

Review Session

By Citizen Development Team

Review Points

- 
1. **Data Type**
 2. **Data Relation**
 3. **Delete Rules**
 4. **ERD**
 5. **Entities**
 6. **Dynamic & Static Entity**
 7. **Aggregates**
 8. **Join Aggregates**
 9. **Filter & Sort Aggregates**
 10. **Fetch Properties**
 11. **On After Fetch**
 12. **Widgets**
 13. **UI Flow**
 14. **Screen & Blocks**
 15. **Event & Handlers**
 16. **Variables**
 17. **Input Parameter, Local Variable, Output Parameter**
 18. **Client Variables & Site Properties**
 19. **Logic Tools**
 20. **Server Action & Client Action**
 21. **Roles**
 22. **Role Action**
 23. **Role Exception**
 24. **Bootstrap Entity using Excel File**

Review Points

- 25. **Debugging**
- 26. **Timer**
- 27. **Process**
- 28. **User Outsystems & User App**
- 29. **Login Integrated with IDGov**
- 30. **Using SQL**
- 31. **Input & Output Parameter SQL**
- 32. **Using API**
- 33. **Audit Trail**
- 34. **Object Storage**
- 35. **Cara Akses Data HCP**
- 36. **Type File yang Disimpan di HCP**

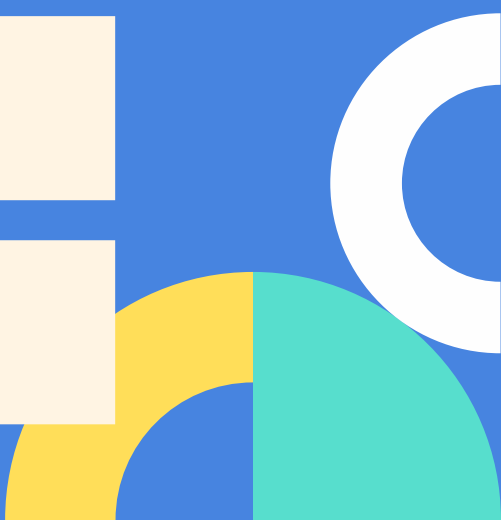




Data Type

Tipe Data yang tersedia dalam Outsystems :

Basic Data Type		
Binary Data	Time	Decimal
Boolean	Date Time	Email
Currency	Integer	Phone Number
Date	Long Integer	Text
<Entity> Identifier		





Data Type

Compound Data Type

<Entity> or <Structure>

Object

Record

Collection Data Type

List



Data Relation

Data relation dibuat dengan mendefinisikan *single reference attribute* (juga dikenal sebagai "*foreign key*" dalam terminologi database). Tipe datanya adalah <Entity> Identifier yang sesuai dengan tipe Identifier dari entitas lain yang terkait.

Beberapa tipe Data Relation :

