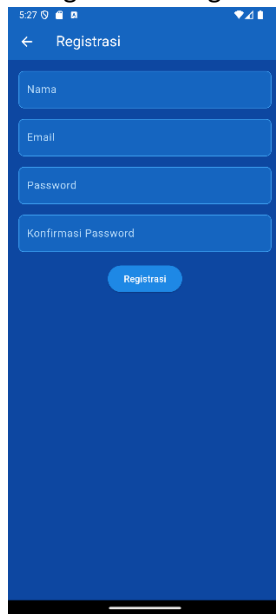


## 1. Proses Registrasi

### a. Mengisi Form Registrasi

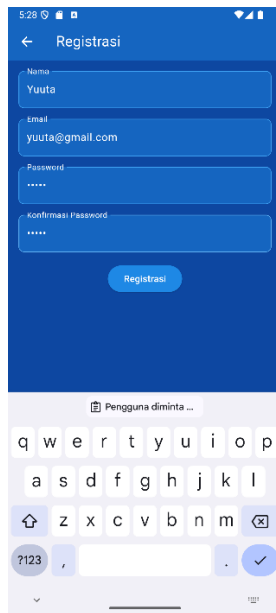


Pengguna diminta untuk mengisi nama, email, password, dan konfirmasi password.

Kode terkait:

```
Widget _namaTextField() {  
  return TextFormField(  
    decoration: const InputDecoration(labelText: "Nama"),  
    keyboardType: TextInputType.text,  
    controller: _namaTextboxController,  
    validator: (value) {  
      if (value!.length < 3) {  
        return "Nama harus diisi minimal 3 karakter";  
      }  
      return null;  
    },  
  );  
}  
  
//... (Kode untuk email, password, dan konfirmasi password)
```

### b. Proses Pengiriman Data Registrasi



Setelah menekan tombol registrasi, aplikasi akan mengirim data ke API.

Kode terkait:

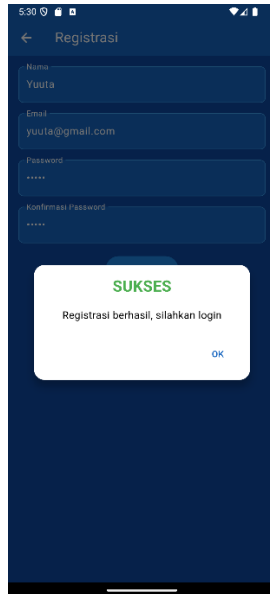
```
void _submit() {
  _formKey.currentState!.save();
  setState(() {
    _isLoading = true;
  });
  RegistrasiBloc.registrasi(
    nama: _namaTextboxController.text,
    email: _emailTextboxController.text,
    password: _passwordTextboxController.text,
  ).then((value) {
    if (value['status']) {
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => SuccessDialog(
          description: "Registrasi berhasil, silahkan login",
          onClick: () {
            Navigator.pop(context);
          },
        ),
      );
    } else {
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => WarningDialog(
          description: value['message'],
        ),
      );
    }
  }).catchError((error) {
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) => const WarningDialog(
        description: "Registrasi gagal, silahkan coba lagi",
      ),
    );
  });
}
```

```

    }).whenComplete(() {
      setState(() {
        _isLoading = false;
      });
    });
  });
}

```

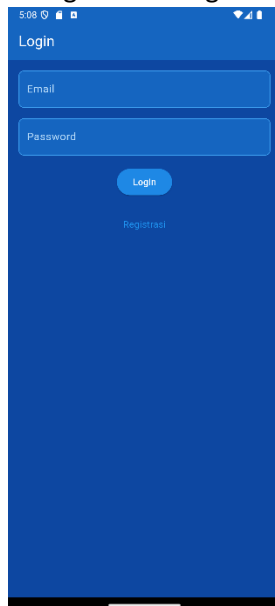
### c. Hasil Registrasi



Pengguna akan melihat popup yang menginformasikan hasil registrasi.

## 2. Proses Login

### a. Mengisi Form Login



Pengguna diminta untuk memasukkan email dan password pada form login.

