

PRIMEROS EJERCICIOS

Crear base de datos

The screenshot shows a database management interface for a server named 'buwg'. The left sidebar displays a tree view with 'Funciones', 'Procedimientos', and 'Tablas'. The main window shows the execution of SQL queries. The first query creates a table named 'address' with columns: address_id (int(11) not null), shipping_address (varchar(100)), unit_number (int(5)), city (varchar(100)), state_or_province (varchar(2)), and postal_code (int(5)). The second query is an ALTER TABLE statement to add a primary key on 'address_id'. The third query is a MODIFY statement to set 'address_id' as auto-incrementing. Below the SQL editor, a table view shows the data inserted into the 'address' table:

address_id	shipping_address	unit_number	city	state_or_province	postal_code
1	47 New Saddle Ave.	NULL	Goose Creek	SC	29445
2	7208 Oakland Drive	50	Ravenna	OH	44266
3	358 Mayfair Dr.	23	Lynn	MA	01902

Crear CDT

Create Data Type

- Create from scratch
- Duplicate existing data type
- Create from database table or view
- Import XSD

Data Source*

jdbc/Appian (Tomcat)

Table or View*

address

CANCEL

CONTINUE

Create Data Type

Namespace *
urn:com:appian:types:ADV
Formatted as a URI, for example 'urn:com:appian:types:COB' for a client onboarding application

Name *
ADV_Address

Description
CDT para prueba de interfaces

Column Name	Field Name	Column Type	Field Type	Key	Nulls	Unique			
address_id	addressid	INT	Number (Integer)		<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑	↓	×
shipping_address	shippingAddress	VARCHAR(100)	Text		<input type="checkbox"/>	<input type="checkbox"/>	↑	↓	×
unit_number	unitNumber	INT	Number (Integer)		<input type="checkbox"/>	<input type="checkbox"/>	↑	↓	×
city	city	VARCHAR(100)	Text		<input type="checkbox"/>	<input type="checkbox"/>	↑	↓	×
state_or_province	stateOrProvince	VARCHAR(2)	Text		<input type="checkbox"/>	<input type="checkbox"/>	↑	↓	×

Crear constantes y reglas de expresión

Create Constant

Create from scratch Duplicate existing constant

Name *
ADV_DSE_ADDRESS

Description
Constante para entidad

Type *
Data Store Entity
 Array (multiple values)

Data Store *
ADV_DS

Entity *
ADV_Address

Environment Specific ?
 Different environments need to have different values for this constant

```
1. allLocalVariables(  
2.   localQueryResults:  
3.     alqueryentity:  
4.       entity: const(ADV_DSE_ADDRESS,  
5.         query: alquery(  
6.           mappingInfo: alpagingInfo(1, -1),  
7.           logicalExpression: alqueryLogicalExpression(  
8.             operator: "AND",  
9.             ignoreFilterWithEmptyValues: true,  
10.            filters: [  
11.              alqueryFilter(  
12.                field: "addressid",  
13.                operator: "in",  
14.                value: ritaddressid  
15.              )  
16.            ]  
17.          )  
18.        )  
19.      ),data,  
20.      cast(  
21.        typeof('type[urn:com:appian:types:ADV]ADV_Address'()),  
22.        localQueryResults  
23.      )  
24.    )
```