a). One to One

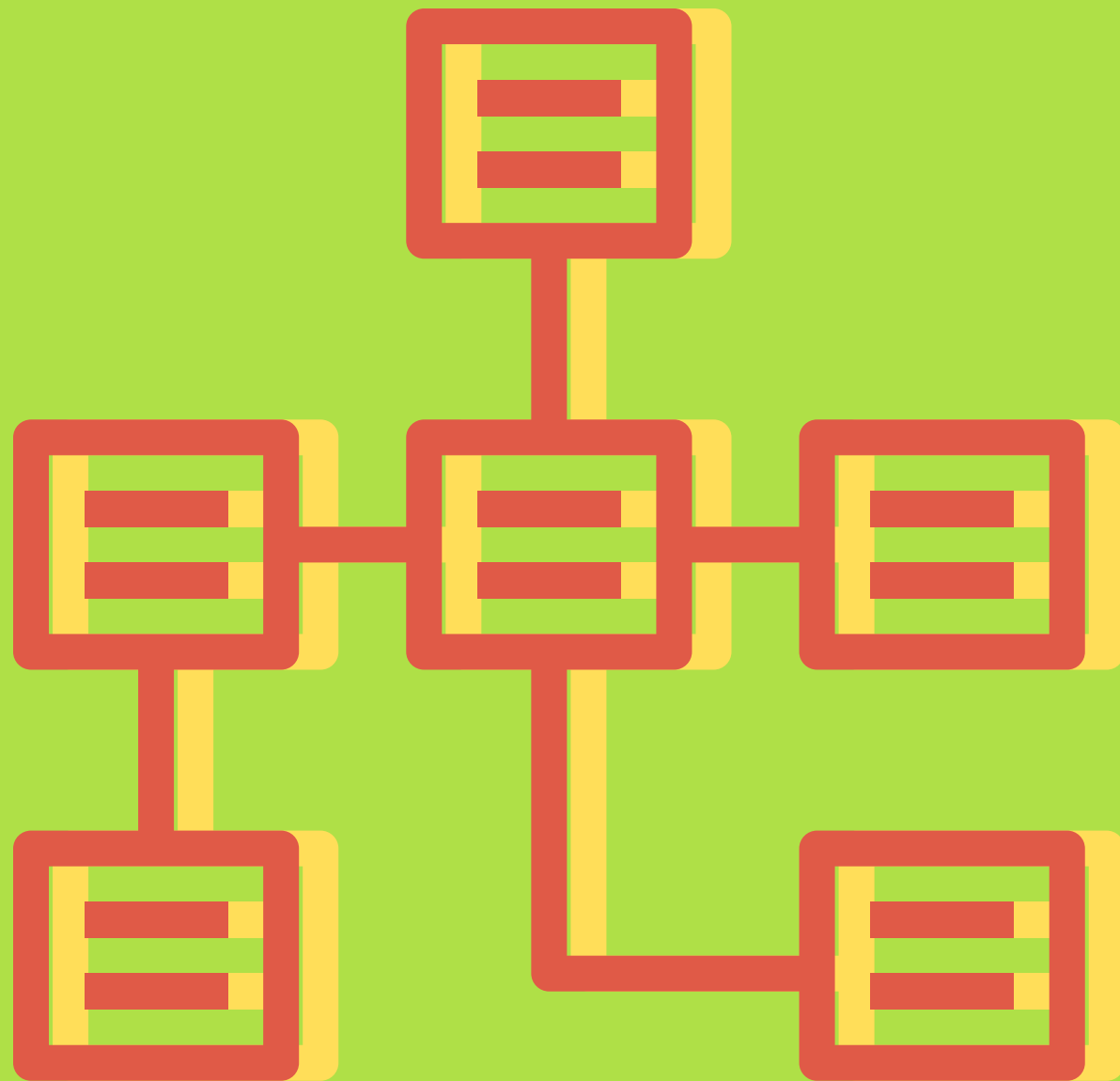
1 record di entity A berkaitan dengan 1 record di entity B, begitupula sebaliknya.

b). One to Many

1 record di entity A memiliki banyak record di entity B, namun 1 record di entity B hanya memiliki 1 record di entity A.

c). Many to Many

1 record di entity A berkaitan dengan banyak record di entity B, begitupula 1 record di entity B memiliki banyak record di entity A.





Delete Rules

Ketika membuat suatu relasi data, kita harus menentukan **referential integrity** yang akan digunakan.

Referential Integrity menentukan apa yang terjadi pada record milik Entity B yang mereferensikan record Entity A, ketika record Entity A dihapus.

Terdapat 3 macam referential integrity :

a). Protect

Mencegah penghapusan record milik entity utama selama record tersebut masih terdapat pada Entity lain.

b). Delete (Cascade Delete)

Ketika record milik entity utama dihapus, semua record entity utama yang terdapat pada entity lain juga dihapus.

c). Ignore

Memperbolehkan penghapusan record milik entity utama dan masih menyimpan record entity utama yang terhapus tersebut di entity lain.

ERD

ERD (Entity Relationship Diagram) atau diagram hubungan entitas adalah diagram yang digunakan untuk perancangan suatu database dan menunjukkan relasi antar entitas beserta atribut-atributnya secara detail.

Disinilah berbagai tipe relasi data diimplementasikan untuk memenuhi kebutuhan aplikasi yang ingin dibuat.

Entities

Merupakan elemen yang memungkinkan developer menyimpan informasi dalam database dan mengimplementasikan model database.

Sederhananya dapat dianggap sebagai suatu **tabel**.

Entitas memiliki Atribut Entitas yang menyimpan nilai informasi yang tersimpan. Contoh atribut entitas adalah: Nama, Alamat, Kode Pos, Kota dan sebagainya

Sederhananya dapat dianggap sebagai **kolom** dalam table database.



Dynamic & Static Entity



Dynamic Entities

Menyimpan data yang dapat dikelola secara dinamis (saat run-time) dengan *Create, Get, Updat, Delete action*. Informasi yang disimpan dalam database, juga dapat diambil menggunakan *Aggregate* atau *Get Entity Method*.



Static Entities

Mereka menyimpan data yang nilainya statis yang ditentukan pada *design time*. Informasi ini disimpan dalam database, tidak berubah saat run-time (tidak dapat melakukan *Create, Update, Delete*) dan dapat diambil menggunakan *Aggregate* atau *Get Entity Method*.

