

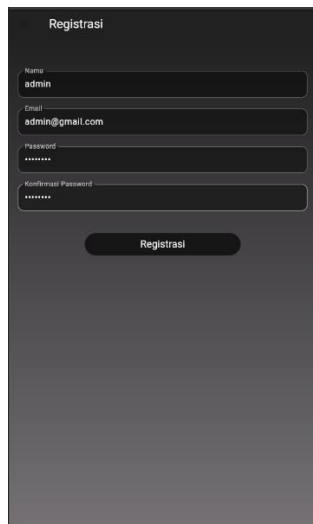
Nama : Muhammad Syaiful Latif
H1D022025
Shift : C

Responsi 1

Aplikasi Manajemen Kesehatan

Tema Monokrom

1. Proses registrasi

A screenshot of a registration form titled "Registrasi". The form has a dark background with light-colored text. It contains four input fields: "Nama" (Name) with the value "admin", "Email" with the value "admin@gmail.com", "Password" with masked characters "*****", and "Konfirmasi Password" (Confirm Password) also with masked characters "*****". Below the fields is a button labeled "Registrasi".

Pengguna mengisi form registrasi dengan nama, email, dan password. Tiga `TextEditingController` digunakan untuk mengelola input pengguna: `_namaTextboxController` untuk nama, `_emailTextboxController` untuk email, `_passwordTextboxController` untuk password

Setiap input memiliki validator untuk memastikan data yang dimasukkan sesuai dengan kriteria: Nama: Minimal 3 karakter. Email: Harus diisi dan valid. Password: Minimal 6 karakter. Konfirmasi Password: Harus sama dengan password yang dimasukkan.

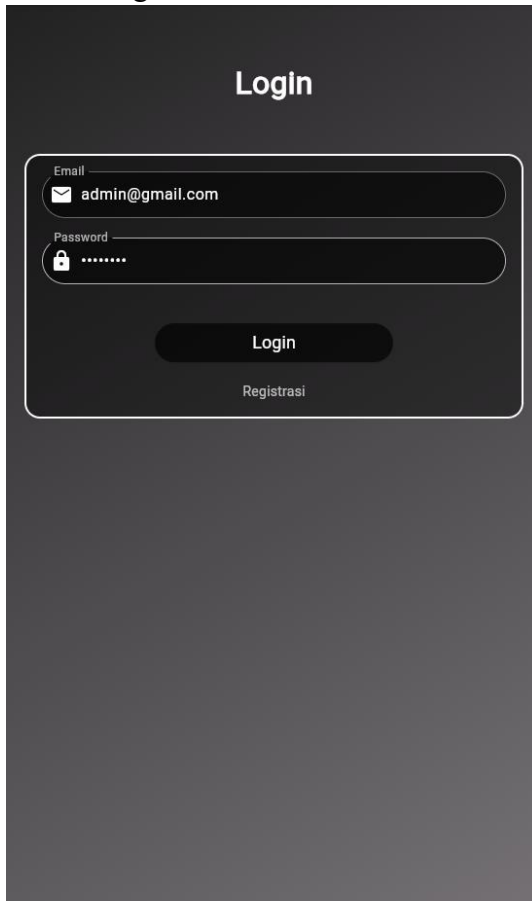
Ketika tombol "Registrasi" ditekan, fungsi `_submit()` dipanggil setelah validasi form berhasil. Jika validasi berhasil dan tombol tidak dalam status loading, proses registrasi dimulai.

Pada proses validasi dan manajemen state, pertama-tama memanggil `save()` pada `_formKey.currentState` untuk menyimpan data input. Selanjutnya, mengubah `_isLoading` menjadi `true` agar indikator loading dapat ditampilkan. Kemudian, memanggil `RegistrasiBloc.registrasi`, yang menggunakan `RegistrasiBloc` untuk mengirimkan permintaan registrasi ke API dengan data yang telah dikumpulkan dari controller, yaitu nama, email, dan password. Setelah itu, kita menangani respons dari API; jika berhasil, kita menampilkan dialog sukses menggunakan `SuccessDialog`, yang memberi tahu pengguna bahwa registrasi telah berhasil dan mereka bisa login. Sebaliknya, jika registrasi gagal, kita menampilkan dialog peringatan menggunakan `WarningDialog` yang memberi tahu pengguna tentang kegagalan registrasi. Terakhir, kita mengatur ulang state dengan mengubah `_isLoading` kembali ke `false` setelah proses selesai, baik itu berhasil maupun gagal.

