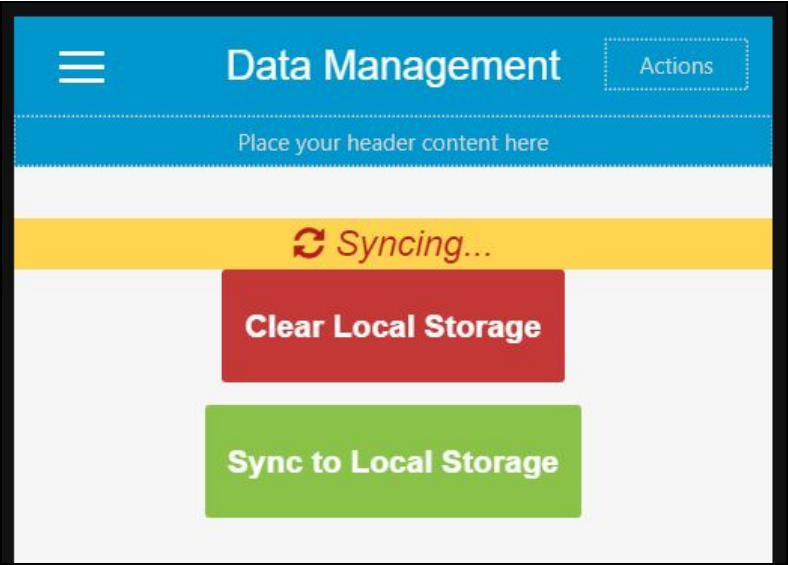


Latihan Sinkronisasi Data



Daftar Isi

| | |
|--|----|
| Daftar Isi | 2 |
| pengantar | 3 |
| Menyinkronkan Kategori, Prioritas, dan Jenis Sumber Daya Membuat | 4 |
| Layar Manajemen Data | 8 |
| Menguji aplikasi: Buat ToDo lokal pertama, Aktifkan | 15 |
| Perubahan Offline ToDos | 16 |
| Menyinkronkan ToDos dan Resource | 25 |
| Menguji aplikasi: Sinkronisasi dan interaksi offline Sinkronisasi Otomatis | 38 |
| | 39 |
| Apakah Umpan Balik Sinkronisasi | 40 |
| Menguji aplikasi: sinkronisasi otomatis dan umpan balik Akhir Lab | 42 |
| | 42 |

pengantar

Lab latihan ini akan menangani dua hal yang sangat penting dalam aplikasi seluler: skenario offline dan sinkronisasi data. Untuk itu, ada beberapa tugas penting yang akan dilakukan dalam latihan ini.

Pertama, kita akan menggunakan akselerator Service Studio untuk menghasilkan logika sinkronisasi hanya-baca untuk Kategori, Prioritas, dan Jenis Sumber Daya. Ini akan membuat logika sisi server dan sisi klien untuk menyinkronkan ketiga elemen ini. Logika ini kemudian akan digunakan dalam Tindakan OfflineDataSync, untuk menjadi bagian dari proses sinkronisasi lengkap.

Kemudian, kami akan menambahkan Layar baru ke aplikasi, DataManagement, untuk memastikan bahwa sinkronisasi dapat dipicu secara manual oleh pengguna akhir. Kami akan menambahkan masing-masing entri Menu dan Bar Bawah untuk Layar ini.

Setelah itu, kami akan mengubah logika untuk menambah / memperbarui Todos agar berfungsi pada skenario offline, menambahkan data ke penyimpanan lokal meskipun aplikasi sedang offline. Ini juga akan memerlukan beberapa perubahan dalam model data untuk menyimpan informasi tentang perubahan yang dibuat saat offline, yang nantinya akan berguna untuk sinkronisasi, saat aplikasi online.

Dengan semua logika itu siap, kami kemudian akan membuat logika Baca / Tulis secara manual untuk menyinkronkan Todos dan Sumber Daya ke server. Ini akan memungkinkan Todos dan Sumber Daya yang dibuat di penyimpanan lokal juga dikirim ke server.

Terakhir, kami akan menentukan pemicu otomatis untuk sinkronisasi, saat aplikasi online, atau saat pengguna masuk atau melanjutkan aplikasi. Kami juga akan memberikan pesan kepada pengguna akhir sehingga jelas saat sinkronisasi terjadi.

Singkatnya, di lab latihan khusus ini, kami akan:

- Tentukan logika untuk menyinkronkan Todos, Resource, ResourceTypes, Kategori, dan Prioritas
- Aktifkan pembuatan dan pembaruan Todos saat offline
- Buat Layar baru untuk memicu sinkronisasi secara manual. Tentukan pemicu otomatis
- untuk sinkronisasi
- Menampilkan pesan kepada pengguna bahwa proses sinkronisasi sedang terjadi

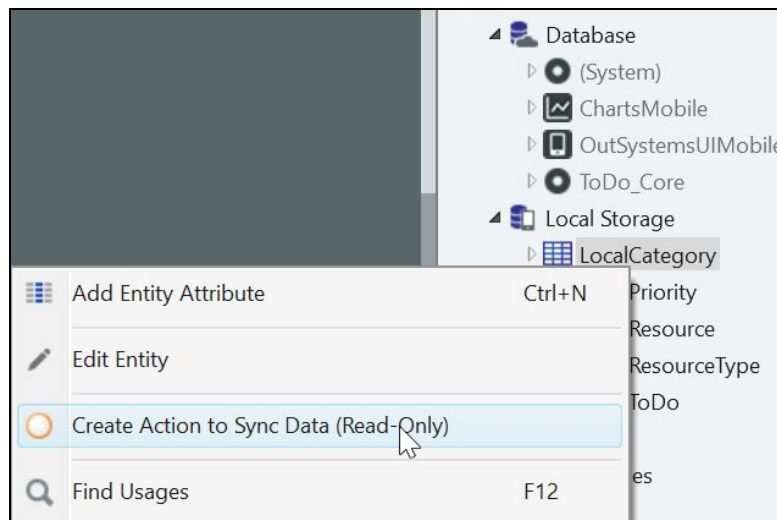
Menyinkronkan Kategori, Prioritas, dan Jenis Sumber Daya

Pada titik ini, aplikasi memiliki Entitas Database dan Entitas Penyimpanan Lokal, dengan Layar sebagai sumber data penyimpanan lokal. Sejak aplikasi dimulai secara eksklusif dengan Entitas Database, masih tidak ada data di penyimpanan lokal.

Di bagian ini, kita akan mulai mendefinisikan logika sinkronisasi, dimulai dengan **Kategori**, **Prioritas** dan **ResourceType** Entitas. Untuk tujuan itu, kita akan menggunakan beberapa akselerator OutSystems yang secara otomatis membuat logika untuk sinkronisasi data, dari Entitas Database ke Penyimpanan Lokal, menggunakan pola Read-Only dan Read / Write.

Logika yang dibuat secara otomatis ini kemudian akan digunakan di **OfflineDataSync** Tindakan Klien. Tindakan ini berjalan ketika proses sinkronisasi dipicu, dan di situlah logika sinkronisasi sisi klien ditentukan.

- 1) Buat file **Read-Only** logika untuk menyinkronkan **Kategori** data dari Database ke masing-masing Entitas Penyimpanan Lokal, menggunakan akselerator Service Studio. Tambahkan logika yang dibuat ke **OfflineDataSync** Tindakan untuk menjadikannya bagian dari sinkronisasi.
 - a) Beralih ke Tab Data, klik kanan **LocalCategory** lalu pilih *Buat Tindakan untuk Sinkronkan Data (Hanya Baca)*.



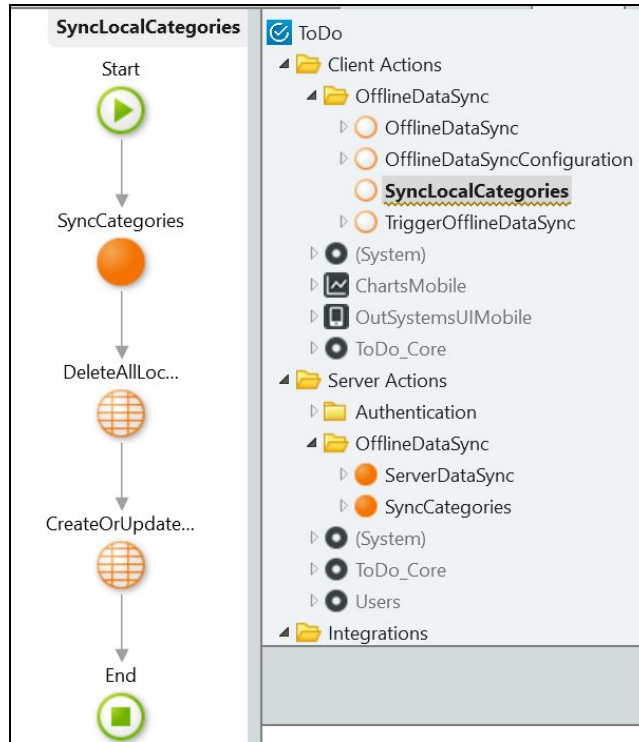
Ini membuat Tindakan Klien di bawah folder OfflineDataSync di tab Logika,

SyncLocalCategories. Tindakan mengambil data dari Server,

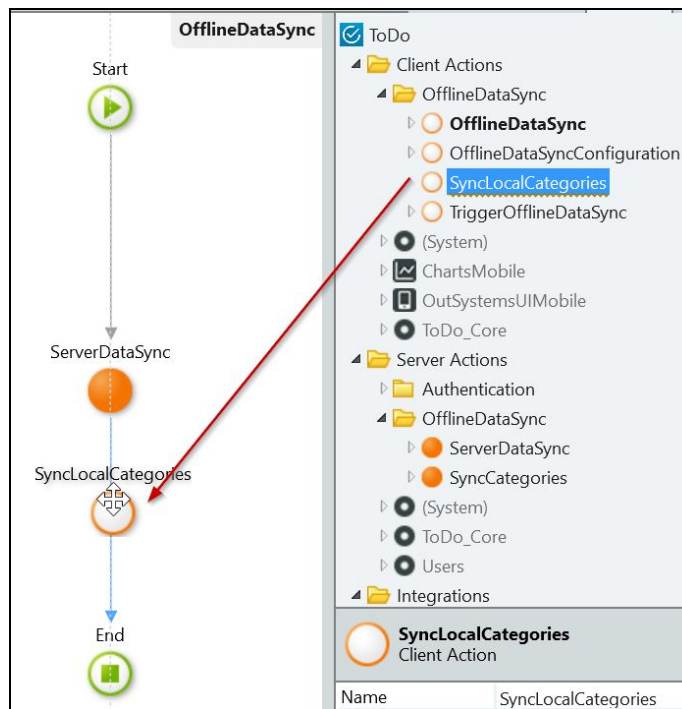
SyncCategories (juga dibuat oleh akselerator), lalu mengganti semua data di Penyimpanan Lokal untuk apa yang diambil dari Server. Pertama,

DeleteAllLocalCategories membersihkan semua data dari **LocalCategory**, dan kemudian

CreateOrUpdateAllLocalCategories menambahkan semua Kategori dari Basis Data, ke Entitas Penyimpanan Lokal masing-masing.