Aggregates

Aggregate memungkinkan developer mengambil data menggunakan query yang dioptimalkan dan disesuaikan dengan penggunaan. Aggregate secara otomatis menyerap perubahan dalam model data dan dapat memuat data milik database lokal dari server.

Aggregate dapat digunakan pada server side dan client side :

a). Client-side Aggregates

Dijalankan dalam client-logic. User dapat menggunakannya untuk mendapatkan data untuk widget saat Screen atau Block dimuat.

b). Server-side Aggregates

Dijalankan dalam server-logic. User dapat menggunakannya dalam *logic flow*.

Aggregates mendukung *entity join*, *advance filter*, ataupun *sorting*.

Joins Aggregates

Dengan melakukan *drag entity* ke agreggate, maka *data joins* akan secara otomatis terbentuk. User juga bisa menyesuaikan *entity* yang digabungkan (*joined entities*) di tab *source*.

Ada beberapa cara untuk melakukan *join record* dari 2 *entities* :

- Hanya ambil record yang memiliki kecocokan di kedua entitas (**Only With / Inner Join**)
- Ambil semua baris dari entitas pertama, meskipun tidak ada kecocokan pada entitas kedua (**With or Without / Left Join**)
- Ambil baris dari kedua entitas (**With / Full Join**)
- Menggabungkan record pada 2 entities, tanpa memedulikan hubungannya



Filter Aggregates

Developer dapat menambahkan kondisi untuk melakukan filter dan mendapatkan data yang ingin diambil dari database.

Sort Aggregates

Sorting dapat bertipe *fixed* atau *dynamic*.

Fixed Sorting : Sorting DESC / ASC ditetapkan saat design time

Dynamic Sorting : Sorting DESC / ASC dapat ditentukan saat runtime



Fetch Propertis

Terdapat 2 pilihan fetch data, yaitu :

a). At Start

Data diambil dan disiapkan pada tahap screen initialization

b). Only on Demand

Data diambil hanya pada saat screen membutuhkan data tersebut
(e.g. saat refresh table)

On After Fetch

Event Handler dari On After Fetch dijalankan tepat setelah Aggregates atau Data Action selesai mengambil data. Karena setiap Aggregates atau Data Action memiliki Event Handler On After Fetch-nya sendiri, developer dapat menerapkan logika untuk menindaklanjuti data spesifik yang diambil dari source data tersebut.

Source

:https://success.outsystems.com/documentation/11/developing_an_application/implement_application_logic/screen_and_block_lifecycle_events/#on-after-fetch

Widgets

Merupakan elemen visual yang membantu developer untuk mendesain dan mengatur user interface dari aplikasi yang dikembangkan.

Commonly-Used Widgets :

Buttons, Label, Text, Expression, Link, CheckBox, Container, Table, Pop-Up

Conditionally-Used Widgets :

Dropdown, Dropdown Tags, Accordion, RangeSlideInterval, Wizard

Jenis-jenis widgets secara lengkap dapat diakses di link berikut :

https://success.outsystems.com/documentation/11/reference/outsystems_language/traditional_web/web_interfaces/designing_screens/

(pada dropdown Designing Screens di side navigation bar)

UI Flow

UI Flow adalah elemen yang mengelompokkan Screen dan Block.

Saat developer membuat aplikasi baru, aplikasi tersebut sudah memiliki beberapa UI Flow default :

- **Common** : berisi UI dan logic yang digunakan kembali oleh aplikasi di Screen dan Block. Misalnya, menu, info signed-in user, logic sign in.
- **Layout** : berisi Block yang menentukan layout screen.
- **MainFlow** : UI Flow default dimana developer dapat mulai menambahkan Screen di aplikasi. UI Flow ini akan kosong saat aplikasi baru dibuat.

Screen & Blocks

Screen adalah halaman yang ditampilkan di *web browser* atau di *mobile apps* user.

Block adalah bagian *screen* yang *reusable* dan didalamnya dapat diimplementasikan *logic* tertentu sesuai kebutuhan aplikasi.

Sebuah screen atau block dapat terdiri dari beberapa block.



Event & Handlers

Event dan Handlers (logic action) dapat diatur pada Screen dan Block. Berikut berbagai macam Event yang dapat dipilih :

Events	
On Initialize	Action dijalankan sebelum me-render screen atau block (dan sebelum mengambil data.
On Ready	Action yang akan dijalankan setelah static content elemen ini dirender. Data mungkin tidak tersedia saat ini.
On Render	Action yang akan dijalankan saat screen/elemen ini sudah dirender sepenuhnya dan setelah perubahan variabel, aggregate, atau data action dari screen/elemen.
On Destroy	Action yang akan dijalankan saat elemen ini akan dihapus dari DOM.

