

# Laboratorio 2

## Contenidos

Objetivos.....	2
Creación Interfaz .....	2
Rule Input.....	2
Agregar componentes.....	3
Pruebas sobre el interfaz .....	7
Conclusiones .....	8

## Objetivos

El objetivo del laboratorio es mostrar el procedimiento para la creación de la parte visual de nuestra aplicación. Crearemos una interfaz, que servirá a los usuarios para añadir los datos de los vehículos que se desee incorporar a nuestro sistema.

En dicho interfaz también incluiremos ciertas validaciones sobre estos campos.

Con ello tendremos una visión sobre cómo funciona Appian respecto a la visualización y edición de la información.

## Creación Interfaz

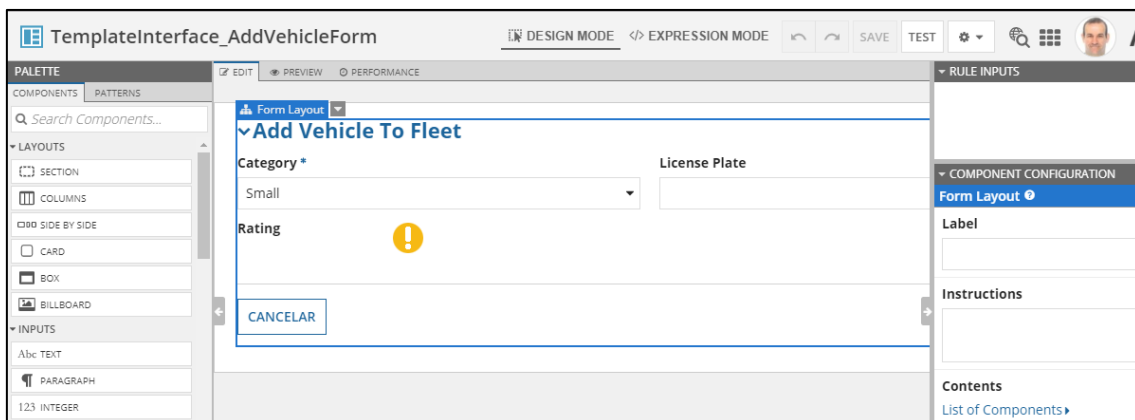
### Descripción:

Para la creación del interfaz, nos apoyaremos en una interfaz ya creada, a la que le agregaremos más información.

### Pasos a realizar:

- Dentro de nuestra aplicación XXX\_VFM, pulsamos New -> Interface.
- Duplicate Existing Interface.
- Interface to duplicate: TemplateInterface\_AddVehicleForm.
- Name: XXX\_AddVehicleForm.
- Description: interfaz de añadir vehículo a la flota.
- Save in: XXX\_Interfaces.
- Create.

Una vez creada la interfaz, se nos abrirá una ventana para diseñarla.



**Nota:** Las partes que componen este entorno de desarrollo de interfaces vienen comentadas en el laboratorio 0, apartado 3.3 interfaces.


## Rule Input

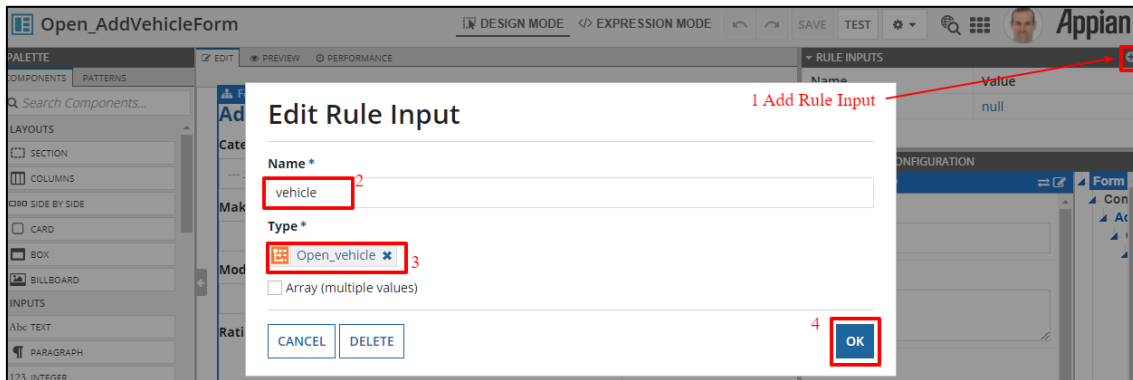
### Descripción:

Las interfaces pueden tener parámetros de entrada, denominados **Rules Input**. Estas Rule Input se usarán tanto para obtener la información que se vaya a mostrar en el interfaz, como para guardar la información insertada en el mismo.