- b) Di tab Logic, buka **OfflineDataSync** Tindakan, dan seret yang baru **SyncLocalCategories** ke aliran, di bawah Tindakan **ServerDataSync**.



- 2) Ulangi langkah sebelumnya untuk **LocalPriority** dan **LocalResourceType**. Kedua Entitas akan disinkronkan menggunakan strategi Hanya-Baca.

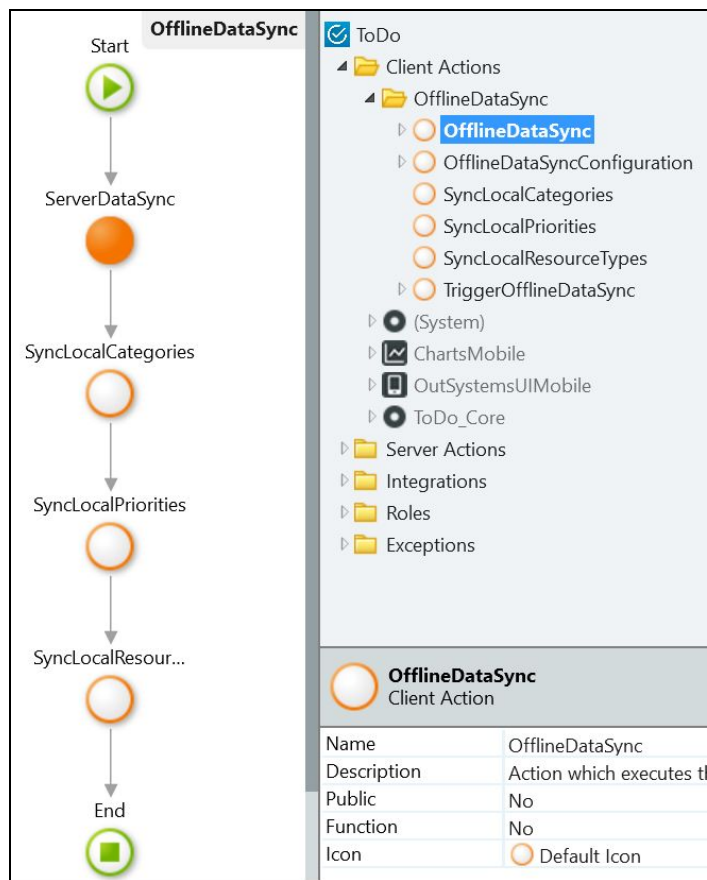
- a) Di tab Data, klik kanan pada **LocalPriority** Entitas dan pilih *Buat Tindakan untuk*

Sinkronkan Data (Hanya Baca).

- b) Ulangi proses untuk **LocalResourceType** Kesatuan.

- c) Ini menciptakan dua Tindakan Klien: **SyncLocalPriorities** Sebuah **SyncLocalResourceTypes**. Kedua Tindakan memiliki logika yang mirip dengan **SyncLocalCategories** yang dibuat di atas.

- d) Buka **OfflineDataSync** Tindakan Klien lagi, lalu seret kedua Tindakan Klien (**SyncLocalPriorities** dan **SyncLocalResourceTypes**) ke aliran OfflineDataSync, di bawah SyncLocalCategories.

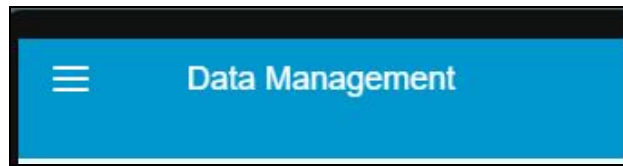


- e) Publikasikan modul untuk menyimpan perubahan di server.

CATATAN: Akselerator ini membuat logika mandiri untuk setiap pasangan Database dan Entitas Penyimpanan Lokal, misalnya Kategori. Ini berarti melakukan ini untuk ketiga Entitas, akan membuat tiga bagian logika yang terpisah. Ketika semua ini digunakan bersama, misalnya, dalam Tindakan OfflineDataSync, kita sebenarnya memiliki tiga panggilan ke server (satu untuk setiap Entitas Database), karena logika dibuat untuk itu menggunakan akselerator. Memiliki Tindakan Klien yang memanggil server beberapa kali sebenarnya harus dilakukan dengan hati-hati dan hanya jika benar-benar diperlukan, karena akan ada banyak komunikasi yang dilakukan ke server.

Dalam skenario ini, praktik terbaik adalah mencoba menggabungkan semua logika server dalam satu Tindakan Server dan memanggilnya. Dengan cara ini, kami hanya memiliki satu koneksi ke server, bukan beberapa.

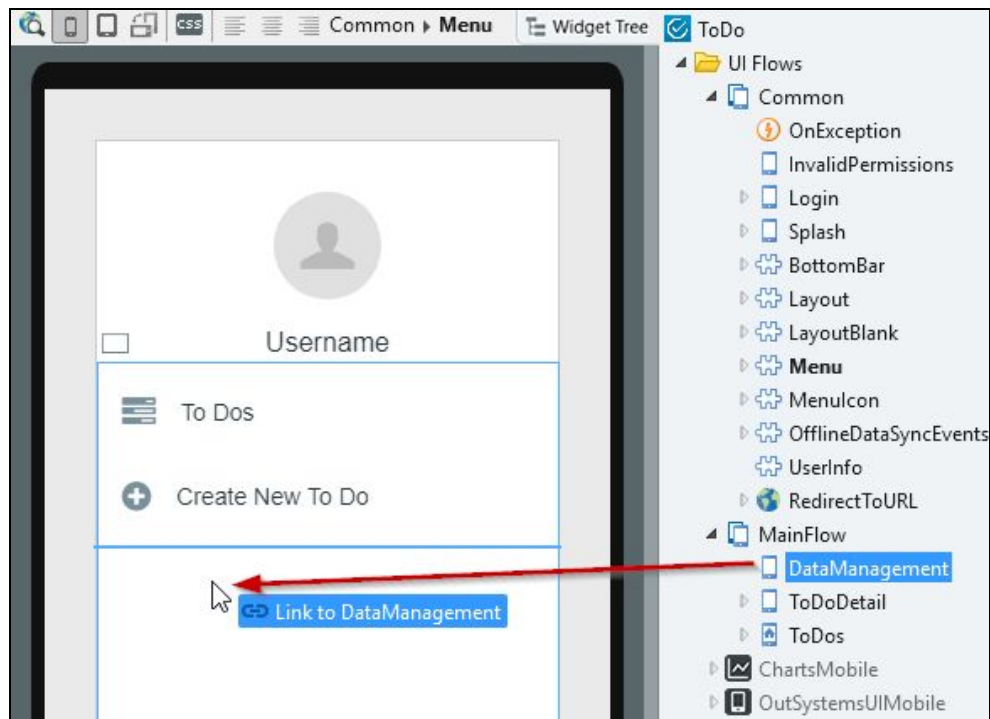
c) Tetapkan Judul Layar sebagai *Manajemen Data*.



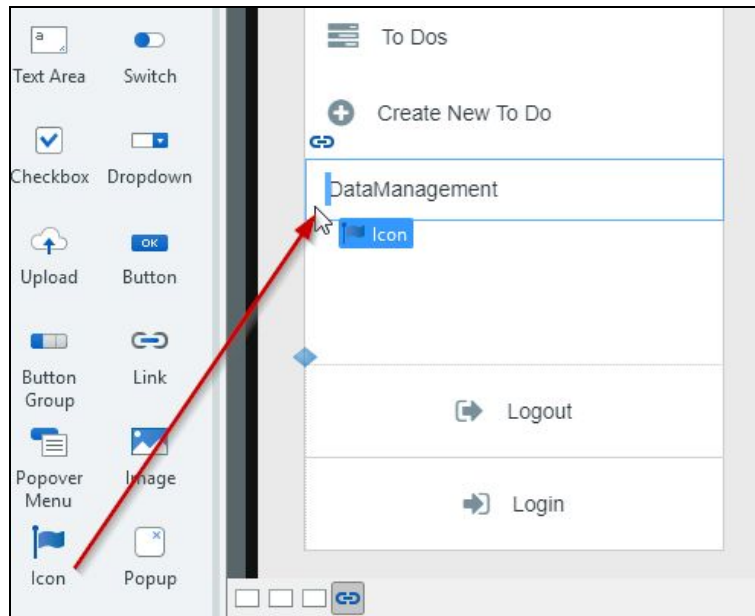
d) Perluas aliran Umum dan buka Blok Menu



e) Seret **Manajemen Data** Layar di bawah **Buat Yang Baru Untuk Dilakukan** Opsi menu



f) Seret **Ik**on tepat sebelum teks DataManagement dan pilih *roda gigi* ikon.