RegistrasiBloc mengelola logika bisnis untuk registrasi dengan fungsi registrasi yang menerima parameter nama, email, dan password. Dalam proses ini, RegistrasiBloc membuat apiUrl dari ApiUrl.registrasi untuk menentukan endpoint API yang tepat. Selanjutnya, ia menyusun body dari permintaan yang berisi data pengguna. Permintaan POST kemudian dikirimkan ke API menggunakan metode Api().post(). Setelah menerima respons, RegistrasiBloc mengonversi body respons yang diterima dalam format JSON menjadi objek Registrasi dengan menggunakan metode fromJson.

Model Registrasi mendefinisikan struktur data yang akan diterima setelah proses registrasi. Model ini memiliki beberapa atribut, yaitu code yang menyimpan kode respons dari server, status yang menyimpan status proses registrasi (apakah berhasil atau gagal), dan data yang menyimpan informasi tambahan yang mungkin disediakan oleh server. Selain itu, model ini juga dilengkapi dengan metode fromJson yang berfungsi untuk mengonversi data JSON menjadi objek Registrasi.

2. Proses Login

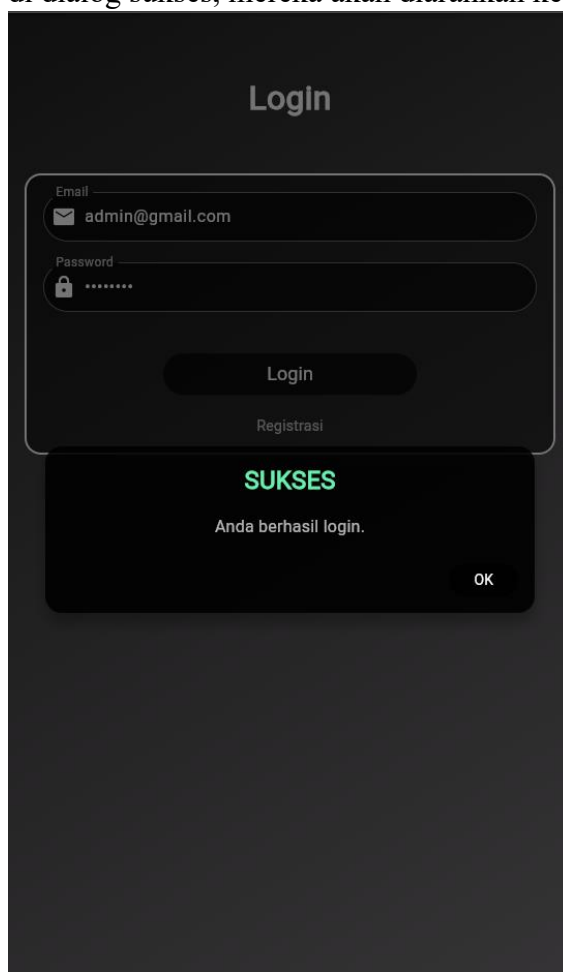


Pengguna memasukkan email dan password di form yang disediakan. Input ini dikelola oleh TextEditingController, yaitu _emailTextboxController untuk email dan _passwordTextboxController untuk password. Validasi dilakukan untuk memastikan kedua field tidak kosong.