Rule Input Name	Expression	Value
addressid (Number (Integer))	1	

Interfaz con componentes dinámicos

Create Interface

Create from scratch Duplicate existing interface

Name *

ADV_DynamicAddressList

Description

Interfaz de componentes dinámicos

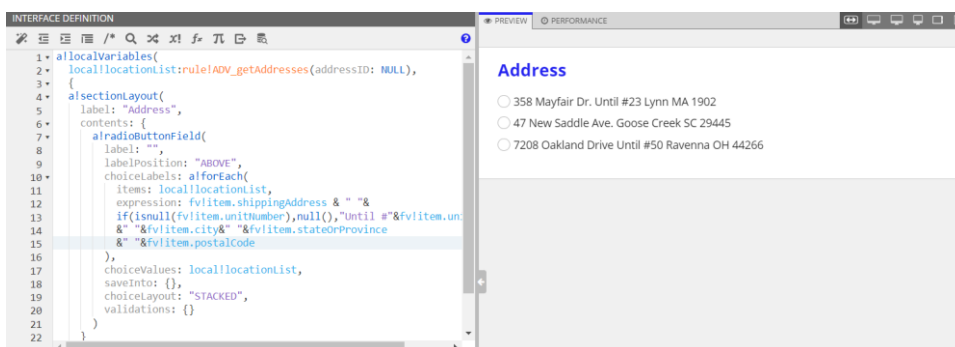
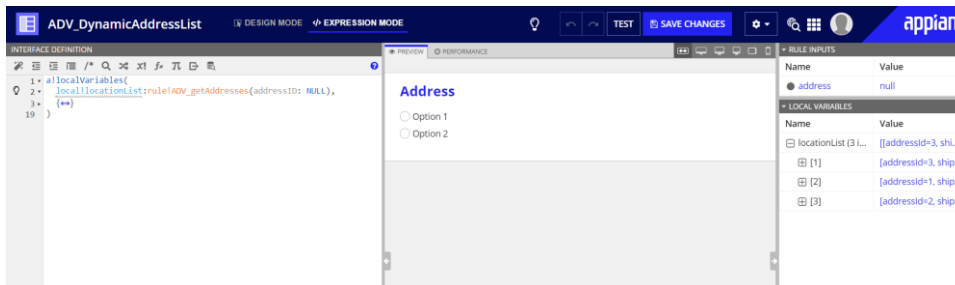
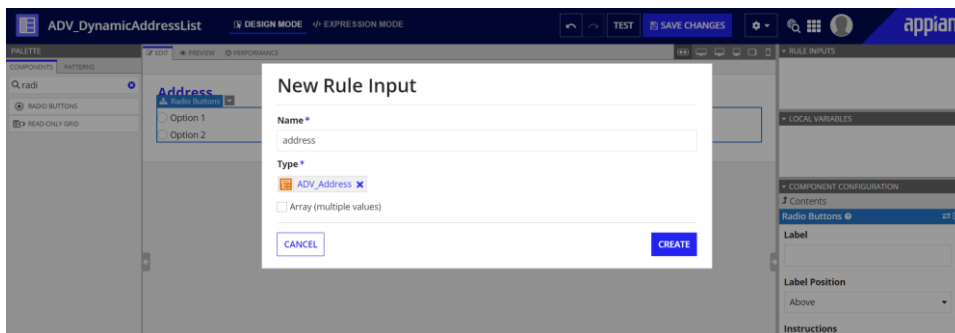
Save In *

ADV Interfaces

Create New Rule Folder

CANCEL

CREATE



SEGUNDO EJERCICIO

Añadir variables locales

```
INTERFACE DEFINITION
1  allocalVariables(
2    local!locationList:rule!ADV_getAddresses(addressID: NULL),
3    local!addingAddress: false(),
4    local!newAddress: 'type!{urn:com:appian:types:ADV}ADV_Address'(),
5  {
```

Formulario de nueva dirección

Address

- 358 Mayfair Dr. Until #23 Lynn MA 1902
- 47 New Saddle Ave. Goose Creek SC 29445
- 7208 Oakland Drive Until #50 Ravenna OH 44266

New Address		
Shipping Address *	Unit Number	
<input type="text"/>	<input type="text"/>	
City *	State/Province *	Postal/ZIP Code *
<input type="text"/>	<input type="text"/>	<input type="text"/>