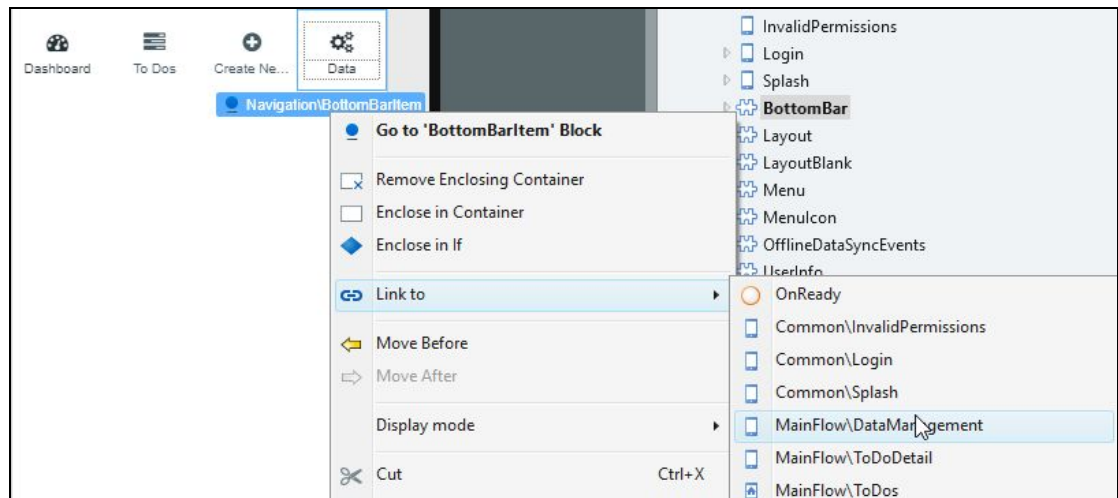


g) Tambahkan ruang kosong antara Data dan Manajemen yang akan dimiliki *Manajemen Data*.

h) Buka **BottomBar** Block, juga tersedia di bawah Common UI Flow.

i) Pilih **Data B**

</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> orang </s>



2) Buat a *Hapus Penyimpanan Lokal*/tombol di **Manajemen Data** Layar, di tengah

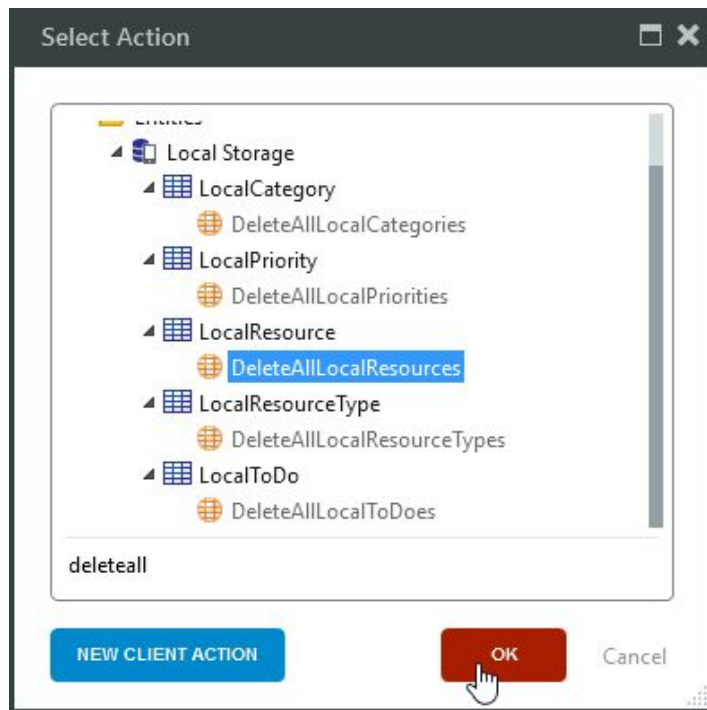
Layar dan dengan warna merah, untuk memungkinkan pembersihan Penyimpanan Lokal jika diinginkan. Tautkan ke Tindakan yang menghapus setiap catatan dari semua Entitas di penyimpanan lokal.

a) Buka **Manajemen Data** Layar.

b) Tarik a **Tombol** Widget, letakkan di dalam konten Layar. Sejajarkan dengan tengah.

- c) Ubah teks Tombol menjadi *Hapus Penyimpanan Lokal*.
- d) Ubah Tombol **Kelas Gaya** properti untuk *"Btn btn-bahaya"*.

 Ini akan membuat Button berwarna merah.
- e) Klik dua kali Tombol untuk membuat **ClearLocalStorageOnClick** Tindakan Klien.
- f) Tarik a **Jalankan Tindakan Klien** pernyataan dan letakkan di antara Awal dan Akhir, di alur Tindakan yang baru dibuat.
- g) Di **Pilih Tindakan** dialog pilih **DeleteAllLocalResources**.

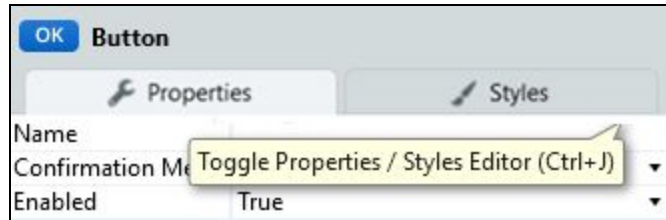


- h) Ulangi proses yang sama untuk menambahkan **DeleteAllLocalToDoes**, **DeleteAllLocalCategories**,

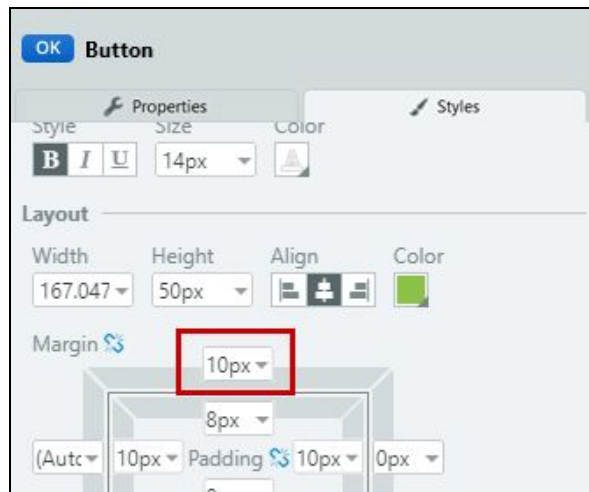
 DeleteAllLocalPriorities dan **DeleteAllLocalResourceTypes**.
- saya) Tarik a **Pesan** dan letakkan di antara DeleteAllLocalResourceTypes dan End.
- j) Setel **Pesan** properti untuk *"Semua data dihapus!"*, dan **Ketik** kepada *Sukses*. Action akan terlihat seperti screenshot berikut

Layar dan dengan warna hijau, untuk memicu sinkronisasi. Tautkan ke Tindakan yang memulai proses sinkronisasi di latar belakang.

- OutSystems Inc. orang 5901 Peachtree Dunwoody Road, NE Building C, Suite 495, Atlanta, GA 30328



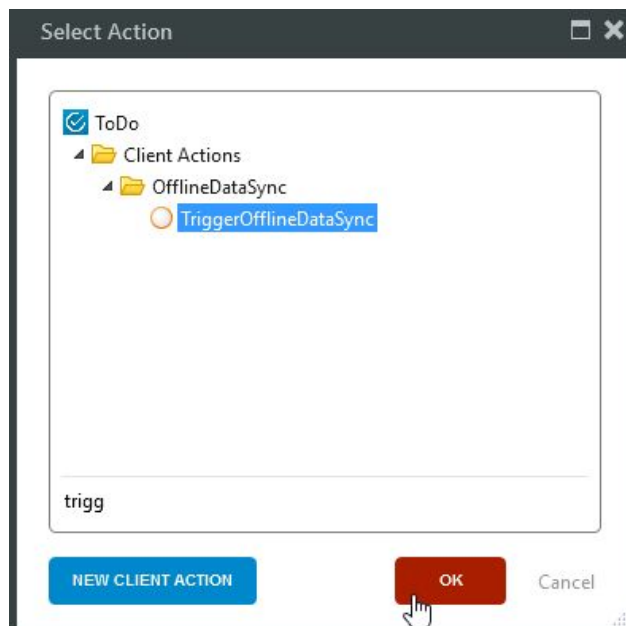
f) Atur **Margin** ke atas sebagai *10px*, untuk memisahkan Tombol ini dari yang di atas.



g) Klik dua kali Tombol untuk membuat file **SynctoLocalStorageOnClick** Tindakan Klien.

h) Tarik a **Jalankan Tindakan Klien** dan letakkan di antara Awal dan Akhir.

i) Pada dialog Select Action pilih **TriggerOfflineDataSync**.



Ini akan memicu sinkronisasi data untuk berjalan di latar belakang. Tindakan ini akan memanggil `OfflineDataSync` **secara asinkron**, tanpa memblokir aplikasi atau membuat pengguna menunggu untuk terus berinteraksi dengan aplikasi. Di sisi lain, jika kami memang memanggil Tindakan `OfflineDataSync` secara langsung, itu akan terjadi.

j) Publikasikan modul untuk menyimpan perubahan terbaru ke server.

Menguji aplikasi: Buat ToDo lokal pertama

Di bagian ini, kita akan melakukan tes pertama aplikasi, dengan membuat ToDo di penyimpanan lokal. Dengan melakukan itu, kita perlu memastikan bahwa sinkronisasi berfungsi, agar Kategori tersedia di Penyimpanan Lokal.

- 1) Buka aplikasi di browser, atau di perangkat. Perhatikan bahwa sebuah pesan muncul yang menunjukkan bahwa aplikasi telah diperbarui ke versi terbaru.
- 2) Arahkan ke Layar Manajemen Data, menggunakan Bar Bawah atau Menu, untuk memastikan perubahan bekerja dengan benar.
- 3) Klik pada Tombol untuk Menyinkronkan ke penyimpanan lokal.
- 4) Arahkan ke Layar untuk membuat ToDos baru. Pastikan Kategori terdaftar di dropdown.
- 5) Arahkan kembali ke Layar Manajemen Data dan klik pada Tombol Hapus Penyimpanan Lokal.
- 6) Kembali ke Layar untuk membuat ToDo baru dan pastikan Dropdown untuk Kategori kosong. Jika ya, itu berarti Tindakan untuk menghapus penyimpanan lokal berfungsi dengan baik.
- 7) Arahkan untuk terakhir kalinya ke Layar Manajemen Data dan klik lagi pada Tombol Sinkronkan ke Penyimpanan Lokal.
- 8) Buat ToDo baru dan pastikan hanya itu yang muncul di daftar ToDos. Ini terjadi karena ToDos mengambil data dari penyimpanan lokal, dan sejauh ini, ini adalah satu-satunya yang dibuat di penyimpanan lokal juga.