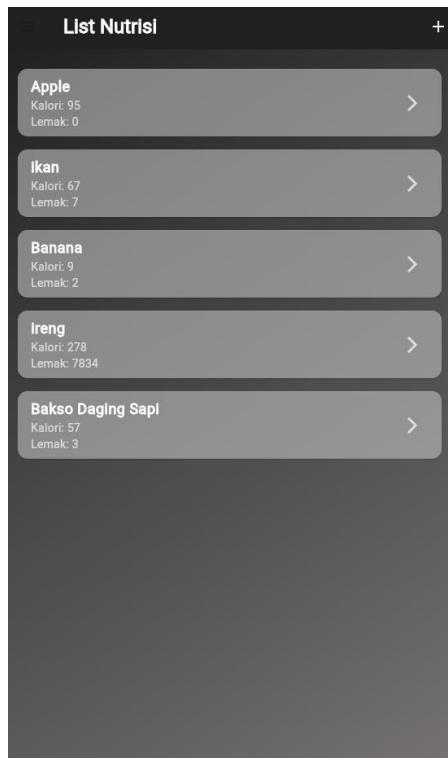
Ketika tombol “Login” diklik, aplikasi akan memvalidasi formulir. Jika formulir valid, fungsi `_submit()` akan dipanggil.

Dalam proses submit, `_isLoading` diatur menjadi `true` untuk menampilkan indikator loading, kemudian memanggil `LoginBloc.login` dengan email dan password yang dimasukkan oleh pengguna. Jika respons `code` dari API adalah 200, berarti login berhasil; dalam hal ini, token dan user ID akan disimpan menggunakan `UserInfo`, dan dialog sukses akan ditampilkan. Pengguna kemudian akan diarahkan ke halaman `NutrisiPage`. Namun, jika login gagal (respons tidak 200 atau terjadi error), dialog peringatan akan ditampilkan dengan pesan bahwa login gagal.

Jika login berhasil (dalam contoh ini, jika value.code sama dengan 200), kita menyimpan token dan ID pengguna menggunakan kelas `UserInfo`. Setelah menyimpan informasi, dialog sukses ditampilkan menggunakan `SuccessDialog`, yang memberi tahu pengguna bahwa login berhasil. Jika pengguna menekan tombol "OK" di dialog sukses, mereka akan diarahkan ke halaman Nutrisi Page.



3. Create Data



Pada halaman Nutrisi_Page, pengguna dapat melihat list data daftar Nutrisi buku. Ketika pengguna mengklik tanda tambah di pojok kanan atas halaman, maka akan masuk ke halaman tambah data.

The screenshot shows a mobile application interface titled 'TAMBAH Nutrisi'. It features a dark background with a central light gray rounded rectangle containing a form. The form has three input fields labeled 'Item Nutrisi', 'Kalori Nutrisi', and 'Lemak Nutrisi', and a 'SIMPAN' button at the bottom.

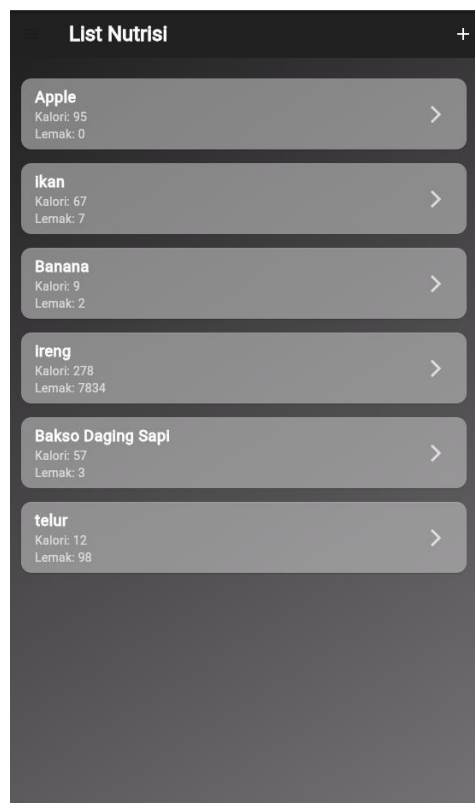
Pengguna mengisi form di NutrisiForm. Dalam NutrisiForm, terdapat tiga input text yang digunakan untuk mengambil data dari pengguna, yaitu originalLanguage untuk memasukkan nama Nutrisi asli, translatedLanguage untuk memasukkan nama Nutrisi

terjemahan, dan `translatorName` untuk memasukkan nama penerjemah. Setiap field menggunakan `TextEditingController` untuk menangani input dari pengguna.