Kode terkait:

```

Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Email"),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    validator: (value) {
      if (value!.isEmpty) {

```

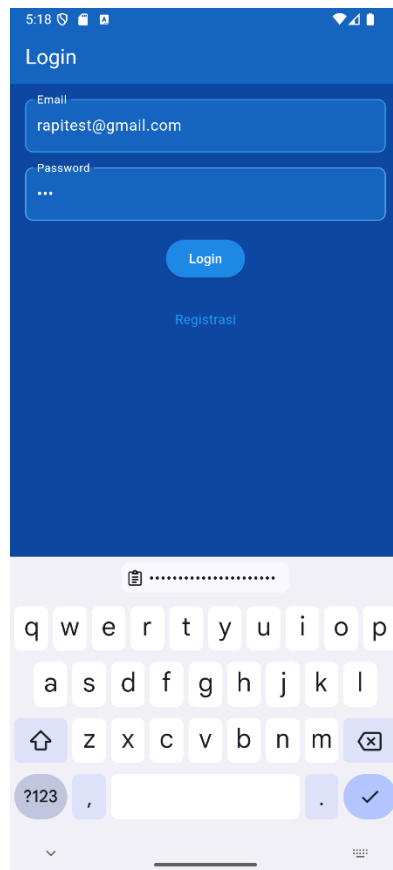
```

        return 'Email harus diisi';
    }
    return null;
  },
);
}

Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Password"),
    keyboardType: TextInputType.text,
    obscureText: true,
    controller: _passwordTextboxController,
    validator: (value) {
      if (value!.isEmpty) {
        return "Password harus diisi";
      }
      return null;
    },
  );
}
}

```

#### b. Proses Autentikasi



Setelah menekan tombol login, aplikasi akan mengirim permintaan ke API untuk melakukan autentikasi.

Kode terkait:

```

void _submit() {
  _formKey.currentState!.save();
  setState(() {
    _isLoading = true;
  });
}

```

```

LoginBloc.login(
  email: _emailTextboxController.text,
  password: _passwordTextboxController.text)
  .then((value) async {
    if (value.code == 200) {
      await UserInfo().setToken(value.token ?? "");
      await UserInfo().setUserID(int.tryParse(value.userID.toString()) ??
0);

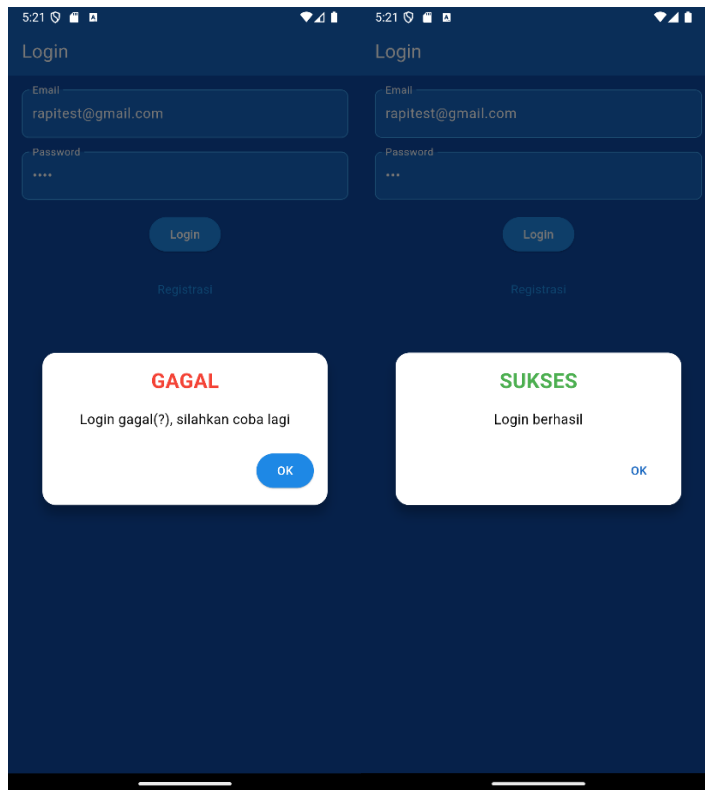
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => SuccessDialog(
          description: "Login berhasil",
          onClick: () {
            Navigator.pushReplacement(
              context,
              MaterialPageRoute(
                builder: (context) => const ProdukPage(),
              ),
            );
          },
        ),
      );
    } else {
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => const WarningDialog(
          description: "Login gagal(?), silahkan coba lagi",
        ));
    }
  }, onError: (error) {
    print(error);
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) => const WarningDialog(
        description: "Login gagal(!), silahkan coba lagi",
      ));
  });

  setState(() {
    _isLoading = false;
  });
}

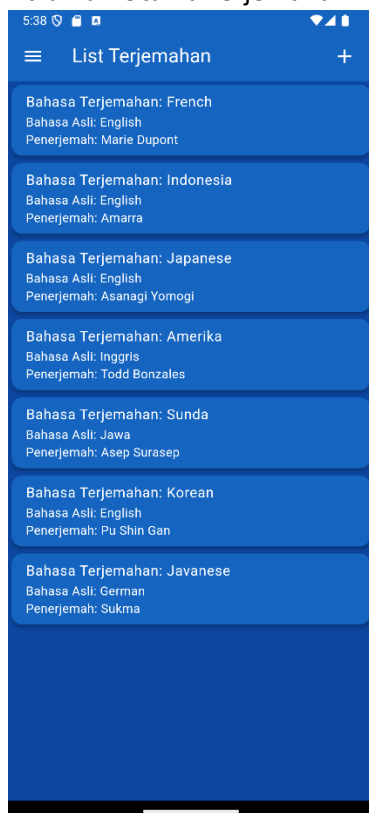
```