En nuestro laboratorio, incorporaremos un objeto de tipo vehículo como ruleinput del interfaz.

#### Pasos a realizar:

- Abrimos nuestro interfaz creado (XXX\_AddVehicleForm)
- En la parte superior derecha, pulsamos en el icono  para añadir parámetros de entrada
  - **Name:** vehicle
  - **Type:** XXX\_Vehicle
  - Create.



Para hacer uso de estas Rule Inputs, haremos referencia a dichos objetos precediéndolos de ri!

Por ejemplo, en nuestro laboratorio: ri!vehicle.make, para acceder a la marca del vehículo.

## Agregar componentes

#### Descripción:

Los componentes son los elementos que conforman el interfaz. Pueden ser elementos organizadores de la información (sectionLayout, columnsLayout, etc.), o elementos visuales para recoger información (textField, dropdownField, fileUploadField, etc.).

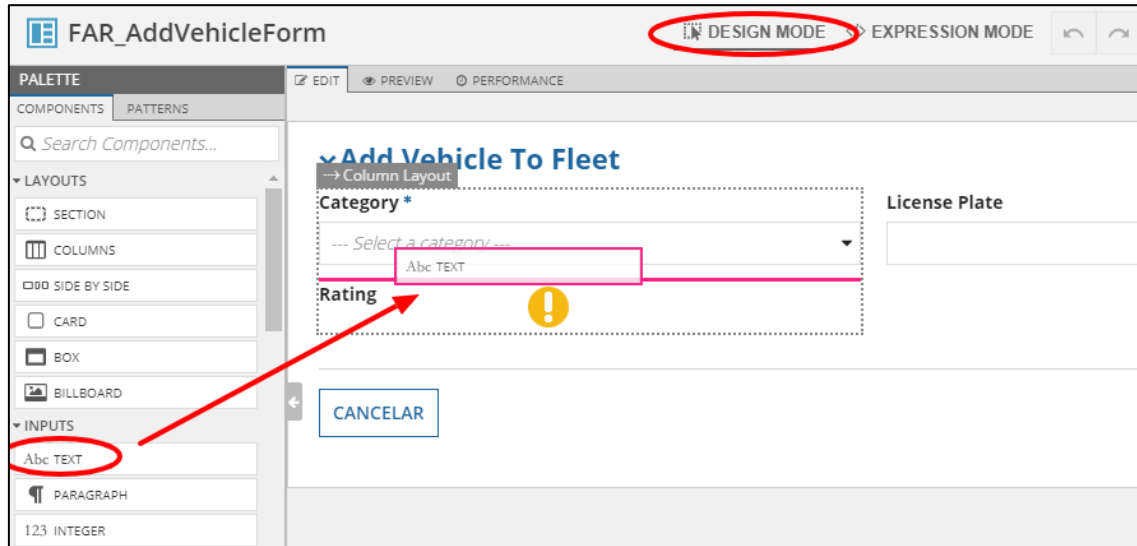
Al interfaz del que hemos partido, le agregaremos más componentes, para que el usuario puede aportar más datos del vehículo a añadir a la flota.

#### Pasos a realizar:

- Abrimos nuestro interfaz creado (XXX\_AddVehicleForm) en modo Design

- Del panel izquierdo (Component palette), arrastrar sobre nuestro panel central (Canvas) un componente de tipo texto, entre los componentes Category y Rating.