Sebelum menyimpan data, form melakukan validasi untuk memastikan bahwa semua field terisi dengan benar. Jika ada field yang kosong, pengguna akan diberi tahu dengan pesan yang sesuai.

Ketika pengguna menekan tombol "Simpan", fungsi `simpan()` dipanggil. Di dalam fungsi ini, pertama-tama dilakukan validasi form. Jika validasi berhasil, data diambil dari `TextEditingController` dan dimasukkan ke dalam model `Nutrisi`.

Selanjutnya, fungsi `addNutrisi()` dari `NutrisiBloc` dipanggil dengan objek `Nutrisi` yang baru dibuat sebagai parameter. Dalam `addNutrisi()` pada `NutrisiBloc`, data kemudian dikirim ke server melalui API menggunakan HTTP POST request. Jika permintaan berhasil, API akan mengembalikan respons yang menunjukkan bahwa data telah berhasil ditambahkan



Setelah data berhasil disimpan, pengguna dapat dinavigasikan kembali ke halaman `NutrisiPage`, yang akan memperbarui daftar `Nutrisi` dengan data terbaru.

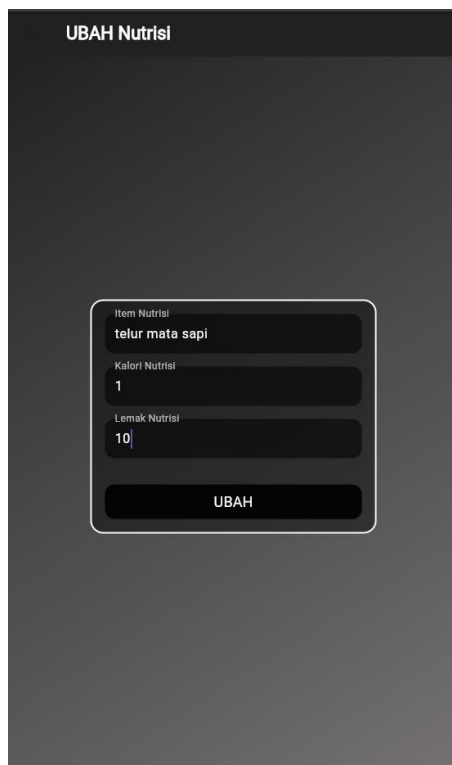
4. Edit data

Ketika pengguna memilih `Nutrisi` yang ditampilkan di halaman `NutrisiPage`, aplikasi akan mengarahkan ke halaman `Detail`.



Ketika pengguna mengklik tombol “edit”, data yang relevan diambil dan dimuat ke dalam field input di NutrisiForm menggunakan TextEditingController. Field yang diisi sebelumnya adalah originalLanguage, translatedLanguage, dan translatorName.

Pada halaman edit data, pengguna dapat mengubah data seperti original language, translated language, maupun translator languagenya.

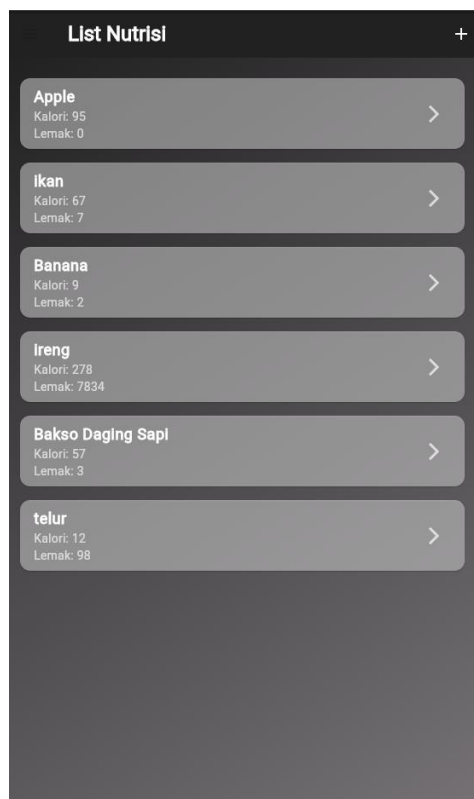


Ketika pengguna melakukan perubahan dan menekan tombol "Ubah", fungsi `simpan()` dipanggil. Di dalam fungsi ini, validasi form dilakukan untuk memastikan bahwa semua

input valid. Jika validasi berhasil, data terbaru diambil dari TextEditingController dan dimasukkan ke dalam objek Nutrisi yang ada.