Aktifkan Perubahan Offline Todos

Pada titik ini, saat kami membuat atau memperbaiki ToDo yang ada, ToDo ditambahkan ke penyimpanan lokal dan ke server, yang mengharuskan pengguna selalu online untuk melakukan operasi itu.

Di bagian ini, kami akan menambahkan beberapa logika untuk menjamin bahwa Todos dapat dibuat dan diperbarui saat offline, tanpa menimbulkan kesalahan. Dengan cara ini, saat aplikasi online, ToDo terus ditambahkan / diperbarui ke database dan penyimpanan lokal. Saat aplikasi offline, ToDo secara eksklusif ditambahkan / diperbarui ke penyimpanan lokal.

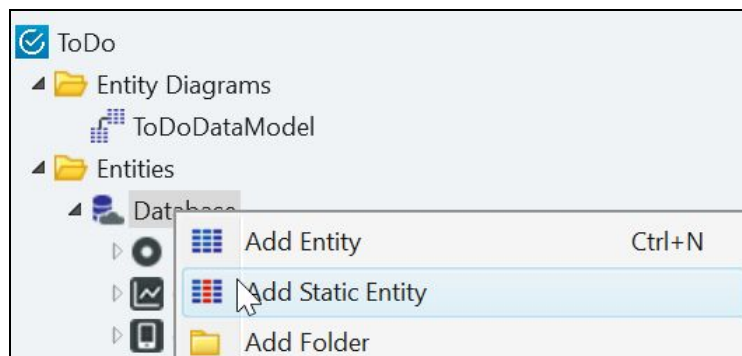
Namun, skenario offline sedikit lebih kompleks dari itu. Saat aplikasi kembali online, kami ingin memastikan bahwa database dan penyimpanan lokal disinkronkan dan ada konsistensi data di antara keduanya. Oleh karena itu, kami perlu melacak perubahan yang dibuat saat offline, untuk memastikan bahwa hanya data yang diperlukan yang disinkronkan saat sinkronisasi dipicu. Sebagai contoh, pertimbangkan bahwa dua Todos telah ditambahkan dengan aplikasi offline. Jika pengguna memiliki sepuluh Todos dalam aplikasinya, sinkronisasi seharusnya hanya mengirim dua Todos ini ke server, bukan seluruh daftar.

Untuk membantu kami dengan itu, kami perlu memodifikasi model data, untuk menyimpan informasi tentang perubahan yang dilakukan saat offline. Informasi itu bisa tentang Todos **ditambahkan** atau **diperbarui**.

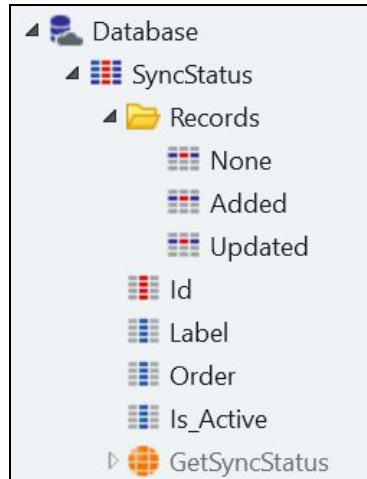
- 1) Tambahkan Status Sinkronisasi ke **LocalTodos**, untuk melacak Todos yang ditambahkan dan diperbarui, sedangkan perangkat sedang offline. Untuk itu, kita perlu membuat Entitas Statis baru **SyncStatus**, dengan

Rekaman *Tidak ada*, </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> orang </s> *Ditambahkan* dan *Diperbarui*.

- a) Beralih ke **Data** tab dan tambahkan Entitas Statis Database baru bernama *SyncStatus*.

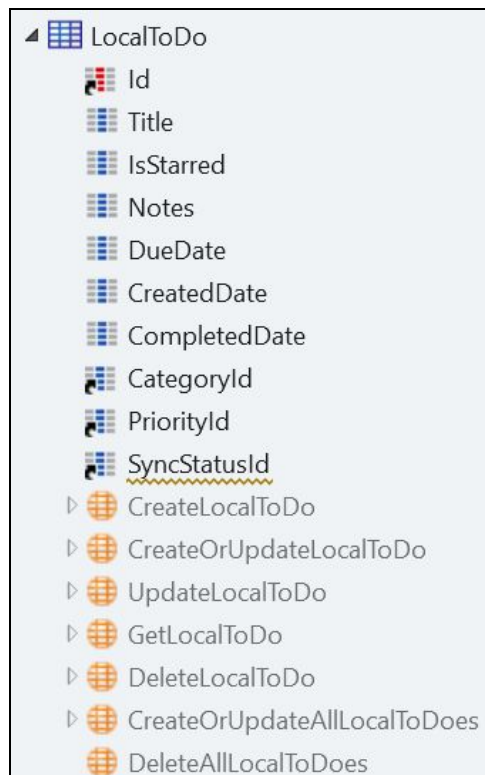


- b) Perluas file **SyncStatus** Entity, lalu klik kanan file **Rekaman** folder dan pilih *Menambahkan* *Rekam*, lalu atur nama record baru menjadi *Tidak ada*.
- c) Tambahkan dua record lagi bernama *Ditambahkan* dan *Diperbarui*.



CATATAN: Itu **Tidak ada** record akan melacak ToDos lokal yang tidak memiliki modifikasi dan sinkron dengan server. Itu **Ditambahkan** merekam trek ToDos baru yang dibuat di penyimpanan lokal tetapi belum disinkronkan ke server. Itu **Diperbarui** akan melacak ToDos yang ada di server tetapi telah dimodifikasi secara lokal.

- d) Tambahkan atribut baru ke **LocalToDo** Entitas, dan setel namanya menjadi *SyncStatusId* kemudian verifikasi bahwa file **Tipe Data** disetel secara otomatis ke *Pengenal SyncStatus*.



2) Ubah **SaveOnClick** Tindakan Klien, dari **ToDoDetail** Layar, untuk hanya membuat / memperbarui

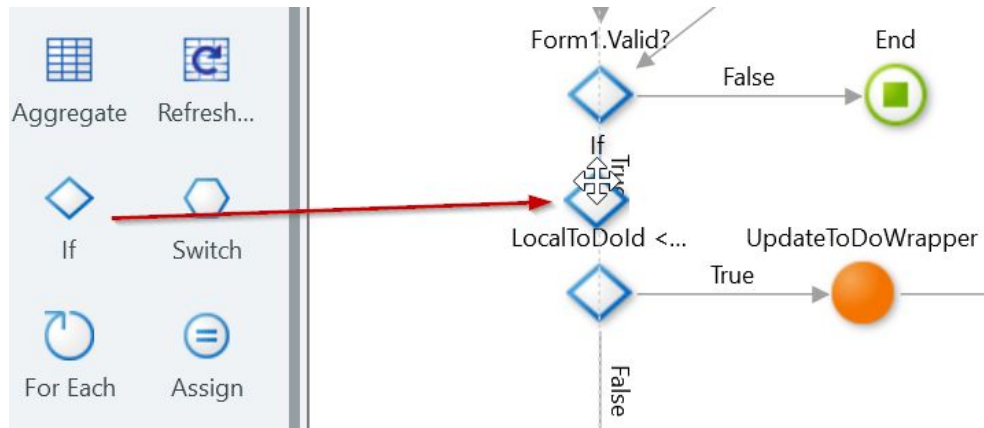
Todos dalam database jika pengguna sedang online. Jika tidak, perbarui **SyncStatus** dari LocalToDo ke *Ditambahkan* atau *Diperbarui*, tergantung pada apakah ToDo sedang dibuat atau diperbarui.

a) Beralih ke **Antarmuka** tab dan buka **SaveOnClick** Tindakan Klien dari

ToDoDetail Layar.

b) Tarik baru **Jika** pernyataan dan letakkan di atas **Benar** konektor cabang,

antara Form1.Valid? </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> orang </s> Jika



c) Atur **Label** milik If to yang baru *Offline?*, dan **Kondisi** properti untuk

bukan GetNetworkStatus ()

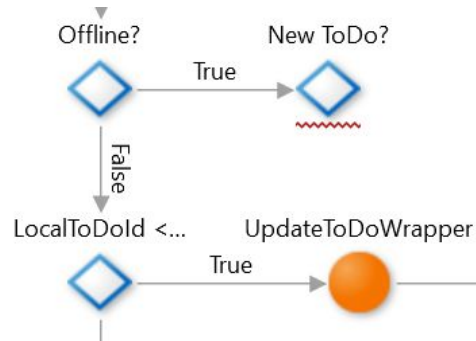
Kondisi ini menentukan apakah aplikasi sedang offline, sejak

GetNetworkStatus () Aksi kembali *Benar* saya </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>

d) Tarik yang lain **Jika** pernyataan dan letakkan di sebelah kanan **Offline?** Jika dibuat </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> langkah sebelumnya. Kemudian, buat **Benar** konektor cabang dari If yang ada ke yang baru.

e) Atur **Label** properti dari If to yang terakhir dibuat *ToDo baru?*, dan **Kondisi** properti untuk

LocalToDoId = NullIdentifier ()



k) Hubungkan **UpdateToDoWrapper** dan **Tetapan** dibawah

```
graph TD
    Start(( )) --> Offline{Offline?}
    Offline -- True --> NewToDo{New ToDo?}
    NewToDo -- True --> StatusAdded((Status Added))
    StatusAdded --> CreateOrUpdateLoc[CreateOrUpdateLoc...]
    CreateOrUpdateLoc --> MainFlow[MainFlow\ToDos]
    Offline -- False --> LocalToDoDold{LocalToDoDold <...}
    LocalToDoDold -- True --> UpdateToDoWrapper((UpdateToDoWrapper))
    UpdateToDoWrapper --> CreateOrUpdateLoc
    LocalToDoDold -- False --> CreateToDoWrapper((CreateToDoWrapper))
    CreateToDoWrapper --> GetLocalToDoById[GetLocalToDoById....]
    GetLocalToDoById --> CreateOrUpdateLoc
```