### c. Hasil Login

Setelah proses autentikasi, pengguna akan melihat popup yang menginformasikan hasil login.



3. Menampilkan Daftar Terjemahan
  - a. Halaman Utama Terjemahan



Setelah login berhasil, pengguna akan diarahkan ke halaman utama yang menampilkan daftar terjemahan.

Kode terkait:

```

class _ProdukPageState extends State<ProdukPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('List Terjemahan'),
        actions: [
          Padding(
            padding: const EdgeInsets.only(right: 20.0),
            child: GestureDetector(
              child: const Icon(Icons.add, size: 26.0),
              onTap: () async {
                Navigator.push(context,
                  MaterialPageRoute(builder: (context) =>
ProdukForm()));
              },
            ),
          ],
        ),
      drawer: Drawer(
        child: ListView(
          children: [
            ListTile(
              title: const Text('Logout'),
              trailing: const Icon(Icons.logout),
              onTap: () async {
                await LogoutBloc.logout().then((value) => {
                  Navigator.of(context).pushAndRemoveUntil(
                    MaterialPageRoute(builder: (context) =>
LoginPage()),
                      (route) => false);
                });
              },
            ),
          ],
        ),
      ),
      body: FutureBuilder<List>(
        future: ProdukBloc.getProduk(),
        builder: (context, snapshot) {
          if (snapshot.hasError) print(snapshot.error);
          return snapshot.hasData
            ? ListProduk(
                list: snapshot.data,
              )
            : const Center(
                child: CircularProgressIndicator(),
              );
        },
      ),
    );
  }
}

```

b. Proses Pengambilan Data Terjemahan

Data terjemahan diambil dari API menggunakan ProdukBloc.getProduk().

Kode terkait:

```

static Future<List<Produk>> getProduk() async {
  String apiUrl = ApiUrl.listProduk;
  var response = await Api().get(apiUrl);
  var jsonObj = json.decode(response.body);
}

```

```

List<dynamic> listProduk = (jsonObj as Map<String, dynamic>)['data'];
List<Produk> produks = [];
for (int i = 0; i < listProduk.length; i++) {
    produks.add(Produk.fromJson(listProduk[i]));
}
return produks;
}

```

#### 4. Melihat Detail Terjemahan

##### a. Memilih Terjemahan dari Daftar

Pengguna dapat melihat detail terjemahan dengan menekan item terjemahan di daftar.

##### b. Halaman Detail Terjemahan



Halaman ini menampilkan informasi lengkap tentang terjemahan yang dipilih.

Kode terkait:

```

class _ProdukDetailState extends State<ProdukDetail> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Detail Produk'),
      ),
      body: Center(
        child: Column(
          children: [
            Text(
              "Bahasa Asli : ${widget.produk!.original_language}",
              style: const TextStyle(fontSize: 20.0),
            ),
            Text(
              "Bahasa Translasi : ${widget.produk!.translated_language}",
              style: const TextStyle(fontSize: 18.0),
            ),
            Text(
              "Penerjemah : ${widget.produk!.translator_name}",

```



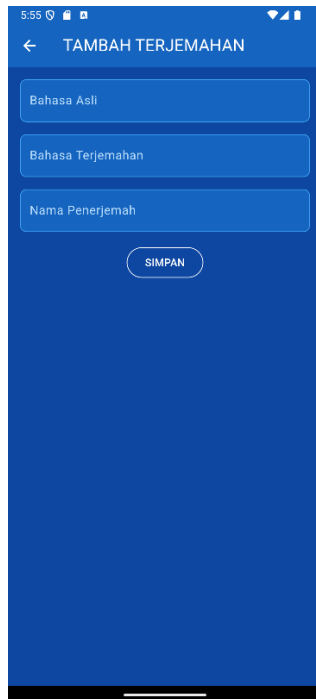
```

        style: const TextStyle(fontSize: 18.0),
      ),
      _tombolHapusEdit()
    ],
  ),
),
);
}