***DOM** => Document Object Model, berkaitan dengan bagaimana web browser merepresentasikan halaman web secara internal





Event & Handlers For Block

Selain 4 events sebelumnya (On Initialize, On Ready, On Render, On Destroy), terdapat event tambahan untuk **block (tidak ada di screen)**, yaitu :

Events

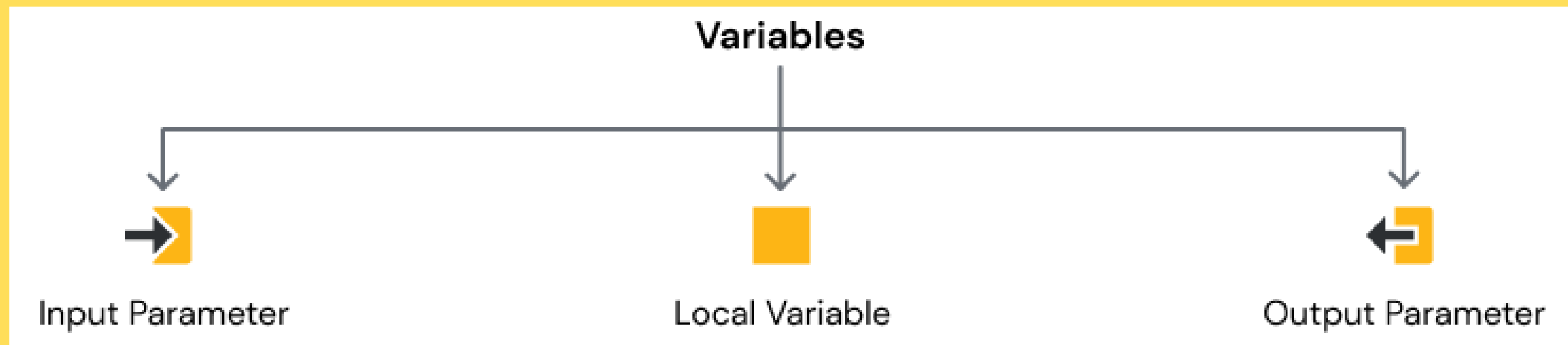
On Parameter Changed

Action yang akan dieksekusi ketika value input parameter dari elemen ini berubah.



Variables

Saat mengembangkan aplikasi, developer dapat menggunakan variabel untuk menyimpan berbagai jenis value, misalnya angka, teks, atau hasil perhitungan. Value ini dapat digunakan di mana saja dalam aplikasi atau di screen tertentu.



Input Parameter, Local Variable, Output Parameter

Input parameter memungkinkan developer untuk menyediakan data ke elemen untuk digunakan lebih lanjut. Input parameter akan tersedia untuk dipakai dalam *scope* elemen terkait.

Jika developer menambahkan input parameter ke client action, maka developer dapat :

- Berikan value untuk input parameter tersebut saat memanggil client action.
- Menggunakan value dalam logic flow client action, misalnya dalam expression atau sebagai bagian dari value input parameter lainnya.

Local variable hanya ada dalam scope elemen induknya, misalnya pada screen atau action. Local variable hanya dapat ditetapkan dan digunakan secara lokal di dalam scope tersebut. Local variable dihancurkan saat eksekusi sudah selesai di ruang lingkup elemen induk.

Output parameter memungkinkan developer untuk mengembalikan *computed values* dari action, process, atau process flow element.