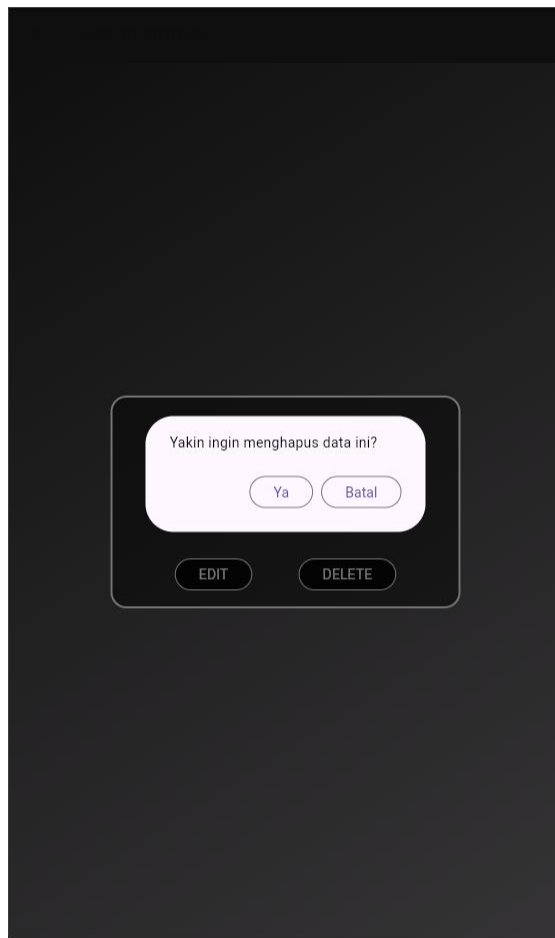
Fungsi updateNutrisi() dari NutrisiBloc dipanggil dengan objek Nutrisi yang telah diperbarui sebagai parameter. Proses pengiriman data ke API dilakukan di dalam fungsi updateNutrisi() di NutrisiBloc, di mana data dikirim menggunakan HTTP PUT request. Jika permintaan berhasil, API akan mengembalikan respons yang menunjukkan bahwa data telah berhasil diperbarui.

Setelah data berhasil diperbarui, pengguna akan diarahkan kembali ke halaman NutrisiPage, di mana daftar Nutrisi akan diperbarui untuk mencerminkan perubahan yang baru saja dilakukan.



5. Hapus Data

Ketika pengguna memilih Nutrisi yang ditampilkan di halaman NutrisiPage, aplikasi akan mengarahkan ke halaman Detail.



Setelah pengguna mengklik tombol “hapus”, aplikasi akan meminta konfirmasi untuk memastikan bahwa pengguna benar-benar ingin menghapus data tersebut. Konfirmasi ini bertujuan untuk menghindari penghapusan data secara tidak sengaja.

Setelah konfirmasi diberikan, fungsi `deleteNutrisi()` pada `NutrisiBloc` dipanggil. Fungsi ini bertanggung jawab untuk menghapus data `Nutrisi` yang telah dipilih oleh pengguna. Di dalam fungsi `deleteNutrisi()`, permintaan `HTTP DELETE` dikirim ke server dengan menggunakan `API`. Permintaan ini berisi `ID Nutrisi` yang ingin dihapus. Server kemudian memproses permintaan dan menghapus data `Nutrisi` dari basis data. Jika permintaan penghapusan berhasil, `API` akan mengembalikan respons yang menyatakan bahwa data `Nutrisi` telah berhasil dihapus.

Setelah data `Nutrisi` berhasil dihapus, pengguna akan diarahkan kembali ke halaman `NutrisiPage`, di mana daftar `Nutrisi` akan diperbarui secara otomatis untuk menghapus `Nutrisi` yang baru saja dihapus. Tidak ada notifikasi yang ditampilkan setelah proses penghapusan berhasil.