

## Reto final Salesforce

Omar Alonso López Valenzuela

Liderly

¿Cómo puedo diseñar e implementar una solución integral en Salesforce que automatice el proceso de cotización y gestión de descuentos para estudiantes, garantizando eficiencia y precisión en el sistema de cobranza de la universidad?

Para lograr resolver esta propuesta se utilizaron algunos de los objetos estándar de Salesforce aprovechando las relaciones que tienen objetos como Contacts, Opportunities y Quotes principalmente.

Además, se implementaron campos personalizados a algunos de los objetos, lo que ayudaría a la implementación lógica de algunos valores solicitados.

Estos son los objetos y campos utilizados para almacenar los datos solicitados.

Campo deseado	Objeto estandar	Tipo de campo	Nombre del campo creado o campo estandar utilizado	Tipo de dato	Valores
Matricula	Contact	Personalizado	Id	Number(2,1)	EST-{00000}
Nombre	Contact	Estándar	Name	Name	
Estado	Opportunity	Estándar	Price Book	Price Book	
Fecha de nacimiento	Contact	Estandar	Birthdate	Date	
Telefono	Contact	Estándar	Phone	Phone	
Correo	Contact	Estándar	Email	Email	
Promedio alcanzado	Contact	Personalizado	Average Grade	Number(2,1)	
Periodo	Opportunity	Estándar	Type	Picklist	Six-monthly Four-monthly
Cantidad de materias	Quote	Estándar	Line Items	Roll-Up Summary (COUNT Quote Line Item)	
Costo total	Quote	Estándar	Subtotal	Roll-Up Summary (SUM Quote Line Item)	

Forma de pago	Opportunity	Personalizado	Payment Type	Picklist	Mensualidades Contado
Descuentos aplicables	Contact	Personalizado	Applicable discounts	Picklist (Multi-select)	Beca deportiva (10%) Beca por familiares docentes (30%) Beca por necesidad económica (30%)

Además, se establecieron las siguientes reglas de validación para definir las restricciones en la creación de nuevos elementos para cada objeto:

**Promedio alcanzado (Debe ser un número del 0-10 y solo tener un decimal)**

Contact Validation Rule

Help for this Page

[Back to Contact Validation Rules](#)

Validation Rule Detail

EditClone

Rule Name

Average\_between\_0\_and\_10

Active

✓

Error Condition Formula

Average\_Grade\_\_c >= 0 && Average\_Grade\_\_c <= 10

Error Message

Average Grade should be a number between 0 and 10.

Error Location

Top of Page

Description

Created By

Omar Alonso Lopez Valenzuela, 12/11/2024, 19:16

Modified By

Omar Alonso Lopez Valenzuela, 12/11/2024, 19:16

EditClone

Cantidad de materias: Quote Usar Line Items:

**Si es en periodo semestral, puede registrar hasta 7 materias.**

**Si es periodo cuatrimestral, puede registrar hasta 4 materias.**

**No permitir avanzar si se sobrepasa alguna de estas opciones.**

Quote Validation Rule

Help for this

[Back to Quote Validation Rules](#)

Validation Rule Detail

EditClone

Rule Name

Maximum\_subject\_quantity

Active

✓

Error Condition Formula

IF(ISPICKVAL( Opportunity.Type , "Six-monthly"), LinetitemCount <= 7, LinetitemCount <= 4)

Error Message

You exceeded the maximum quantity of subjects allowed for this period type.

Error Location

Top of Page

Description

Created By

Omar Alonso Lopez Valenzuela, 17/11/2024, 14:29

Modified By

Omar Alonso Lopez Valenzuela, 17/11/2024, 14:29

EditClone

Para la definición de precios y descuentos se creó primeramente un Price Book para cada estado que tuviera un campus y asignándole su respectivo precio al Producto Materia dentro de cada Price Book, por lo que al crear una oportunidad sólo se podrían seleccionar cuantas materias se ocupen pero del mismo valor dentro del campus.