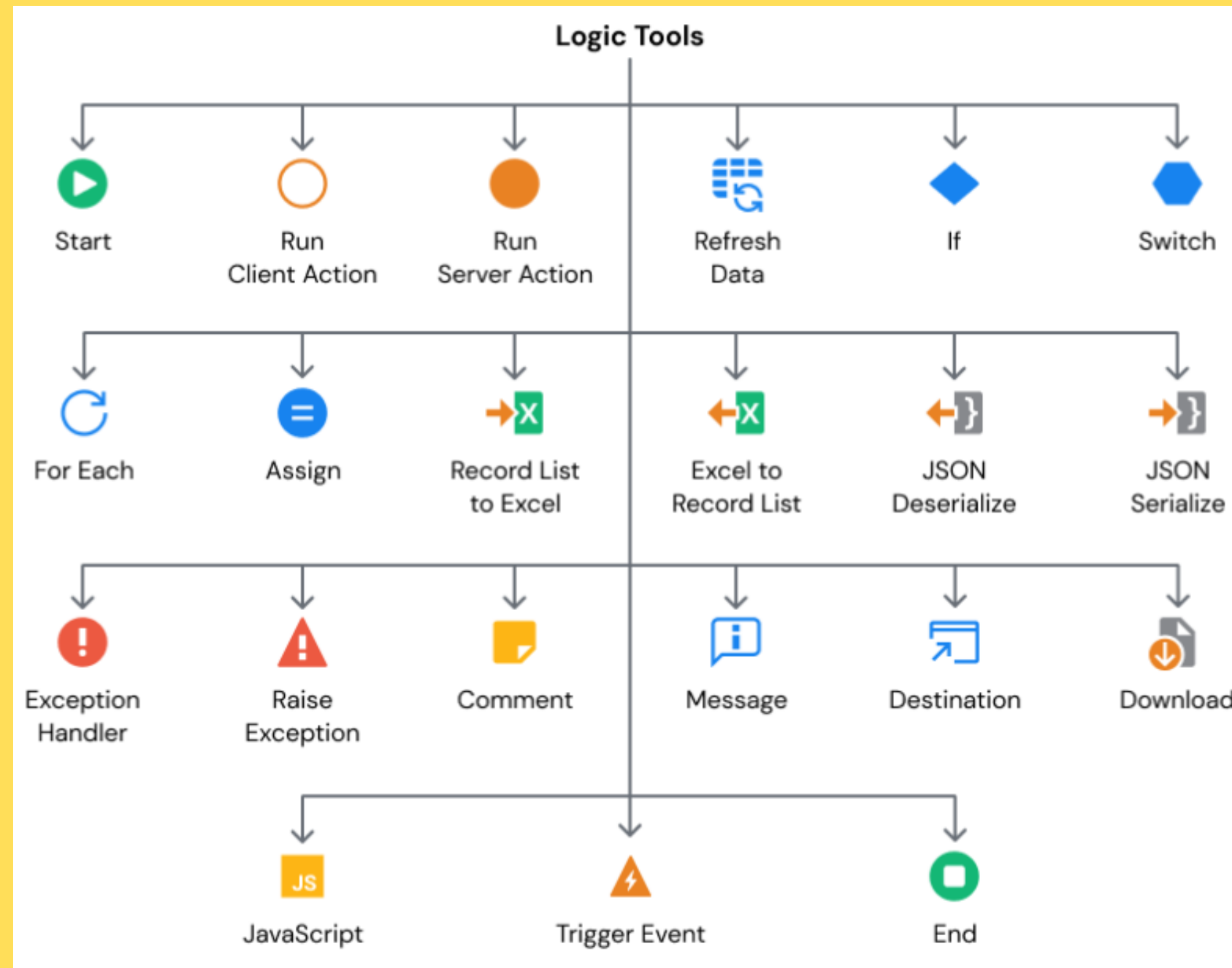
Client Variables & Site Properties

Client variable hanya dapat menyimpan *basic data type*. Satu-satunya pengecualian adalah tipe data binary, yang merupakan tipe data dasar, tetapi tidak dapat disimpan dalam variabel ini.

Client variable di-*reset* ke nilai *default* saat pengguna logout dari aplikasi atau saat platform mengeluarkan pengguna secara otomatis. Namun, jangan gunakan client variable untuk menyimpan informasi sensitif atau rahasia.

Site properties merupakan variabel global yang memiliki nilai konstan, atau nilai yang tidak sering berubah.

Logic Tools



Toolbox Service Studio berisi berbagai *logic tools* yang membantu developer membangun aplikasi yang komprehensif.

Logic tools memungkinkan developer untuk, misalnya, menjalankan server action yang mengimplementasikan *business rules* aplikasi, mengambil updated data dari data source, dan memasukkan value ke variabel.



Server Action & Client Action

Server Action adalah action yang menjalankan logika di sisi server.

Client Action adalah action yang menjalankan logika di sisi client.






Roles

Roles digunakan untuk membatasi atau mengizinkan *end-user* mengakses screen dan operasi tertentu dari aplikasi yang dikembangkan.

Ada 2 jenis Roles yang otomatis terbuat : **Anonymous** (all end-users permitted to access) dan **Registered** (only logged in users permitted to access).





Role Action

Untuk setiap Peran dalam modul Anda, OutSystems menyediakan Tindakan berikut:

a). Check<role_name>Role

Memeriksa apakah end-user tertentu telah diberi akses ke roles tertentu.