Añadirlo tal y como se muestra en la siguiente imagen:



En el panel de la derecha (Component Configuration), completar los siguientes datos del campo que acabamos de incorporar a nuestro interfaz:

PROPIEDAD	DESCRIPCION	VALOR
Label	Etiqueta que se muestra	Make
Display Value	Valor que se muestra	ri!vehicle.make
Saved Input To	Campo donde se guardará el valor introducido	ri!vehicle.make
Required	Indica si el campo se ha de completar obligatoriamente	true

Hay que agregar más campos a nuestro interfaz para que el resultado sea el siguiente:

Modificar el componente de tipo dropdownField:

- **Type Field:** dropdownField
- **Label:** Category
- **Placeholder Label:** --- Select a category ---
- **Choice Labels:** cons!Open\_VEHICLE\_CATEGORIES
- **Choice Values:** cons!Open\_VEHICLE\_CATEGORIES
- **Selected Value:** ri!vehicle.category
- **Save Selection To:** ri!vehicle.category
- **Required:** true

**Nota:** para hacer uso de objetos de tipo constante hay que preceder su nombre de cons!.

Al final del proceso, debemos tener incluidos los siguientes campos con sus propiedades como se indican:

Type Field	Label	Display Value	Saved Into To	Required
Text	Make	ri!vehicle.make	ri!vehicle.make	True
Text	Model	ri!vehicle.model	ri!vehicle.model	True
Text	LicensePlate	ri!vehicle.licensePlate	ri!vehicle.licensePlate	True
Integer	Mileage	ri!vehicle.mileage	ri!vehicle.mileage	True
Paragraph	Vehicle Condition	ri!vehicle.vehicleCondition	ri!vehicle.vehicleCondition	ri!vehicle.mileage > 100000

- Una vez hemos creado / modificado estos componentes, vamos a añadirle validaciones a algunos de ellos. Con estas validaciones, obligaremos a que el contenido incluido en ellos deba cumplir ciertos criterios. En caso de no cumplirse, se mostrará un mensaje de error.

Accederemos a la propiedad Validations de los campos, editándola como expresión, de la forma siguiente:

The screenshot shows the 'Validations' section of a field's properties in Apptio Designer. It includes options for 'Required Message', 'Read-only', 'Disabled', and 'Masked'. Below these, there is a 'Validations' section with a 'List of Text' link and an 'Edit as Expression' button. A red arrow points to the 'Edit as Expression' button, with the text 'Pulsamos aquí' (We click here) next to it.

- Componente LicensePlate. En la propiedad Validations incluiremos la siguiente validación:

```
{
  if(
    len(
      ri!vehicle.licensePlate
    ) <> 7,
    "Debe tener una longitud de 7",
    null
  )
}
```

Con esto, obligaremos a que la longitud de la matrícula sea de 7 caracteres

- Componente Mileage. En la propiedad Validations incluiremos la siguiente información:

```
{
  if(
    ri!vehicle.mileage > 200000,
    "You can only add cars with fewer than 200,000 miles",
    null
  )
}
```

Con esta validación impediremos que se inserten vehículos con más de 200000 kms:


- Pulsamos botón SAVE.
- Cambiar de modo de diseño, al modo Expression (veremos directamente las líneas de código de nuestro interfaz. Podemos pasar indistintamente de un modo a otro).


En el panel de la izquierda (Interface Definitor) buscar el campo Rating. Veremos que tiene una línea comentada. Descomentar esta línea, y comentar la anterior. La línea es:


```
icon:displayvalue(rule!Open_CalculateRating(mileage: ri!vehicle.mileage),
cons!Open_RatingChoices, cons!Open_RatingIcons, "STATUS_WARN")
```

Tratad de analizar y comprender la expresión utilizada. Pulsando sobre el nombre de una función, en el panel inferior se nos muestra una descripción de esta y sus parámetros de entrada requeridos.