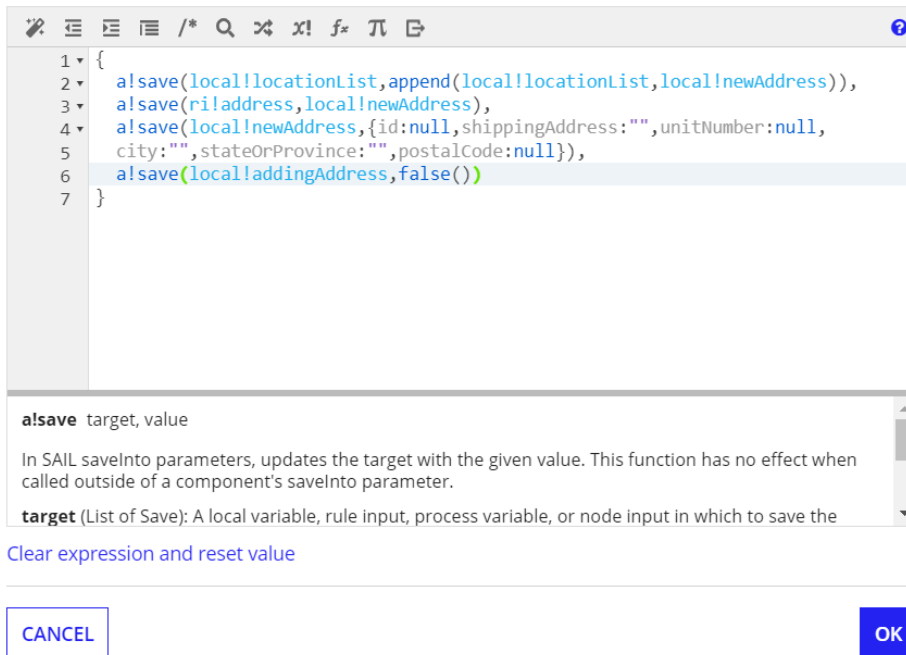
Añadir botón de añadir

New Address		
Shipping Address *	Unit Number	
<input type="text"/>	<input type="text"/>	
City *	State/Province *	Postal/ZIP Code *
<input type="text"/>	<input type="text"/>	<input type="text"/>

Configurar el añadir y seleccionar elementos

Save Value To (List of Save)

One or more variables that are updated with the button value when the user presses it. Use `alsave()` to save a modified or alternative value to a variable.



```
1 {
2   alsave(local!locationList, append(local!locationList, local!newAddress)),
3   alsave(ri!address, local!newAddress),
4   alsave(local!newAddress, {id:null, shippingAddress:"", unitNumber:null,
5     city:"", stateOrProvince:"", postalCode:null}),
6   alsave(local!addingAddress, false())
7 }
```

alsave target, value

In SAIL `saveInto` parameters, updates the target with the given value. This function has no effect when called outside of a component's `saveInto` parameter.

target (List of Save): A local variable, rule input, process variable, or node input in which to save the

[Clear expression and reset value](#)

CANCEL OK

Address

- 358 Mayfair Dr. Until #23 Lynn MA 1902
- 47 New Saddle Ave. Goose Creek SC 29445
- 7208 Oakland Drive Until #50 Ravenna OH 44266
- s Until #51 s s 51

[+ Add New Address](#)

Dynamic Link

Label
Dynamic Link

Value
true

Save Value To
local!addingAddress ✕

Visibility
 Always show

TERCER EJERCICIO

Crear un CDT

Create Data Type

- Create from scratch
- Duplicate existing data type
- Create from database table or view
- Import XSD

Namespace*

urn:com:appian:types:ADV

Formatted as a URI, for example 'urn:com:appian:types:COB' for a client onboarding application

Name*

ADV_Items

Description

Provides the structure for each item in the purchase request

CANCEL

CREATE

Fields (6)


Name	Type	Length	Array	Key			
idItem	Number (Integer)		<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑	↓	×
description	Text	255	<input type="checkbox"/>		↑	↓	×
category	Text	255	<input type="checkbox"/>		↑	↓	×
qty	Number (Integer)		<input type="checkbox"/>		↑	↓	×
unitPrice	Number (Decimal)		<input type="checkbox"/>		↑	↓	×
amount	Text	255	<input type="checkbox"/>		↑	↓	×

[New Field](#)

ADV_DS

Data Store for the ADV app

Data Source

jdbc/Appian (Tomcat) 

Data Entities

ADV_Address *ADV_Address*

items *ADV_Items*

 **Add Entity**

Create Constant

Create from scratch Duplicate existing constant

Name *
ADV_PURCHASE_REQUEST_CATEGORIES

Description
Constant to hold the categories for the Purchase Request grid

Type *
Text

Array (multiple values)