Catatan: Fungsi ini hanya berlaku untuk user aktif. Saat Anda menggunakannya untuk user yang tidak aktif, hasilnya selalu "False", bahkan jika pengguna telah diberikan roles tersebut.

b). Grant<role_name>Role

Menyediakan akses untuk end-user tertentu ke roles tertentu. Developer dapat menggunakannya untuk user aktif dan tidak aktif. Role yang diberikan kepada end-user tetap ada sampai di-revoke.

Action ini tidak tersedia untuk Role Anonymous dan Registered.

c). Revoke<role_name>Role

Menolak akses untuk end-user tertentu ke roles tertentu.

Action ini tidak tersedia untuk Role Anonymous dan Registered.

Role Exception

Untuk setiap Role dalam modul Anda, OutSystems membuat exception berikut:

Not <role_name>

Exception ini dimunculkan saat user mencoba mengakses elemen, misalnya screen, yang memerlukan role tertentu dan user belum diberikan role tersebut.

Bootstraps Entity Using Excel File

Developer outsystems dapat mengimpor data dari file Excel untuk upload data ke entiti. Fitur ini berguna saat akan melakukan development dan testing aplikasi. Dengan cara ini, developer dapat dengan cepat menggunakan data di aplikasi saat melakukan development.

Saat developer melakukan bootstrap dari excel, maka ada 3 hal yang terjadi :

1. Pengunggahan excel ke server.
2. Pembuatan Server Action yang akan memuat data excel ini ke dalam entity.
3. Membuat timer yang akan berjalan saat di-publish, dimana server action yang dibuat di langkah 2 akan dijalankan.

Debugging

Debug aplikasi Anda di Service Studio dengan menjeda eksekusi di breakpoint, yaitu titik tertentu dalam modul. Kemudian jalankan logic step by step. Ini memungkinkan developer untuk menemukan masalah apa pun pada design logic.

Tab Debugger menampilkan informasi aplikasi, seperti variabel dan runtime values. Selain itu ditunjukkan juga *current debug context* (current thread, event name, UI Flow, Screen dan Action, saat *applicable*). Gunakan *debugger commands* yang tersedia di *Debugger Toolbar* dan di *Debugger Menu*.

https://success.outsystems.com/documentation/11/developing_an_application/troubleshooting_applications/debugging_applications/



Timer

Timer adalah tools OutSystems yang memungkinkan eksekusi logika aplikasi secara berkala pada waktu yang dijadwalkan (*scheduled*). Hal ini juga dikenal sebagai ***batch job***.

Timer yang berbeda dapat dieksekusi pada saat yang sama, tetapi Timer yang sama tidak pernah memiliki lebih dari satu eksekusi dalam satu waktu.

Skenario umum dimana kita bisa menggunakan Timers :

- a). Eksekusi job yang sama setiap hari pada waktu yang sama. Misalnya, kirim email setiap hari pada pukul 4 pagi ke user tertentu dari aplikasi.

- b). Eksekusi logika aplikasi yang biasanya membutuhkan waktu lama untuk selesai. Misalnya, pada jam 2 pagi di hari pertama setiap bulan, sistem harus mengarsipkan banyak record database, dimana dibutuhkan sekitar 2 jam untuk menyelesaikannya.

Process

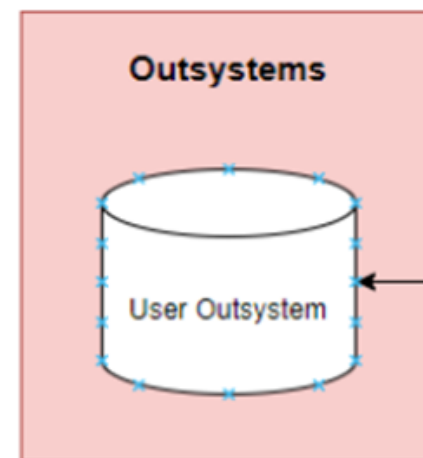
Process adalah elemen yang memungkinkan developer mengintegrasikan proses bisnis ke dalam aplikasinya. Suatu proses dirancang dalam *process flow*, yang biasanya mewakili aktivitas yang harus dilakukan selama siklus hidup entity.

Proses dapat berjalan secara paralel. Proses dapat berhenti dan melanjutkan eksekusi tergantung pada database events (**Launch On, Close On, Conditional Start**), dapat menunggu tanggal/waktu yang ditetapkan (**Wait Activity**), dan dapat menunggu interaksi manusia (**Human Activity**).