The flowchart illustrates the logic for handling 'Offline?' status. It starts with a decision diamond 'Offline?'. If 'True', it proceeds to 'New ToDo?', which if 'True', leads to 'Status Added', then 'CreateOrUpdateLoc...', and finally 'MainFlow\ToDos'. If 'Offline?' is 'False', it goes to 'LocalToDoDold <...'. From there, if 'True', it goes to 'UpdateToDoWrapper' and then to 'CreateOrUpdateLoc...'. If 'False', it goes to 'CreateToDoWrapper', then 'GetLocalToDoById....', and finally to 'CreateOrUpdateLoc...'. The 'CreateOrUpdateLoc...' node is represented by a grid icon, and 'MainFlow\ToDos' is represented by a smartphone icon.

o) Tarik baru **Tetapkan** pernyataan dan letakkan di bawah **Menambahkan SyncStatus?** Jika. Kemudian, buat **Salah** konektor cabang dari If ke pernyataan Assign.

p) Pilih **Tetapkan** pernyataan. Ubah itu **Label** kepada *Status Diperbarui* dan tentukan tugas berikut

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.Updated
```

Jika ToDo tidak ditandai sebagai Ditambahkan, itu berarti ToDo sudah ada sebelum aplikasi offline. Dengan demikian, kami memperbarui ToDo yang ada di database dan mengubah Status Sinkronisasi menjadi **Diperbarui**.

q) Buat konektor yang hilang dari **Tetapkan** pernyataan kepada **CreateOrUpdateLocalToDo** pernyataan.

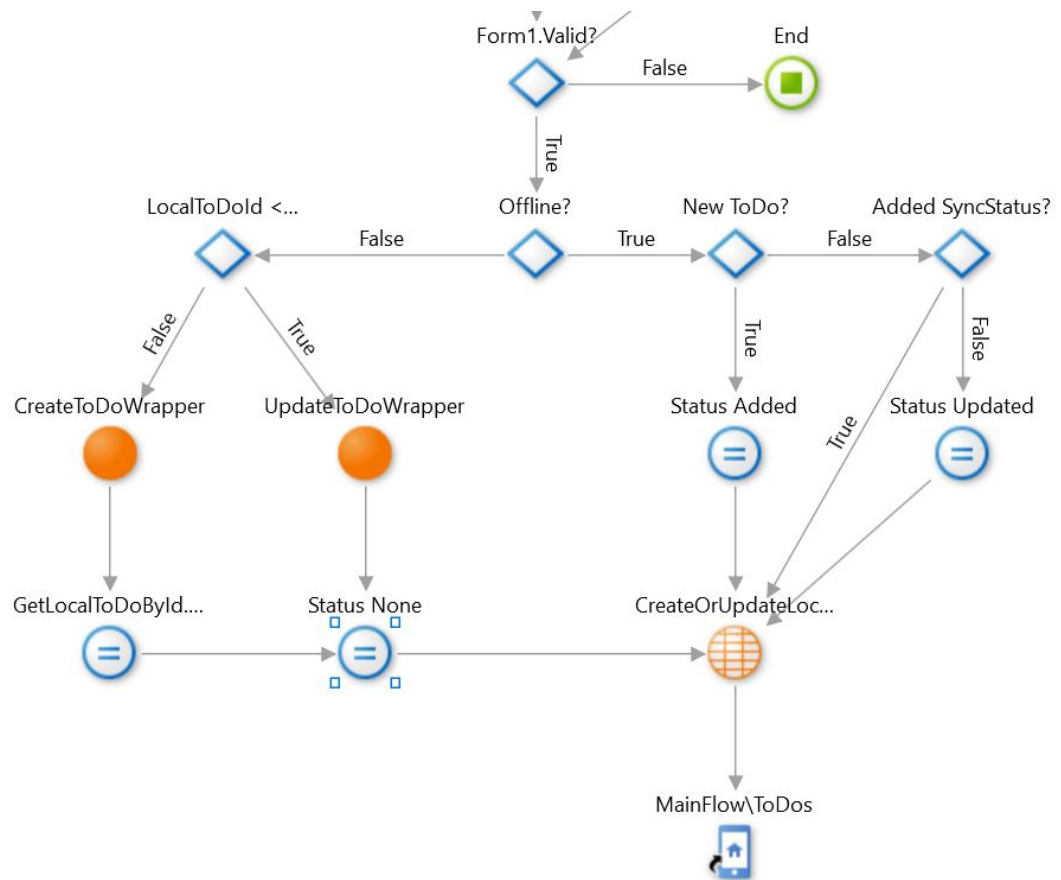
r) Seret Tugas baru dan jatuhkan di antara **UpdateToDoWrapper** dan **CreateOrUpdateLocalToDo**. Hubungkan Assign tepat di bawah CreateToDoWrapper ke yang baru ini.

s) Pilih Tugas baru. Atur itu **Label** kepada *Status Tidak Ada* dan tambahkan tugas berikut

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.None
```

Jika ToDo ditambahkan / diperbarui dalam database, karena aplikasi online, itu disimpan di penyimpanan lokal dengan Status Sinkronisasi diatur ke **Tidak ada**. Ingatlah bahwa Status Sinkronisasi hanya berguna untuk sinkronisasi data di masa mendatang dengan server, saat aplikasi kembali online.

t) Bagian terakhir dari **SaveOnClick** Tindakan Klien akan terlihat seperti berikut ini
tangkapan layar



3) Ubah **StarOnClick** Tindakan Klien dari **ToDoDetail** Layar untuk memperbarui Sinkronisasi

Status To Do. Ikuti strategi yang sama yang diterapkan pada langkah sebelumnya.

- Buka **StarOnClick** Tindakan Klien.
 - Ubah **Label** milik yang ada **Jika** kepada *ToDo ada?*
 - Seret file **Jika** Widget dan letakkan di atas **T**
- Jika ada dan **UpdateToDoWrapper** pernyataan.

d) Atur **Label** milik baru **Jika** kepada *Offline?*, dan **Kondisi** properti untuk

bukan GetNetworkStatus ()

- e) Tarik yang baru **Jika** pernyataan dan letakkan di bawah **Offline?** Jika, buat file **Benar** konektor cabang antara kedua lfs.

- f) Atur **Label** milik If to yang baru *Menambahkan SyncStatus?* dan **Kondisi** properti untuk

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.Added
```

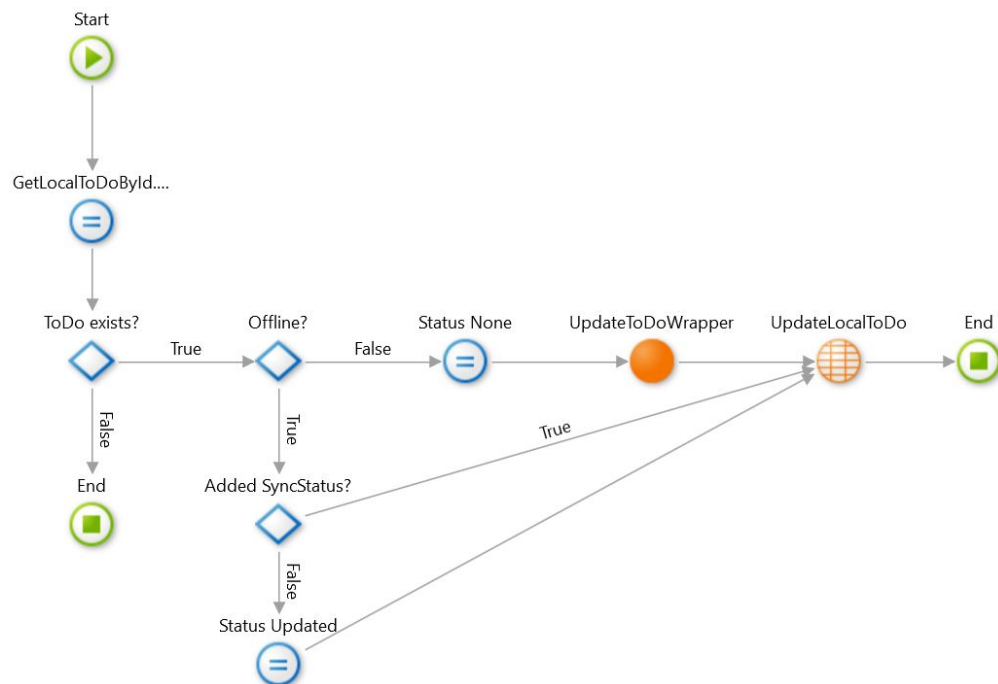
- g) Buat file **Benar** konektor cabang dari **Menambahkan SyncStatus?** Jika ke **UpdateLocalToDo** pernyataan.
- h) Seret **Tetapkan** pernyataan dan letakkan di bawah **Menambahkan SyncStatus?** Jika kemudian buat **Salah** konektor cabang antara keduanya.
- i) Pilih **Tetapkan** pernyataan yang dibuat pada langkah sebelumnya. Atur itu **Label** sebagai *Status Diperbarui* dan tentukan tugas berikut

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.Updated
```

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.Updated
```

- j) Buat konektor yang hilang dari **Tetapkan** ke **UpdateLocalToDo** pernyataan.
- k) Tarik yang baru **Tetapkan** pernyataan dan jatuhkan di antara **Offline?** Jika dan **UpdateToDoWrapper**. Atur itu **Label** ke *S **tatus Tidak ada*** dan tentukan tugas berikut