```

## 5. Menambah Terjemahan Baru

### a. Membuka Form Tambah Terjemahan



5:55

← TAMBAH TERJEMAHAN

Bahasa Asli

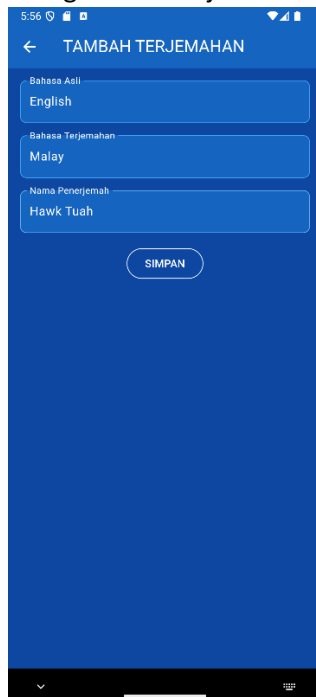
Bahasa Terjemahan

Nama Penerjemah

SIMPAN

Pengguna dapat menambah terjemahan baru dengan menekan ikon "+" di halaman utama.

### b. Mengisi Data Terjemahan Baru



5:56

← TAMBAH TERJEMAHAN

Bahasa Asli

English

Bahasa Terjemahan

Malay

Nama Penerjemah

Hawk Tuah

SIMPAN

Pengguna diminta untuk mengisi bahasa asli, bahasa terjemahan, dan nama penerjemah.

Kode terkait:

```
Widget _kodeProdukTextField() {  
  return TextFormField(  
    decoration: const InputDecoration(labelText: "Bahasa Asli"),  
    keyboardType: TextInputType.text,  
    controller: _originalLanguageTextboxController,  
    validator: (value) {  
      if (value!.isEmpty) {  
        return "Bahasa Asli harus diisi";  
      }  
      return null;  
    },  
  );  
}  
//... (Kode untuk bahasa terjemahan, nama penerjemah)
```

c. Proses Penyimpanan Terjemahan Baru

Setelah menekan tombol simpan, aplikasi akan mengirim data ke API.

Kode terkait:

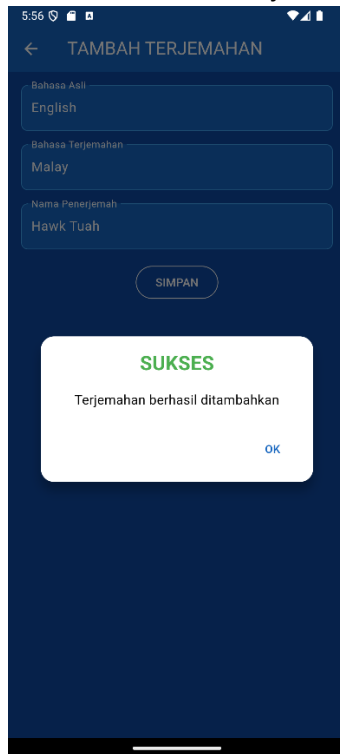
```
simpan() {  
  setState(() {  
    _isLoading = true;  
  });  
  Produk createProduk = Produk(id: null);  
  createProduk.original_language = _originalLanguageTextboxController.text;  
  createProduk.translated_language =  
  _translatedLanguageTextboxController.text;  
  createProduk.translator_name = _translatorNameTextboxController.text;  
  ProdukBloc.addProduk(createProduk).then((value) {  
    if (value['status']) {  
      showDialog(  
        context: context,  
        barrierDismissible: false,  
        builder: (BuildContext context) => SuccessDialog(  
          description: "Terjemahan berhasil ditambahkan",  
          onClick: () {  
            Navigator.of(context).pushReplacement(  
              MaterialPageRoute(  
                builder: (BuildContext context) => const ProdukPage(),  
              ),  
            );  
          },  
        ),  
      );  
    } else {  
      showDialog(  
        context: context,  
        builder: (BuildContext context) => WarningDialog(  
          description: value['message'],  
        ),  
      );  
    }  
  }).catchError((error) {  
    showDialog(  
      context: context,  
      builder: (BuildContext context) => const WarningDialog(  
        description: "Simpan gagal, silahkan coba lagi",  
      ),  
    );  
  });  
}
```

```

    ),
  );
}).whenComplete(() {
  setState(() {
    _isLoading = false;
  });
});
});
}

