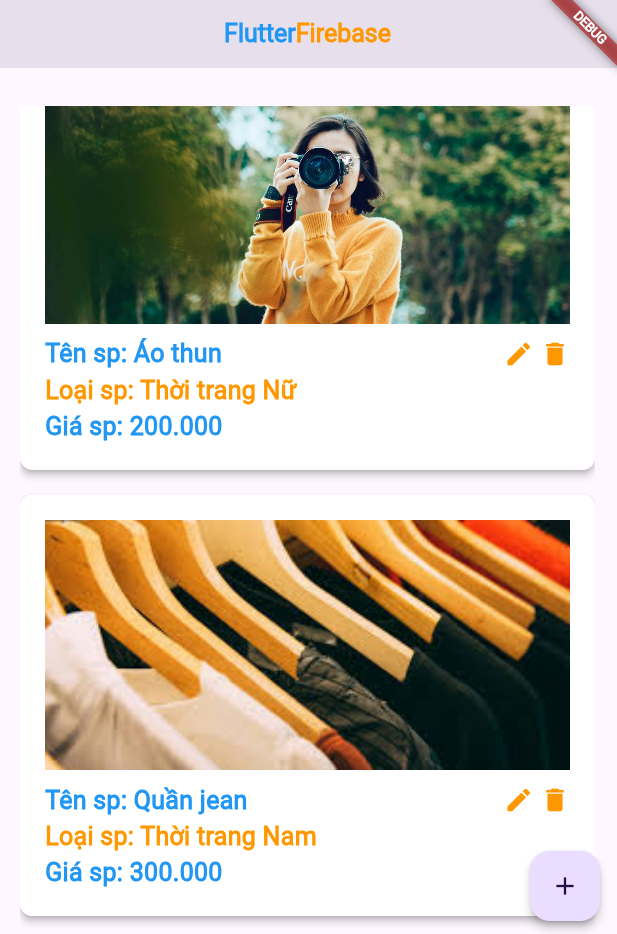


Tại file home.dart tạo hàm getontheload() có nhiệm vụ lấy dữ liệu sản phẩm từ cơ sở dữ liệu gán vào ProductStream. Tạo một phương thức initState() được gọi khi một widget được khởi tạo lần đầu tiên, nó gọi hàm getontheload() để tải dữ liệu sản phẩm ngay khi widget được khởi tạo.



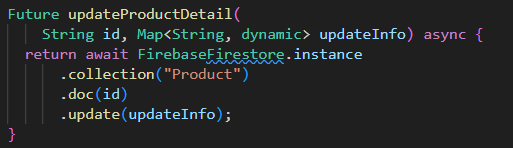
Gọi PoductStream để lắng nghe thay đổi từ cơ sở dữ liệu, nó sẽ trả về snapshot có dữ liệu. Khai báo biến ds gán cho một đối tượng Document Snapshot để lấy dữ liệu mà snapshot đang chứa. Để truy cập các thuộc tính sử dụng ds[“Ten”]. Tạo thêm 2 button dùng để sửa và xóa.



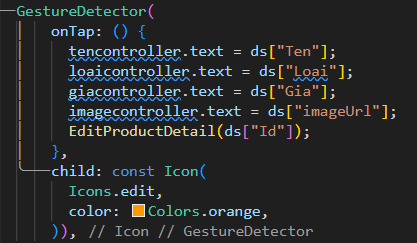


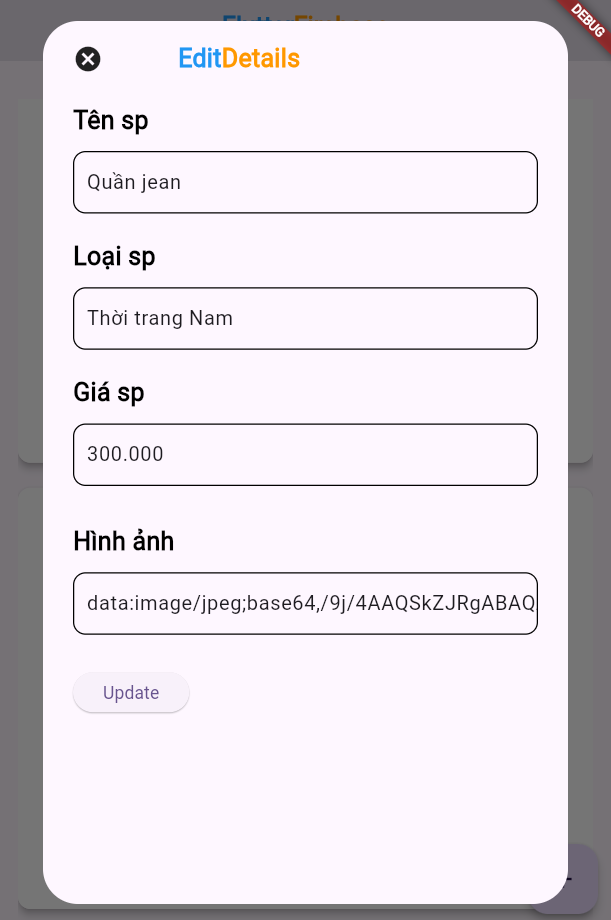
**4. Sửa sản phẩm**

Tại file database.dart tạo hàm updateProductDetail để cập nhật thông tin sản phẩm lên Firestore.

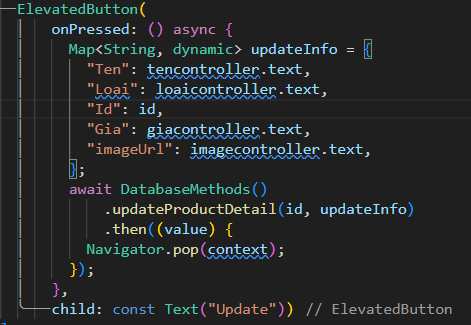


Tại file home.dart tạo hàm EditProductDetail để hiển thị 1 AlertDialog có 3 trường TextField giúp cập nhật sản phẩm. Khai báo 3 đối tượng Text EditingController, khi người dùng nhấn vào biểu tượng chỉnh sửa, giá trị từ các trường "Ten", "Loai", và "Gia" của sản phẩm sẽ được tải vào các text fields thông qua các TextEditingController tương ứng. Sau đó, hàm Edit ProductDetail() sẽ được gọi với ID sản phẩm.



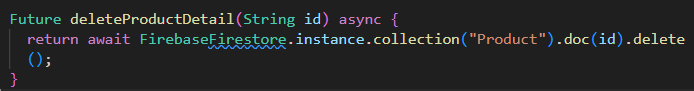


Sau khi cập nhật sản phẩm và ấn button Update, dữ liệu cập nhập từ trường TextField được quản lý bởi controller và biến Id sẽ được lưu vào khóa “Ten”, “Loai”, “Id”, “Gia”. Sau đó được lưu trong updateInfo, gọi phương thức updateProductDetails từ đối tượng DatabaseMethods để cập nhật chi tiết sản phẩm vào Firestore.



**5. Xóa sản phẩm**

Tại file database.dart tạo hàm deleteProductDetail để xóa thông tin sản phẩm trong Firestore.



Ở file home.dart, sau khi nhấn biểu tượng xóa , gọi phương thức deleteProductDetails chứa Id từ đối tượng DatabaseMethods để xóa sản phẩm từ Firestore. Sau khi xóa thành công sẽ thông báo đã xóa thành công.