```
GetLocalToDoById.List.Current.LocalToDo.SyncStatusId =
Entities.SyncStatus.None
```



Menyinkronkan Todos dan Resource

Sekarang setelah kita menyiapkan aplikasi untuk menambahkan Todos saat offline, sekarang saatnya untuk kembali ke proses sinkronisasi dan membuat logika untuk menyinkronkan Todos dan Sumber dayanya, jika ada. Pada titik ini, tidak ada Todo tunggal yang memiliki Resource, tetapi logikanya sudah dapat dibuat dengan mudah. Dengan cara ini, di lab mendatang saat Resource akan ditambahkan ke Todos, logika sinkronisasi akan siap.

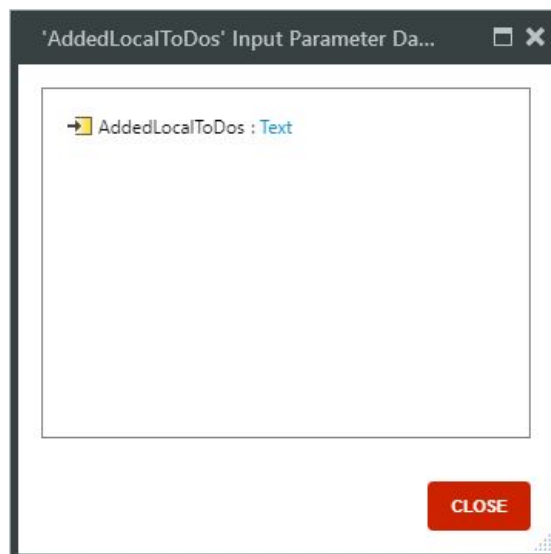
Logika untuk sinkronisasi akan bekerja berdasarkan Status Sinkronisasi. Todos yang ditandai sebagai **Ditambahkan** akan dibuat dalam database dan Todos ditandai sebagai **Diperbarui** akan diperbarui dalam database.

Untuk itu, kami akan mengubah Tindakan ServerDataSync dan OfflineDataSync, dimulai dari yang pertama.

1) Buat Parameter Input yang diperlukan untuk **ServerDataSync**:

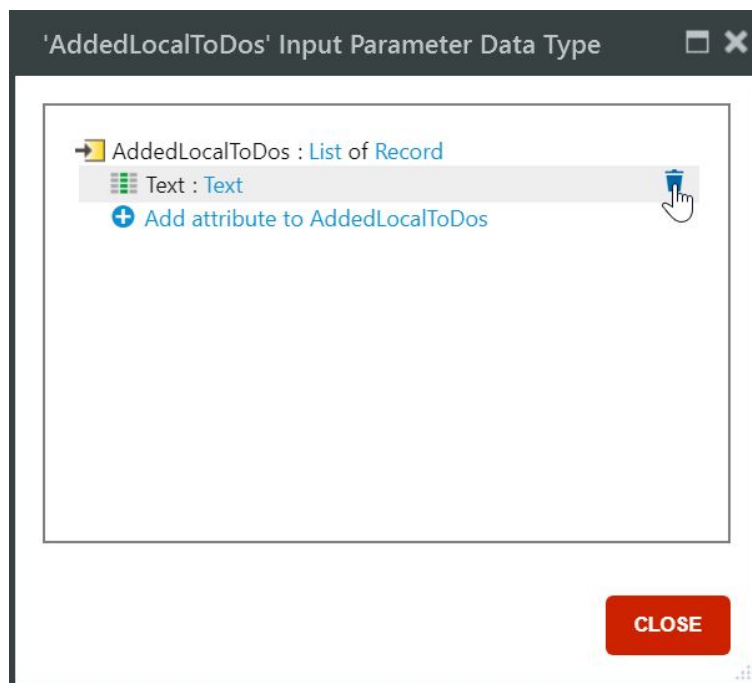
Daftar Todos yang ditambahkan di penyimpanan lokal, dan *UpdatedLocalTodos*, dengan Daftar Todos yang diperbarui di penyimpanan lokal.

- Di **Logika** tab, cari dan buka **ServerDataSync** Tindakan Server di bawah **OfflineDataSync** folder dari Tindakan Server.
- Tambahkan Parameter Input baru bernama *AddedLocalTodos*.
- Klik dua kali file **Type Data** properti untuk mengedit tipe data Parameter Input.
- Klik **SMS** tipe data untuk mengubahnya.



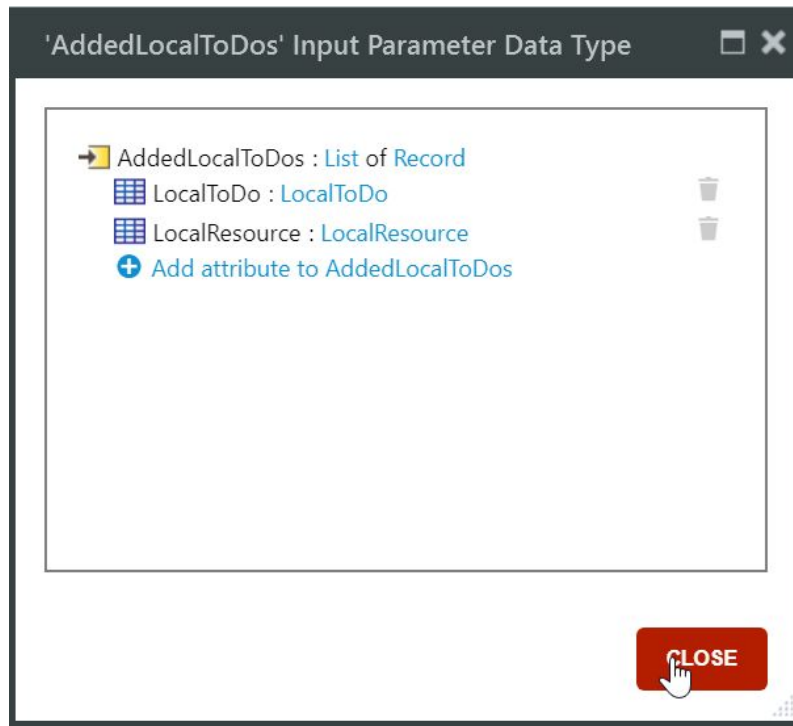
- Pilih *Daftar* di dialog berikutnya dan kemudian klik **Ok**.

g) Klik **sampah** ikon untuk menghapus T

[illegible]

LocalToDo dan verifikasi bahwa jenisnya telah berubah menjadi **LocalToDo**.

- i) Klik **Tambahkan atribut ke AddedLocalToDos** dan setel nama atribut baru ke *LocalResource*, lalu verifikasi bahwa tipe datanya disetel ke **LocalResource**.
- j) Klik **Tutup** untuk menutup editor tipe data.




- k) Tambahkan Input lain ke **ServerDataSync** Tindakan Server bernama *UpdatedLocalToDos*, lalu tentukan **D**

2) Buat logika di **ServerDataSync**, untuk Todos yang ditambahkan di penyimpanan lokal, saat aplikasi sedang offline. Untuk itu, kami akan membuat ToDo di database untuk setiap LocalToDo di file **AddedLocalToDos** Daftar.

- a) Buka **ServerDataSync** Tindakan Server.
- b) Tarik a **Untuk Setiap** pernyataan dan letakkan di bawah **LogMessage** pernyataan.
- c) Tarik a **Jalankan Tindakan Server** pernyataan dan letakkan di sebelah kanan Untuk Setiap. Di itu **Pilih Tindakan** dialog pilih **CreateToDoWrapper** Tindakan.
- d) Buat **Siklus** konektor dari For Each
- e) Pilih **Untuk Setiap** dan setel **Daftar Rekaman** properti ke *AddedLocalToDos* Parameter Masukan.
- f) Pilih **CreateToDoWrapper** Sebuah atribut LocalToDo dari Daftar AddedLocalToDos. Misalnya, setel **Judul** kepada *AddedLocalToDos.Current.LocalToDo.Title*

Lakukan logika yang sama untuk Parameter Input Aksi yang tersisa.

|  CreateToDoWrapper Run Server Action | |
|---|---|
| Name | CreateToDoWrapper |
| Action | PublicDBActions\CreateToDoWrapper |
| Title | AddedLocalToDos.Current.LocalToDo.Title |
| IsStarred | AddedLocalToDos.Current.LocalToDo.IsStarred |
| Notes | AddedLocalToDos.Current.LocalToDo.Notes |
| DueDate | AddedLocalToDos.Current.LocalToDo.DueDate |
| CreatedDate | AddedLocalToDos.Current.LocalToDo.CreatedDate |
| CompletedDate | AddedLocalToDos.Current.LocalToDo.CompletedDate |
| CategoryId | AddedLocalToDos.Current.LocalToDo.CategoryId |
| PriorityId | AddedLocalToDos.Current.LocalToDo.PriorityId |

CATATAN: Untuk Setiap memungkinkan iterasi melalui semua LocalToDos di file

AddedLocalToDos Daftar. Kemudian, di dalam loop kami menggunakan

CreateToDoWrapper untuk menambahkan masing-masing ke database.

- g) Seret **Jika** pernyataan dan letakkan di bawah **CreateToDoWrapper** pernyataan, lalu buat penghubung antara keduanya.

- h) Atur **Label** milik dari **Jika** kepada *Memiliki Sumber Daya?*, dan **Kondisi** untuk

AddedLocalToDos.Current.LocalResource.Id <> NullIdentifier (

- sayalah) Tarik yang lain **Jalankan Tindakan Server** pernyataan dan letakkan di sebelah kiri **Memiliki Sumber Daya?** Jika.

- j) Di **Pilih Tindakan** dialog pilih **CreateOrUpdateResourceWrapper**

Action dan kemudian buat file **Benar** konektor cabang dari If to the Action. Jika CreateOrUpdateResourceWrapper tidak ada, karena tidak dibuat dalam latihan 6.2x, pilih **CreateOrUpdateResource** Tindakan.

- k) Atur Parameter Input dari **CreateOrUpdateResourceWrapper** sesuai, dengan informasi yang disimpan di **AddedLocalToDos** Daftar. Sebagai contoh, atur

Nama file untuk

AddedLocalToDos.Current.LocalResource.Filename

Dalam kasus **ToDold**, setel ke *CreateToDoWrapper.ToDold*, keluaran Tindakan CreateToDoWrapper, karena Resource memiliki Id ToDo yang sama.

AddedLocalToDos.Current.LocalResource

-
- ```
stateDiagram-v2
 [*] --> AddedLocalToDo
 AddedLocalToDo --> End : End
 AddedLocalToDo --> CreateToDoWrapper : Cycle
 CreateToDoWrapper --> HasResource{
 HasResource?
 }
 HasResource --> AddedLocalToDo : False
 HasResource --> CreateOrUpdate : True
 CreateOrUpdate --> AddedLocalToDo
```

29

[illegible]

nilai-nilai di **UpdatedLocalToDos** Daftar.



lalu buat penghubung antara keduanya.

*UpdatedLocalToDos.Current.LocalResource.Id* <> *NullIdentifier* ()

**Memiliki Sumber Daya? Jika pernyataan.**

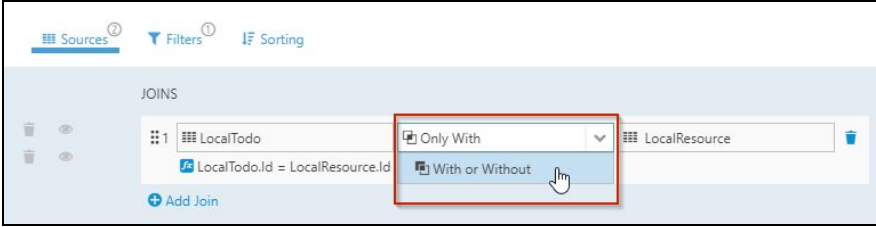
Tindakan.

## CreateOrUpdateResourceWrapper2.

demikian. Setel **ToDold** kepada *UpdatedLocalToDos.Current.LocalToDo.Id*







Ini akan membuat Agregat mengambil semua Todos, terlepas dari apakah mereka memiliki atau tidak Resource yang menyertainya.

g) Beralih ke **Filter** tab dan tambahkan filter berikut