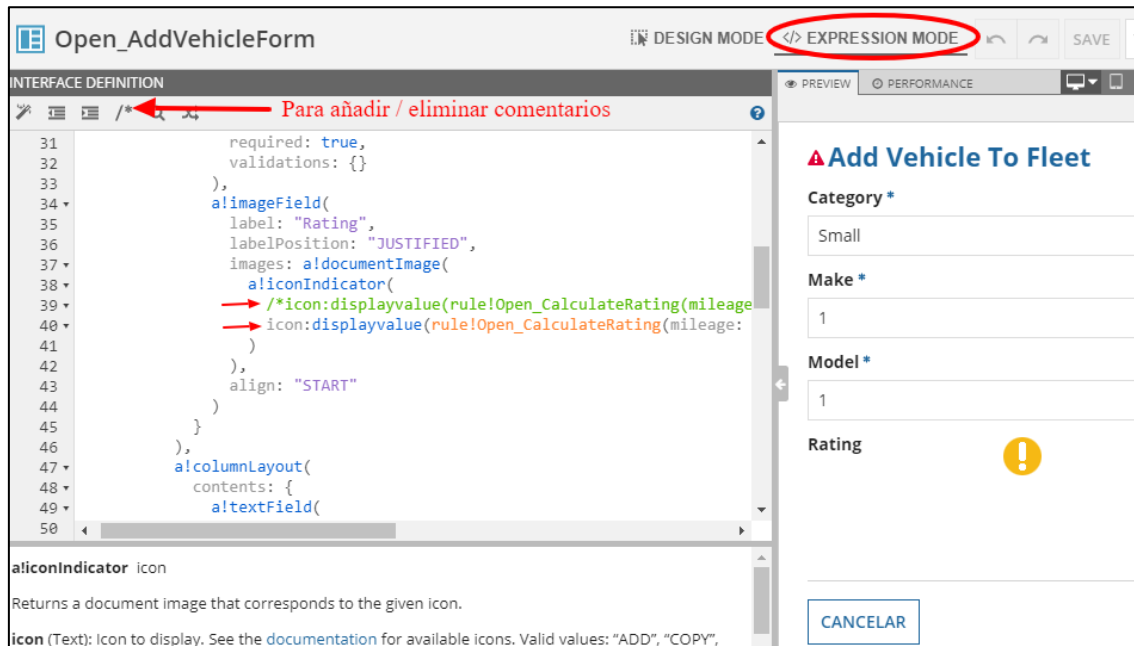
Con esta línea conseguimos que se muestre un icono en función de los kilómetros del vehículo

Si ri!vehicle.mileage <  5000

Si 5000 < ri!vehicle.mileage <  15000

Si ri!vehicle.mileage >  15000

En otro caso el resultado debe ser el siguiente:



- Pulsamos SAVE

Hasta este punto hemos creado un interfaz para poder añadir los datos del vehículo, le hemos añadido los campos necesarios, sus características, y reglas que deben cumplir.

## Pruebas sobre el interfaz

Estando nuestra interfaz abierta, podemos ir completando los campos que en ella aparecen.

Observaremos que en nuestro objeto vehicle que hemos agregado como rule input al interfaz (parte superior derecha), se van guardando los campos.

Podemos realizar también pruebas para comprobar que las validaciones que hemos incluido funcionan correctamente.

Pruebas a realizar:

- Verificar que los campos requeridos muestran mensaje de error caso de no estar completados (Si pulsamos botón Aceptar, nos deben salir los errores).
- Verificar que si incluimos en el campo mileage introducimos un valor superior a 200000, se nos muestra un error.
- Si el valor del campo mileage es superior a 100000, el campo 'Vehicle Condition' pasa a ser obligatorio.
- Verificar que la longitud del campo LicensePlate ha de ser 7.
- Verificar que, si completamos todos los campos, estos aparecen recogidos en nuestro objeto vehicle añadido como ruleInput.

**DESIGN MODE** <> **EXPRESSION MODE** [SAVE] [TEST] [Settings] [Search] [Appian]

**EDIT** [PREVIEW] [PERFORMANCE]

### ▼ Add Vehicle To Fleet

**Category \***  
 Economy

**Make \***  
 Text [renault]

**Model \***  
 clio

**Rating**  
 😊

**License Plate \***  
 1122JPO

**Mileage \***  
 13000

**Vehicle Condition**  
 Nice!

**RULE INPUTS**

Name	Value
vehicle	[id=, make=renaul...
● id	null
● make	renault
● model	clio
● licensePlate	1122JPO
● category	Economy
● vehicleCond...	Nice!
● lastUpdated	null
● nextService...	null
● mileage	13000
● pictureId	null

**COMPONENT CONFIGURATION**  
 Contents

## Conclusiones

En este laboratorio hemos creado el interfaz que usaremos para insertar los valores de los vehículos. En los campos del interfaz, hemos definido reglas sobre sus valores, y hemos realizado pruebas para comprobar que los valores se recogen en la regla de entrada (vehicle).

Copyright © 2019 Accenture  
 All rights reserved.

Accenture, its Signature, and  
 High Performance Delivered  
 are trademarks of Accenture.