(Bedanya dengan Timer adalah Timer hanyalah satu action yang dijalankan dari awal hingga selesai pada jadwal tertentu)

User Outsystems & User Apps

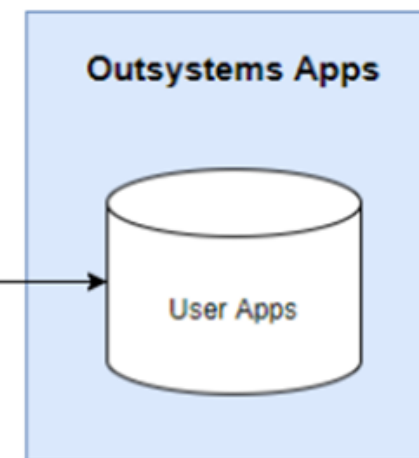
List user yang punya akses ke
aplikasi2 di outsystems
`getUserId()` => user identifier



(System) User

ID Governance

List user pada aplikasi
spesifik di outsystems

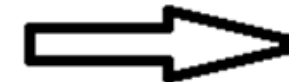


Entity user on specific app

Request
akses ke apps



Approval By
S3 & SA



IDGov Hit API
Create user app



Login Integrated with IDGOV

IDGovernance

Aplikasi yang mensentralisasi pendaftaran hak akses user ke aplikasi-aplikasi yang ada di BCA

Semua aplikasi di outsysteams harus terintegrasi pendaftarannya di IDGovernance, sehingga hak akses user ke aplikasi tidak bisa didaftarkan melalui aplikasi yang dikembangkan, tetapi **centralized** di aplikasi IDGovernance.

Using SQL

SQL memungkinkan developer melakukan *running, testing, dan review query* SQL di aplikasi yang dikembangkan. SQL dapat menjadi *tools* serbaguna untuk developer yang menguasai *SQL languages*. Walaupun begitu, untuk manipulasi data yang lebih mudah dan optimal, sebaiknya menggunakan Aggregate.

SQL mengakses data hanya melalui input parameter, dan *logic* lain hanya dapat mengakses apa yang dikembalikan *query* SQL melalui output parameter.



Input & Output Parameter SQL

a) Input Parameter

Input parameter bersifat opsional. Untuk mereferensikan input parameter dalam SQL statement, gunakan awalan @, misalnya @CustomInputParameter.

b) Output Parameter

SQL dalam OutSystems selalu memiliki dua output parameter, meskipun query yang dijalankan tidak mengembalikan result. 2 parameter tersebut adalah :

- List

List berisikan result yang dikembalikan oleh query. List kosong akan dikembalikan apabila query tidak mengembalikan result.

- Count

Jumlah record yang dikembalikan oleh query tanpa mempertimbangkan properti SQL Max Records.



Using API

Dengan API, developer dapat mengintegrasikan aplikasi yang mereka kembangkan dengan sistem external, dan mengakses fitur OutSystems secara terprogram. Biasanya kita menggunakan REST API dalam melakukan pertukaran data dalam pengembangan aplikasi di Outsystems.

a). Expose Rest API

Expose API method untuk memungkinkan sistem lain mengambil atau memanipulasi informasi.

b). Consume Rest API

Saat developer perlu mengambil atau memanipulasi informasi dari sistem lain dan sistem tersebut menyediakan REST API, maka developer dapat menggunakan Consume Rest API di aplikasi Anda.

Developer dapat melakukan consume beberapa API method (*all* atau *selected*) ataupun *single* API method saja.

Audit Trail

Definisi

Audit trail adalah fitur yang terdapat pada suatu aplikasi yang berfungsi untuk mencatat semua aktivitas user ketika mengakses sebuah aplikasi. Aktivitas tersebut akan tercatat dalam bentuk log history, mulai dari data apa yang diakses, hingga waktu data tersebut diakses.

Action yang akan di-record




INSERT



UPDATE



DELETE




Audit Trail

Fungsi

- Menjamin seluruh perubahan data pada aplikasi Low Code terdokumentasi dengan rapi
- Standarisasi seluruh aplikasi Low Code sesuai dengan standar aplikasi di BCA.

LocoAuditTrail

LocoAuditTrail adalah Dashboard log aktivitas Audit Trail yang dimaintain oleh tim Digital Factory, dashboard ini dapat diakses oleh PIC aplikasi untuk melihat log perubahan data pada aplikasi Low Code.