Sede	Costos por materia
Guanajuato	\$17,000.00
Nuevo León	\$18,000.00
Jalisco	\$13,000.00
Querétaro	\$15,000.00

Price Books

Recently Viewed

New

5 items • Updated a few seconds ago

Search this list...

<input type="checkbox"/>	Price Book Name	Description	Last Modified Date	Active	
1	<input type="checkbox"/> Querétaro		17/11/2024, 15:02	<input checked="" type="checkbox"/>	<div></div>
2	<input type="checkbox"/> Jalisco		17/11/2024, 15:02	<input checked="" type="checkbox"/>	<div></div>
3	<input type="checkbox"/> Nuevo Leon		17/11/2024, 15:01	<input checked="" type="checkbox"/>	<div></div>
4	<input type="checkbox"/> Guanajuato	Costo de materias para alumnos de Guanajuato	17/11/2024, 14:36	<input checked="" type="checkbox"/>	<div></div>
5	<input type="checkbox"/> Standard Price Book		12/11/2024, 17:42	<input type="checkbox"/>	<div></div>

Posterior a esto se definió un trigger para que al crear un Quote (cotización) se establecieran los descuentos correspondientes dependiendo de la cantidad de materias:

- Descuentos por cantidad de materias
  - Si se cotiza solo una materia, su costo se cobra al 100%.
  - Si se cotizan 2 materias, el costo de la segunda baja 10%.
  - Si se cotizan 3 materias o más, el costo de la primera se cobra al 100% y las siguientes bajan 15%.
- Descuento de 5% con pago de contado

```

1 trigger ApplySubjectQuantityDiscount on Quote (after insert, after update) {
2     // Crear una lista para almacenar los QuotelineItems que necesitan ser actualizados
3     List<QuotelineItem> lineItemsToUpdate = new List<QuotelineItem>();
4
5     // Obtener los OpportunityIds de los Quotes en el Trigger
6     Set<Id> opportunityIds = new Set<Id>();
7     for (Quote q : Trigger.new) {
8         opportunityIds.add(q.OpportunityId);
9     }
10
11     // Consultar las oportunidades relacionadas
12     Map<Id, Opportunity> oppMap = new Map<Id, Opportunity>{
13         [SELECT Id, Payment_Type__c FROM Opportunity WHERE Id IN :opportunityIds]
14     };
15
16     // Iterar sobre las cotizaciones en el Trigger
17     for (Quote q : Trigger.new) {
18         // Obtener los productos asociados a esta cotización
19         List<QuotelineItem> lineItems = [
20             SELECT Id, UnitPrice, Quantity, TotalPrice, Discount
21             FROM QuotelineItem
22             WHERE QuoteId = :q.Id
23         ];
24
25         Integer productCount = lineItems.size();
26
27         if (productCount > 1) {
28             for (Integer i = 0; i < productCount; i++) {
29                 QuotelineItem lineItem = lineItems[i];
30                 if (productCount == 2 && i == 1) {
31                     // Si hay 2 productos, el segundo tiene 10% de descuento
32                     lineItem.Discount = 10;
33                 } else if (productCount >= 3 && i >= 1) {
34                     // Si hay 3 o más productos, los productos 2 en adelante tienen 15% de descuento
35                     lineItem.Discount = 15;
36                 }
37                 lineItemsToUpdate.add(lineItem);
38             }
39         }
40
41         // Verificar si la oportunidad tiene tipo de pago de contado
42         Opportunity opp = oppMap.get(q.OpportunityId);
43         if (opp != null && opp.Payment_Type__c == 'Cash Payment') {
44             // Aplicar un 5% de descuento adicional a cada QuotelineItem
45             for (QuotelineItem lineItem : lineItems) {
46                 // Sumar el 5% de descuento adicional al descuento existente
47                 lineItem.Discount += 5;
48                 lineItemsToUpdate.add(lineItem);
49             }
50         }
51     }
}