Values (4) ?
Office Supplies
Hardware
Software
Miscellaneous



Crear una interfaz con variables

Create Interface

Create from scratch Duplicate existing interface

Name *
ADV_EditableGrid

Description
Interfaz para grilla

Save In *
ADV Interfaces  

Create New Rule Folder

```
INTERFACE DEFINITION
1  a!localVariables(
2    local!newGridRow:'type!{urn:com:appian:types:ADV}ADV_Items'(),
3  a!formLayout(↔)
30 )
```


New Rule Input

Name *

items

Type *

ADV_Items

Array (multiple values)

CANCEL

CREATE

Agregar grilla editable

The image displays two screenshots of a software interface for configuring an 'Editable Grid' component. The top screenshot shows the grid with columns for Description, Category, Qty, Unit Price, and Amount, and a 'No items available' message. The bottom screenshot shows the same grid with a dropdown menu in the Category column. Both screenshots include a 'CANCEL' button and a 'SUBMIT' button. The right sidebar shows the configuration panel for the 'Editable Grid' component, including 'RULE INPUTS', 'LOCAL VARIABLES', and 'COMPONENT CONFIGURATION'.

Top Screenshot Configuration:

- RULE INPUTS:**

Name	Value
cancel	null
items	null
- LOCAL VARIABLES:**

Name	Value
newGridRow	[iditem=, descripti...
- COMPONENT CONFIGURATION:**
 - Editable Grid:**
 - Header Cells:
 - Description (Header Cell)
 - Category (Header Cell)
 - Qty (Header Cell)
 - Unit Price (Header Cell)
 - Amount (Header Cell)
 - Header Cell
 - ADD ITEM

Bottom Screenshot Configuration:

- RULE INPUTS:**

Name	Value
cancel	null
items	null
- LOCAL VARIABLES:**

Name	Value
newGridRow	[iditem=, descripti...
- COMPONENT CONFIGURATION:**
 - Row:**
 - Contents:
 - Text (Text)
 - Dropdown (Dropdown)
 - Integer (Integer)
 - Decimal (Decimal)
 - Text (Text)
 - Rich Text
 - ADD COMPONENT

Añadir un link

Rows (List of Variant)

Array of grid rows created with `a!gridRowLayout()`.

```
1 {a!forEach(  
2   items: r!items,  
3   expression:a!gridRowLayout(  
4     contents: {  
5       a!textField(  
6         label: "Text",  
7         labelPosition: "ABOVE",  
8         value: fv!item.description,  
9         saveInto: {fv!item.description},  
10        refreshAfter: "UNFOCUS",  
11        validations: {}  
12      )  
13      a!dropdownField(  
14        label: "Dropdown",  
15        labelPosition: "ABOVE",
```

Place cursor on function, rule, or constant to display help

CANCEL

OK

Item 1

```
1 a!save(r!items, append(r!items, local!newGridRow))
```

Place cursor on function, rule, or constant to display help

CANCEL

OK

Editable Grid

Description	Category	Qty	Unit Price	Amount
<input type="text"/>	--- Select a Value ---			0

[Add New Item](#)

CANCEL

SUBMIT

Añadir funcionalidad al botón eliminar

Display Value



Use editor



Configure items

Styled Icon ▶



ADD RICH TEXT

Styled Icon

Icon

(Clear)

Alternative Text

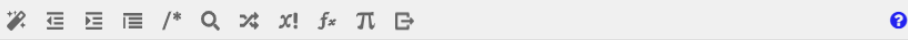
Caption

Link

Dynamic Link (Dynamic Link) ▶

Save Value To (List of Save)

One or more variables that are updated with the link's value when the user clicks it. Use `alsave()` to save a modified or alternative value to a variable.



```
1  {{alsave(ri!items, remove(ri!items, fv!index))}}
2  }}
```

Place cursor on function, rule, or constant to display help

[Clear expression and reset value](#)

CANCEL

OK

Editable Grid					
Description	Category	Qty	Unit Price	Amount	
	--- Select a Value ---			0	X

[Add New Item](#)

CANCEL

SUBMIT