Object Storage

Definisi

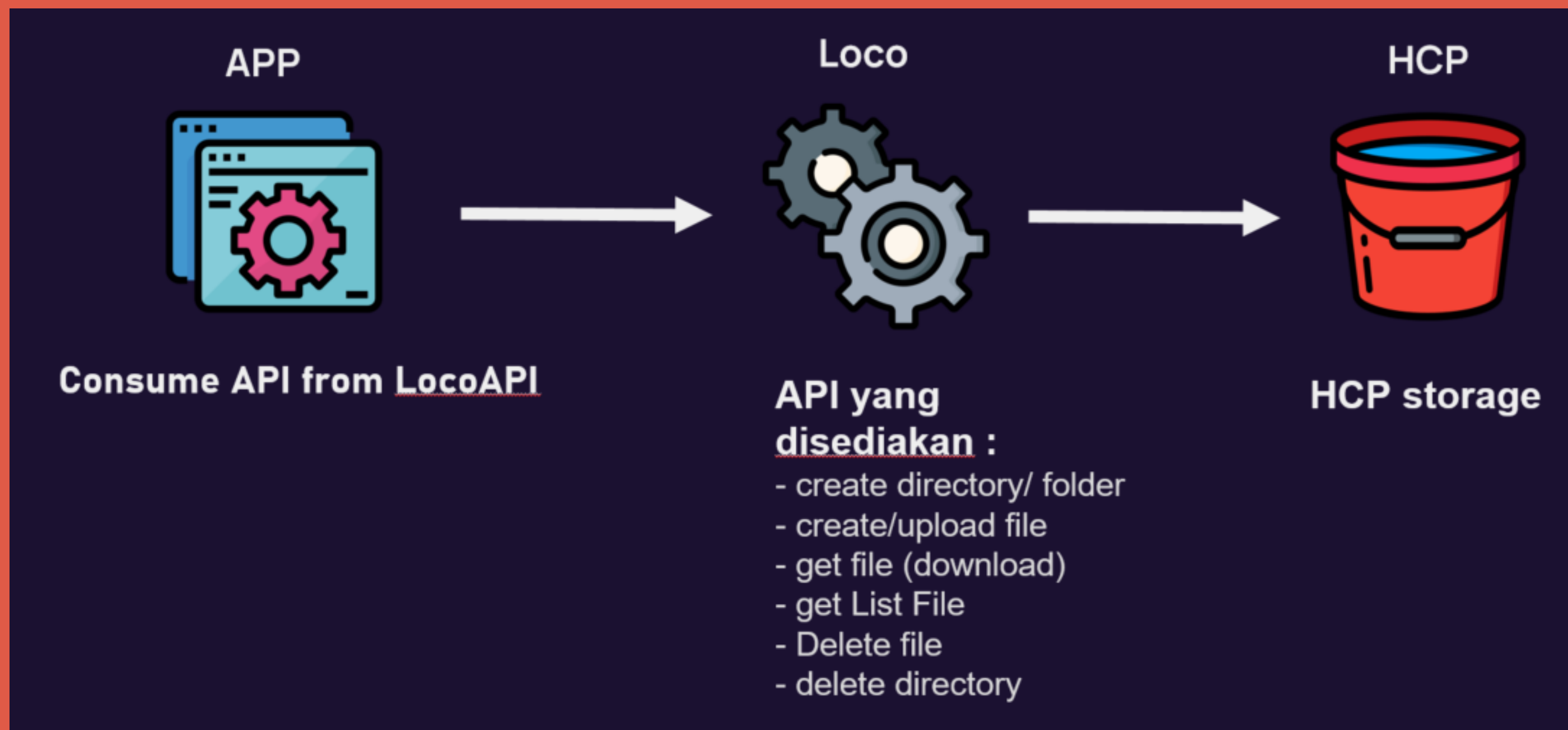
Object Storage adalah sebuah arsitektur penyimpanan data yang biasa disimpan dalam bentuk file. Storage ini mampu menyimpan data hingga skala exabytes. Oleh karena itu, Object Storage dianggap sebagai solusi kebutuhan ruang penyimpanan data berskala besar.

Benefit

1. Memiliki kapasitas yang besar
2. Hemat Biaya
3. Cocok digunakan untuk menyimpan file dengan format yang beragam

Di BCA, kita menggunakan ***Hitachi Content Platform (HCP)*** sebagai object storage.

Cara Akses Data HCP





Type File yang Disimpan HCP

txt, img, png, jpg, jpeg, pdf, mp3, mp4.

Ketika diupload, file akan di convert ke base64.

Ketika di download, file akan di convert kembali menjadi binary.

Tetapi file akan disimpan dalam format sebenarnya pada HCP storage.