```

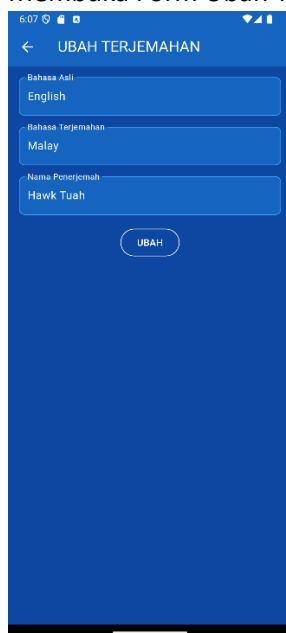
#### d. Hasil Penambahan Terjemahan



Pengguna akan melihat popup yang menginformasikan hasil penambahan terjemahan.

### 6. Mengubah Terjemahan

#### a. Membuka Form Ubah Terjemahan



Dari halaman detail terjemahan, pengguna dapat menekan tombol "EDIT" untuk membuka form ubah terjemahan.

b. Mengisi Perubahan Data Terjemahan

Form ubah terjemahan akan terisi dengan data produk yang ada, dan pengguna dapat mengubahnya.

c. Proses Penyimpanan Perubahan Terjemahan

Setelah menekan tombol ubah, aplikasi akan mengirim data perubahan ke API.

Kode terkait:

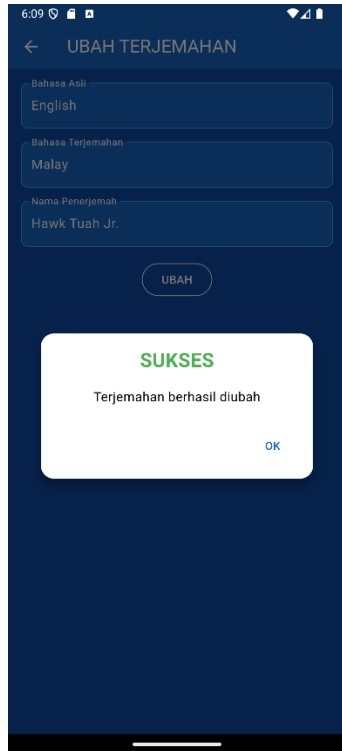
```
ubah() {
  setState(() {
    _isLoading = true;
  });
  Produk updateProduk = Produk(id: widget.produk!.id!);
  updateProduk.original_language = _originalLanguageTextboxController.text;
  updateProduk.translated_language =
    _translatedLanguageTextboxController.text;
  updateProduk.translator_name = _translatorNameTextboxController.text;
  ProdukBloc.updateProduk(produk: updateProduk).then((value) {
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) => SuccessDialog(
        description: "Terjemahan berhasil diubah",
        onClick: () {
          Navigator.of(context).push(
            MaterialPageRoute(
              builder: (BuildContext context) => const ProdukPage(),
            ),
          );
        },
      ),
    );
  }, onError: (error) {
    showDialog(
```

```

    context: context,
    builder: (BuildContext context) => const WarningDialog(
      description: "Permintaan ubah data gagal, silahkan coba lagi",
    ));
  });
  setState(() {
    _isLoading = false;
  });
}

```

d. Hasil Pengubahan Terjemahan



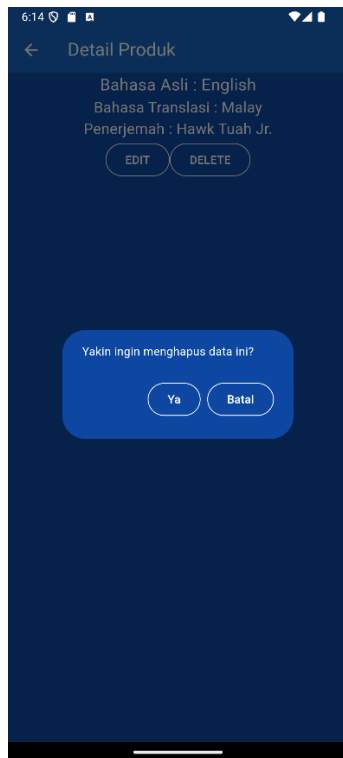
Pengguna akan melihat popup yang menginformasikan hasil perubahan terjemahan.