```

Y además se realizó otro trigger para adicionar a cada una de las materias los descuentos correspondientes a las becas disponibles:

- Crear campos y lógica para aplicar los diferentes tipos de descuentos:
  - Beca de excelencia (10% si el promedio es  $\geq 9.5$ )
  - Beca deportiva (10%)
  - Beca por familiares docentes (30%)
  - Beca por necesidad económica (30%)
- Asegurar que la suma total de descuentos no exceda el 60%

```
ApplySubjectQuantityDiscount.apxt | ApplyContactDiscount.apxt
Code Coverage: None | API Version: 62

1 trigger ApplyContactDiscount on Quote (after insert, after update) {
2     // Crear un conjunto para almacenar los IDs de las oportunidades relacionadas con los Quotes en el Trigger
3     Set<Id> opportunityIds = new Set<Id>();
4     for (Quote q : Trigger.new) {
5         opportunityIds.add(q.OpportunityId); // Agregar cada OpportunityId asociado al Quote
6     }
7
8     // Consultar las oportunidades relacionadas con los Quotes, incluyendo el ContactId asociado
9     Map<Id, Opportunity> oppMap = new Map<Id, Opportunity>{
10         [SELECT Id, ContactId FROM Opportunity WHERE Id IN :opportunityIds]
11     };
12
13     // Crear un conjunto para almacenar los ContactIds asociados a las oportunidades
14     Set<Id> contactIds = new Set<Id>();
15     for (Opportunity opp : oppMap.values()) {
16         if (opp.ContactId != null) {
17             contactIds.add(opp.ContactId); // Agregar el ContactId si existe
18         }
19     }
20
21     // Consultar los contactos relacionados, incluyendo sus campos relevantes
22     Map<Id, Contact> contactMap = new Map<Id, Contact>();
23     if (!contactIds.isEmpty()) { // Verificar si hay ContactIds antes de realizar la consulta
24         contactMap = new Map<Id, Contact>{
25             [SELECT Id, Applicable_discounts__c, Average_Grade__c FROM Contact WHERE Id IN :contactIds]
26         };
27     }
28
29     // Consultar los QuoteLineItems relacionados con los Quotes
30     Set<Id> quoteIds = new Set<Id>();
31     for (Quote q : Trigger.new) {
32         quoteIds.add(q.Id); // Agregar los IDs de los Quotes en el Trigger
33     }
34
35     // Crear un mapa para agrupar los QuoteLineItems por QuoteId
36     Map<Id, List<QuoteLineItem>> quoteLineItemMap = new Map<Id, List<QuoteLineItem>>();
37     if (!quoteIds.isEmpty()) { // Verificar si hay QuoteIds antes de consultar
38         for (QuoteLineItem lineItem : [
39             SELECT Id, UnitPrice, Quantity, TotalPrice, Discount, QuoteId
40             FROM QuoteLineItem
41             WHERE QuoteId IN :quoteIds
42         ]) {
43             // Agrupar los QuoteLineItems bajo su QuoteId correspondiente
44             if (!quoteLineItemMap.containsKey(lineItem.QuoteId)) {
45                 quoteLineItemMap.put(lineItem.QuoteId, new List<QuoteLineItem>());
46             }
47             quoteLineItemMap.get(lineItem.QuoteId).add(lineItem);
48         }
49     }
50 }
```

```

51 // Lista para almacenar los QuoteLineItems que necesitan ser actualizados
52 List<QuoteLineItem> lineItemsToUpdate = new List<QuoteLineItem>();
53
54 // Procesar cada Quote en el Trigger
55 for (Quote q : Trigger.new) {
56     // Obtener la oportunidad relacionada con el Quote
57     Opportunity opp = oppMap.get(q.OpportunityId);
58
59     if (opp != null && opp.ContactId != null) { // Verificar que la oportunidad y el contacto existen
60         // Obtener el contacto relacionado con la oportunidad
61         Contact contact = contactMap.get(opp.ContactId);
62
63         if (contact != null) { // Verificar que el contacto existe
64             // Inicializar el descuento total
65             Decimal totalDiscount = 0;
66
67             // Agregar un 10% de descuento si el promedio es mayor o igual a 9.5
68             if (contact.Average_Grade__c >= 9.5) {
69                 totalDiscount += 10;
70             }
71
72             // Verificar los valores seleccionados en el picklist múltiple
73             String discounts = contact.Applicable_discounts__c;
74             if (discounts != null) {
75                 if (discounts.contains('Beca deportiva 10%')) {
76                     totalDiscount += 10; // Agregar un 10% por Beca deportiva
77                 }
78                 if (discounts.contains('Beca por familiares docentes 30%')) {
79                     totalDiscount += 30; // Agregar un 30% por familiares docentes
80                 }
81                 if (discounts.contains('Beca por necesidad economica 30%')) {
82                     totalDiscount += 30; // Agregar un 30% por necesidad económica
83                 }
84             }
85
86             // Limitar el descuento total a un máximo del 60%
87             totalDiscount = Math.min(totalDiscount, 60);
88
89             // Aplicar el descuento calculado a los QuoteLineItems relacionados
90             if (quoteLineItemMap.containsKey(q.Id)) {
91                 for (QuoteLineItem lineItem : quoteLineItemMap.get(q.Id)) {
92                     lineItem.Discount = totalDiscount; // Asignar el descuento total al campo Discount
93                     lineItemsToUpdate.add(lineItem); // Agregar el QuoteLineItem a la lista para actualizar
94                 }
95             }
96         }
97     }
98 }
99
100 // Actualizar los QuoteLineItems con los descuentos aplicados