```
LocalToDo.SyncStatusId = Entities.SyncStatus.Added
```

Filter ini akan menjamin bahwa hanya LocalToDos dengan Status Sinkronisasi yang ditetapkan sebagai **Ditambahkan**, akan diambil dari penyimpanan lokal. Hasil dari Agregat ini adalah **AddedLocalToDos** Daftar.

#### h) Kembali ke Tindakan Klien OfflineDataSync.

i) Tarik yang lain **Agregat** dan letakkan di antara GetAddedLocalToDos dan Pernyataan ServerDataSync.

j) Ubah namanya menjadi *GetUpdatedLocalToDos* dan mengaturnya **Sumber** menjadi **LocalToDo** dan **LocalResource**, dengan a *Dengan atau Tanpa* bergabung klausa.

k) Di tab Filter, tambahkan filter berikut

```
LocalToDo.SyncStatusId = Entities.SyncStatus.Updated
```

Agregat ini memiliki ide yang sama dengan yang di atas, tetapi kali ini untuk LocalTodos dengan Status Sinkronisasi ditetapkan sebagai **Diperbarui**.

l) Kembali ke **OfflineDataSync** Tindakan Klien.

m) Pilih **ServerDataSync** pernyataan, lalu di area properti setel

**AddedLocalToDos** Parameter ke *GetAddedLocalToDos.List* dan

**UpdatedLocalToDos** Parameter ke *GetUpdatedLocalToDos.List*.

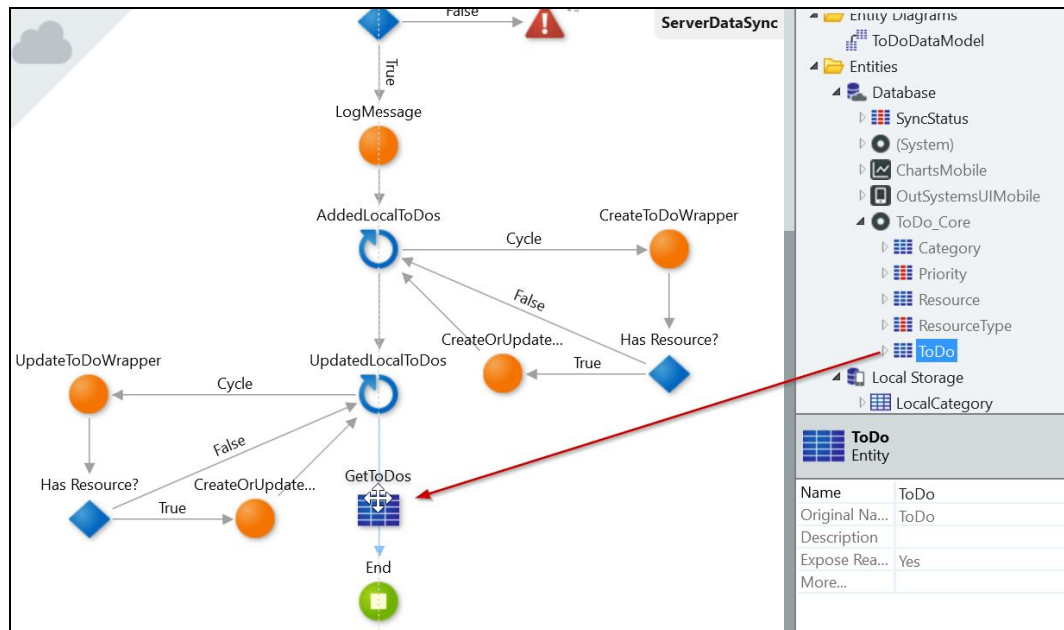
5) Sekarang setelah kami menyiapkan logika untuk mengirim ToDos yang ditambahkan dan diperbarui ke server, satu-satunya hal yang tersisa terkait logika sinkronisasi adalah memperbarui penyimpanan lokal dengan data yang dikembalikan dari server. Setelah ToDos dan Sumber Daya ditambahkan / diperbarui ke database, kita juga harus memperbarui penyimpanan lokal dengan data baru dari server, sehingga kedua sumber data disinkronkan dan cocok. Agar ini terjadi, file

**ServerDataSync** harus mengembalikan semua Todos dan Resource.

a) Di bawah tab Logic, klik kanan file **ServerDataSync** dan pilih *Tambahkan Output Parameter*.



- </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>



- f) Buka GetTodos Aggregate dan tambahkan filter berikut untuk hanya mendapatkan Todos dari pengguna yang login.

```
ToDo.UserId = GetUserId ()
```

- g) Kembali ke Tindakan `ServerDataSync` dan seret dan lepas file **Sumber daya** Kesatuan di bawah **`GetToDosByUserId`** Agregat.
- h) Buka **`GetResources`** Mengumpulkan dan juga memfilter Sumber Daya (dan `ToDos`) menurut pengguna saat ini masuk

```
ToDo.UserId = GetUserId ()
```

**CATATAN:** Ingatlah bahwa kondisi gabungan ditetapkan sebagai **Hanya Dengan**, yang berarti hanya akan mengembalikan Sumber daya yang terkait dengan ToDos dari pengguna yang masuk.

- i) Kembali ke Tindakan ServerDataSync, seret file **Tetapkan** dan jatuhkan di antara GetResources dan End.

