Día 2

23/Mar2012023

Manage Users and Groups

User Types

Los grupos preparacnan las bases para la organización y seguridad de la aplicación.

- Administratos → Permiso por defecto por administrar la app. ← diseñar obj admin obj acceso a la consola de admin.
- Usuarios básicos → en primera instancia no tienen acceso a nada hasta ser agregado a un grupo

* Appian NO puede eliminar usuarios solo puede activarlos o desactivarlos

Se puede transformar un usuario básico en un desarrollador agregandolo al grupo: "Designers System Group".

Agregar usuarios

- Manual Addition → Se realiza primariamente en los entornos de desarrollo o de pruebas
- Automate → En ambientes productivos se automatiza este proceso mediante la autenticación LDAP y SAML accediendo desde la consola de administración.

Why Groups?

Los grupos se usan para proteger la edición de objetos de diseño y para definir quien puede ver determinadas funciones de la aplicación

Razones por las cuales se usan los grupos

1. Seguridad
2. Acceso a las funciones de la aplicación
3. Tareas y mensajería

Best Practice: se debe establecer la seguridad de una aplicación usando grupos NO usuarios individuales.

23/Marzo/2023

Application Groups

*Best Practice → todas las applicaciones deberían contener los siguientes grupos

- All Users
- Administrators
- Functional & Role-Based → Grupos creados para usuarios empresariales y difieren de una app a otra.

*Best Practices en la creación de grupos

- Crear ~~grupos~~ solo los grupos necesarios
- No crear grupos por adelantado
- Los grupos no deben reflejar un organigrama completo
- Crear los grupos sobre la marcha

SUPPORT → The Appian Playbook → Para revisar más buenas practicas

Create Groups

1. Crear primero el grupo "All Users", es el grupo principal para el resto de grupos de la aplicación.
2. Crear grupo "Administrators" y utilizarlo para proteger el grupo "All Users"
3. Otros grupos heredarán la seguridad de grupo "All Users"
4. NO agregar usuarios al grupo "All Users"

Ive got my groups whats next?

Tareas después de configurar mis grupos.

1. Agregar usuarios a mis grupos
2. Proteger la aplicación

23/Marzo/2022

Expressions: Transform your Data

- **Using the Expression Editor**
  - Crear una ~~expr~~ regla de expresión independiente
  - Uso de las diferentes secciones del editor
  - Construir una expresión utilizando diferentes componentes

> Es importante guardar los cambios constantemente ya que NO se autoguarda

- **Data Types (Tipos de datos)**

Datos
- Primitivos → text, números, booleanos y fechas
- Complejos (Estructura de campos predefinidos) → Ej: DataSubset para la información recuperada de la DB
- Personalizados (Custom) → No almacena datos sino que solo admite la organización de los datos

- **Appian Functions (más comunes)**

- Logical functions
  * if (and, or)
- Funciones de fecha y hora
  * today() → fecha de hoy
  * now() → fecha y hora
  * calWorkdays → #de días hábiles entre dos fechas
- Funciones matemáticas
  * operadores (+, -, x, ÷)
- Funciones de texto
  * text
  * LoggedInUser() → Devuelve el usuario que está interactuando actualmente con una pantalla, ya sea un formulario, registro o informe.
  * a!isusermemberofgroup() para verificar si un usuario es parte de un determinado grupo

Las funciones que inician con "a!" son las funciones SAIL de Appian y en su mayor parte funcionan de la misma manera que otras que no tienen el prefijo

SAIL ⇒ Self-Assembling Interface Layer
"Capa de interfaz de autoensamblaje".

23/NOV/2023

• Variables

- Representan datos dinámicos se debe de colocar el prefijo "r!" siempre que se quiera utilizar la variable en el editor.
- También se pueden definir constantes y hacer referencia a estas en multiples expresiones. (Enums)

¿mas menos?

```
a! local variables(
    local!a : 1,
    local!b : 2,
    local!a + local!b
)
```

Ej <= a! para definir variables locales

• Arrays

- Definición de arreglo: Colección de datos que se relacionan entre sí de alguna manera
- Cómo navegar en un arreglo: Mediante índices, en Appian inician en 1
- Tipos de arreglos:

Sintaxis:

{"elemento1", "7.12", "rojo24"}

- Diccionario → tipo de dato más común cuando se trata de una matriz de tipos de datos, con este cada valor está vinculado a una "clave" específica.

Ej:

```
{
  nombre: "Tania Luis",
  edad: 26,
  sexo: "Femenino",
}
```

a!Map → permite que las aplicaciones conozcan los tipos de datos subyacentes de los conjuntos de datos eliminar las conversiones de datos.

Ej: a!Map(nombre: "Tania", edad: 26, sexo: "Femenino")

- make + navega en la matriz index()

↓

Recuperar datos específicos

23/Marzo/2023

- **Conversions** → Cast()

Convertir un valor de un tipo de dato en otro

- todate()
- toboolean()

Ej de Any Type (Text) a Text:

- to string()
- to uniformstring() → todos los valores de un array en strigs distintos ej

{"honda", "sedan", 2005, 16597}

↓ to uniformstring()

{"honda", "sedan", "2005", "16597"}

- **Calling Objects**

Business Rules:

- Reglas de expresión
- Interfaces
- Integraciones
- Objetos de decisión
- Constantes

Todas las reglas de negocio pueden ser excepto las constantes referenciadas con: el prefijo

rule!

Para las constantes se usa el prefijo

cons!

Ej: → rule!NombreObjeto
cons!NombreObjeto

- **Testing a Troubleshooting Expressions**

- Es importante (sino más bien crucial) cubrir todos los escenarios posible al escribir expresiones
- Appian permite guardar casos de prueba
- También se puede acceder al historial y versiones anteriores de un objeto al hacer click en el engrane.

definiciones formales

Constante: valor literal que puede ser llamado desde cualquier expresión, y puede ser reutilizado en múltiples objetos a través de la aplicación.

Regla de expresión: Sentencia que se evalúa para devolver un valor