7. Menghapus Terjemahan

a. Memilih Terjemahan Untuk Dihapus

Dari halaman detail terjemahan, pengguna dapat menekan tombol "DELETE" untuk menghapus terjemahan.

b. Konfirmasi Penghapusan



Sebelum menghapus, aplikasi akan menampilkan dialog konfirmasi.

Kode terkait:

```
void confirmHapus() {  
  if (widget.produk?.id == null) {  
    showDialog(  
      context: context,  
      builder: (BuildContext context) => const WarningDialog(  
        description: "ID produk tidak ditemukan, tidak bisa menghapus.",  
      ),  
    );  
  };  
  return;  
}  
  
AlertDialog alertDialog = AlertDialog(  
  content: const Text("Yakin ingin menghapus data ini?"),  
  actions: [  
    OutlinedButton(  
      child: const Text(  
        "Ya",  
        style: TextStyle(  
          color: Colors.white  
        ),  
      ),  
      onPressed: //Proses Penghapusan  
    ),  
    OutlinedButton(  
      child: const Text(  
        "Batal",  
        style: TextStyle(  
          color: Colors.white  
        ),  
      ),  
      onPressed: () => Navigator.pop(context),  
    ),  
  ],  
);
```

```
],  
);
```

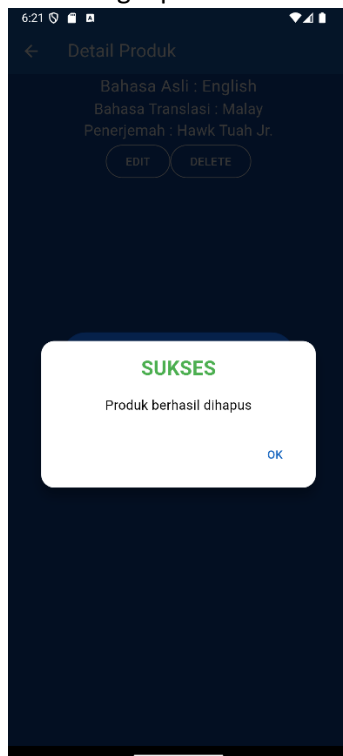
#### c. Proses Penghapusan

Jika pengguna mengkonfirmasi, aplikasi akan mengirim permintaan hapus ke API.

Kode terkait:

```
onPressed: () async {  
  bool success = await ProdukBloc.deleteProduk(  
    id: widget.produk!.id!);  
  if (success) {  
    showDialog(  
      context: context,  
      barrierDismissible: false,  
      builder: (BuildContext context) => SuccessDialog(  
        description: "Produk berhasil dihapus",  
        onClick: () {  
          Navigator.of(context).pushReplacement(  
            MaterialPageRoute(  
              builder: (context) => const ProdukPage(),  
            ),  
          );  
        },  
      );  
    },  
  );  
} else {  
  // Jika penghapusan gagal  
  showDialog(  
    context: context,  
    builder: (BuildContext context) => const WarningDialog(  
      description: "Hapus gagal, silahkan coba lagi",  
    ),  
  );  
}
```

#### d. Hasil Penghapusan



Pengguna akan melihat popup yang menginformasikan hasil penghapusan terjemahan.

## 8. Proses Logout

### a. Memilih Menu Logout



Pengguna dapat mengakses menu logout dari drawer aplikasi.

Kode terkait:

```
drawer: Drawer(  
  child: ListView(  
    children: [  
      ListTile(  
        title: const Text('Logout'),  
        trailing: const Icon(Icons.logout),  
        onTap: () async {  
          await LogoutBloc.logout().then((value) => {  
            Navigator.of(context).pushAndRemoveUntil(  
              MaterialPageRoute(builder: (context) => LoginPage()),  
              (route) => false)  
          });  
        },  
      ),  
    ],  
  ),  
)
```

### b. Proses Logout

Ketika pengguna memilih logout, aplikasi akan menghapus token dan informasi pengguna dari penyimpanan lokal.

Kode terkait:

```
class LogoutBloc {  
  static Future logout() async {  
    await UserInfo().logout();  
  }  
}
```



```
//Di user_info.dart
Future logout() async {
  final SharedPreferences pref = await SharedPreferences.getInstance();
  pref.clear();
}
```

c. Kembali ke Halaman Login

Setelah proses logout selesai, pengguna akan diarahkan kembali ke halaman login.