```
LocalResources = GetResources.List
```

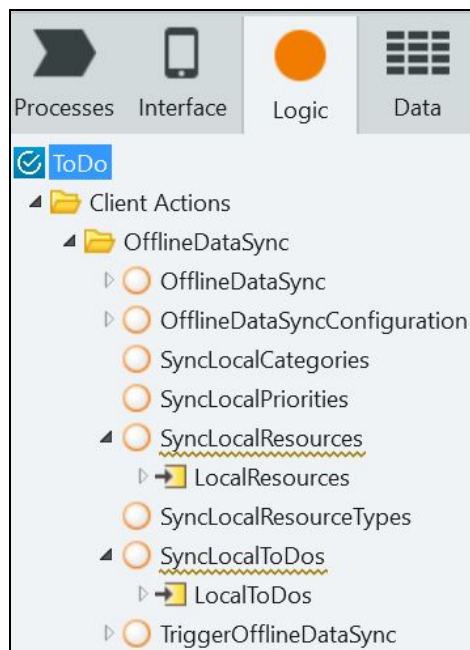
|                                |                    |   |
|--------------------------------|--------------------|---|
| LocalToDos                     |                    | ▼ |
| = GetToDosByUserId.List        |                    | ▼ |
| Mapping from ToDo to LocalToDo |                    |   |
| Id                             | ToDo.Id            | ▼ |
| Title                          | ToDo.Title         | ▼ |
| IsStarred                      | ToDo.IsStarred     | ▼ |
| Notes                          | ToDo.Notes         | ▼ |
| DueDate                        | ToDo.DueDate       | ▼ |
| CreatedDate                    | ToDo.CreatedDate   | ▼ |
| CompletedDate                  | ToDo.CompletedDate | ▼ |
| CategoryId                     | ToDo.CategoryId    | ▼ |
| PriorityId                     | ToDo.PriorityId    | ▼ |
| SyncStatusId                   |                    | ▼ |

| Assignments              |                         |   |
|--------------------------|-------------------------|---|
| LocalResources           |                         | ▼ |
| = GetResources.List      |                         | ▼ |
| Mapping to LocalResource |                         |   |
| Id                       | Resource.Id             | ▼ |
| ResourceTypeId           | Resource.ResourceTypeId | ▼ |
| Filename                 | Resource.Filename       | ▼ |
| MimeType                 | Resource.MimeType       | ▼ |
| BinaryContent            | Resource.BinaryContent  | ▼ |

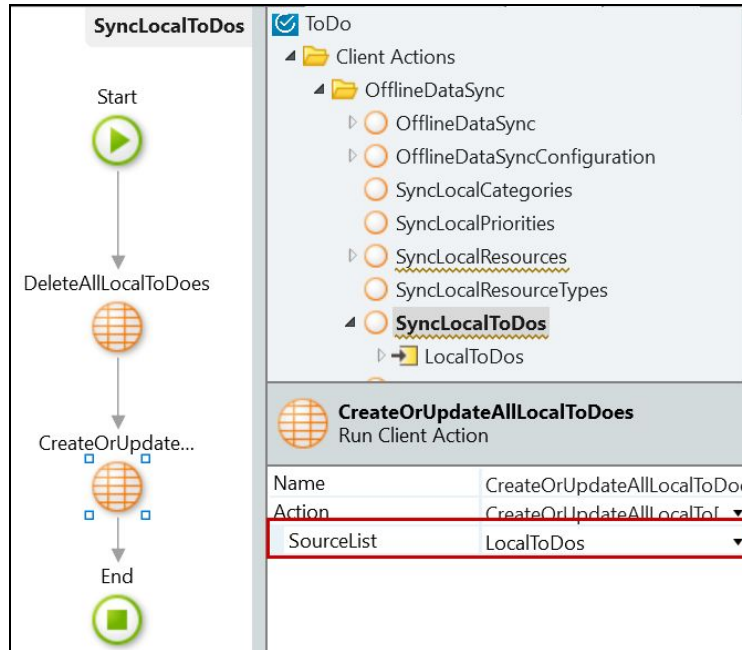
34

*SyncLocalResources*, yang akan menghapus penyimpanan lokal ToDos dan Resource masing-masing, sebelum menambahkan ToDos dan Resource yang diperoleh dari server. Data dari server akan diteruskan sebagai Parameter Input ke dua Tindakan ini. Kedua Tindakan ini kemudian akan digunakan di *OfflineDataSync* untuk menyelesaikan logika sinkronisasi.

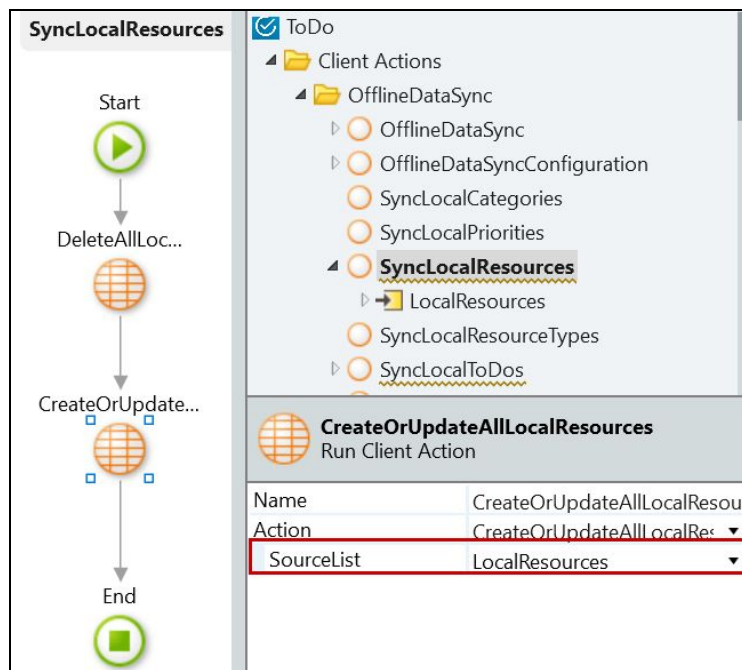
- Di bawah **OfflineDataSync** folder di Tindakan Klien, buat file *SyncLocalToDos* Tindakan Klien.
- Klik kanan Action baru dan pilih *Tambahkan Parameter Input*. Atur itu **Nama** untuk *LocalToDos* dan pastikan itu **Tipe Data** diatur ke *Daftar LocalToDo*.
- Ulangi dua langkah sebelumnya dan buat Tindakan Klien *SyncLocalResources*, dengan sebuah Input *LocalResources*, dengan **Tipe Data** *Daftar LocalResource*.



- d) Buka **SyncLocalToDo**s Tindakan dan seret dan lepas a **Jalankan Tindakan Klien** ke alirannya. Pada dialog berikutnya, pilih **DeleteAllLocalToDo**s Tindakan Entitas. Ini akan menjamin bahwa semua ToDos dihapus, sebelum membuat yang baru dengan data yang berasal dari server.
- e) Tarik yang baru **Jalankan Tindakan Klien** dan jatuhkan setelah yang sebelumnya. Pilih **CreateOrUpdateAllToDo**s Tindakan Entitas dan berikan sebagai **Sumber** itu *LocalToDo*s Parameter Masukan.



f) Ulangi langkah sebelumnya untuk **SyncLocalResources** Tindakan.

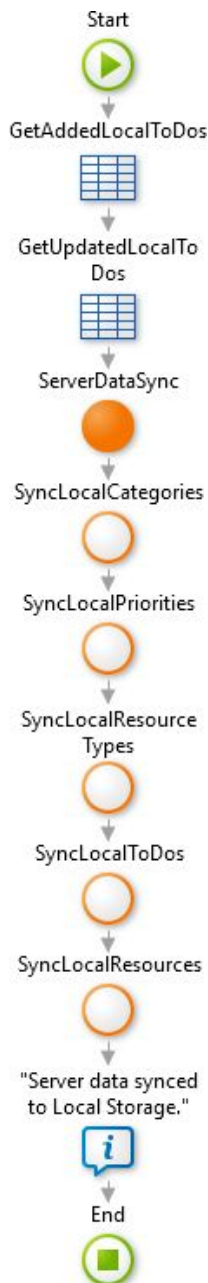


g) Buka **OfflineDataSync** Tindakan.

h) Tarik dan lepas file **SyncLocalToDo** Aksi untuk **OfflineDataSync** mengalir, benar sebelum pernyataan End. Setel **LocalToDo** Parameter masukan ke

`ServerDataSync.LocalToDo`.  
 Dengan cara ini, kita meneruskan Action baru ToDos yang dikembalikan dari server.

- i) Seret dan lepas file **SyncLocalResources** Aksi untuk **OfflineDataSync** mengalir, tepat sebelum pernyataan End. Setel **LocalResources** Parameter masukan ke *ServerDataSync.LocalResources*.
- j) Tarik a **Pesan** pernyataan dan jatuhkan tepat sebelum Akhir.
- k) Atur **Pesan** parameter ke *"Data server disinkronkan ke Penyimpanan Lokal."* dan **Tipe** kepada *Info*.
- l) Itu **OfflineDataSync** Tindakan Klien akan terlihat seperti ini

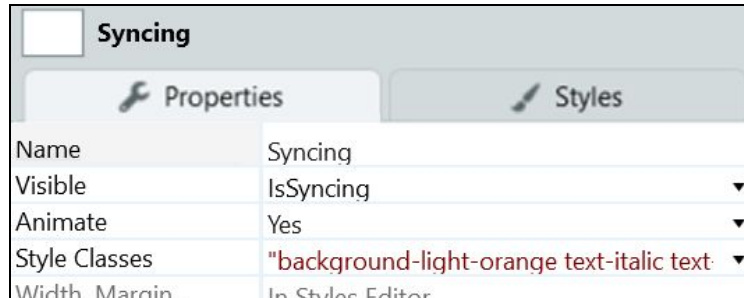












f) Seret **Ikona**, jatuhkan di dalam Container dan pilih *segarkan* ikon.

g) Atur **Ukuran** properti Ikona ke `100px` orang `100px` uk

h) Klik di sebelah kanan ikon dan ketik *Menyinkronkan...* (dengan ruang kosong sebelumnya).

2) Sekarang, kita perlu memastikan bahwa semua Layar menampilkan pesan ini saat sinkronisasi terjadi. Kami sudah

mengatur **Terlihat** properti ke **IsSyncing**

Variabel, jadi sekarang kita perlu mengatur Variabel IsSyncing ke *Benar*, saat sinkronisasi dimulai, dan ke *Salah* ketika itu tidak terjadi.

a) Buka **ActionHandler\_OnSyncStartTrigger** Tindakan Klien di bawah **Tata Letak**

Blok.

b) Tambahkan **Tetapkan** pernyataan tepat setelah Start, dan tentukan tugas berikut

*IsSyncing = Benar*

Karena Tindakan ini berjalan saat sinkronisasi dimulai, yang dipicu oleh Peristiwa, kami memastikan bahwa **IsSyncing** variabel disetel ke *Benar*.

c) Buka **ActionHandler\_OnSyncCompleteTrigger** Tindakan Klien.

d) Tambahkan **Tetapkan** pernyataan tepat setelah Start, dan tentukan tugas berikut

*IsSyncing = False*

Mengikuti logika yang sama, karena Tindakan ini berjalan saat sinkronisasi selesai, file

**IsSyncing** Variabel harus disetel ke *Salah*.

e) Buka **ActionHandler\_OnSyncErrorTrigger** Tindakan Klien dan tambahkan **Menetapkan**

pernyataan seperti yang sebelumnya.

f) Publikasikan modul untuk menyimpan perubahan di server.

## Menguji aplikasi: sinkronisasi otomatis dan umpan balik

Pada bagian latihan ini, kita akan menguji aplikasi dan pemicu sinkronisasi otomatis. Selain itu, kami perlu memastikan bahwa umpan balik yang dibuat di bagian sebelumnya muncul dengan benar di Layar.

- 1) Buka aplikasi di perangkat dan logout.
- 2) Login lagi dengan pengguna Anda. Pastikan bahwa pesan yang menunjukkan sinkronisasi data selesai muncul.
- 3) Arahkan ke **Manajemen Data** Saring dan bersihkan penyimpanan lokal. Lalu, jadikan aplikasi offline.
- 4)
- 5) Nyalakan kembali jaringan perangkat dan pastikan **Menyinkronkan...** pesan muncul lagi di perangkat.
- 6) Arahkan ke Layar ToDos dan jamin semua ToDos muncul.

## Akhir Lab

Dalam latihan ini, kami mendefinisikan interaksi offline dan logika sinkronisasi data untuk aplikasi.

Setelah latihan ini, aplikasi akan berfungsi baik secara online maupun offline, artinya ToDos dapat dibuat / diperbarui ketika aplikasi tersebut offline. Saat online, maka akan berfungsi seperti semula dan segera mengirimkan data ke server. Saat offline, itu hanya akan menyimpan ToDos di penyimpanan lokal, dan kemudian menyimpan status perubahan yang dibuat pada ToDos, yang nantinya berguna untuk sinkronisasi.

Terkait logika sinkronisasi, kami menetapkan pendekatan Hanya-baca untuk Prioritas, Kategori, dan Jenis Sumber Daya, menggunakan akselerator Service Studio. Untuk ToDos dan Resource, kami secara manual menentukan perilaku Baca / Tulis, menggunakan informasi yang disimpan saat offline.

Kami juga menentukan cara untuk memicu sinkronisasi. Opsi manual dengan menentukan Layar baru untuk membantu pengguna akhir memicunya dengan mengklik Tombol, dan opsi otomatis dengan menyetel pemicu otomatis OnLogin, OnOnline, dan OnResume pada Tindakan OfflineDataSyncConfiguration.

Akhirnya, kami mendefinisikan pesan umpan balik bagi pengguna untuk mengetahui kapan data sedang disinkronkan.