```

Se creó una plantilla para los correos electrónicos enviados al crearse un Quote con la herramienta de las páginas de Visualforce, y para el envío automático de estos correos se implementó un trigger más:

- Pago total
- Becas aplicadas
- Costo después de descuentos
- Forma de pago
  - Si es en mensualidades, mostrar la fecha de cada pago:
    - La inscripción se paga el 10 de julio.
    - Cada mensualidad se paga el día 10 de cada mes (agosto, septiembre, octubre, noviembre y diciembre).

- Si la transacción es de contado, se muestra un pago único el 10 de julio.

```
1 <apex:page standardController="Quote" renderAs="pdf" showHeader="false">
2   <h1>Resumen de Cotización</h1>
3   <p><strong>Nombre del estudiante:</strong> {!Quote.Contact.Name}</p>
4   <p><strong>Matrícula:</strong> {!Quote.Contact.Id}</p>
5   <p><strong>Forma de pago:</strong> {!Quote.Opportunity.Payment_Type__c}</p>
6   <p><strong>Total antes de descuentos:</strong> {!Quote.Subtotal}</p>
7   <p><strong>Descuentos aplicados:</strong> {!Quote.Contact.Applicable_discounts__c}</p>
8   <p><strong>Total después de descuentos:</strong> {!Quote.TotalPrice}</p>
9
10 <apex:repeat value="{!Quote.QuoteLineItems}" var="lineItem">
11   <p><strong>Producto:</strong> {!lineItem.Product2.Name}</p>
12   <p><strong>Precio Unitario:</strong> {!lineItem.UnitPrice}</p>
13   <p><strong>Cantidad:</strong> {!lineItem.Quantity}</p>
14   <p><strong>Descuento:</strong> {!lineItem.Discount}</p>
15   <p><strong>Total:</strong> {!lineItem.TotalPrice}</p>
16   <br/>
17 </apex:repeat>
18
19 <h2>Calendario de Pagos</h2>
20
21 <apex:outputPanel rendered="{!Quote.Opportunity.Payment_Type__c == 'Monthly payments'}">
22   <p>Inscripción: 10 de julio</p>
23   <p><strong>Mensualidades:</strong></p>
24   <ul>
25     <!-- Cálculo de cada mensualidad -->
26     <apex:variable var="totalAfterDiscount" value="{!Quote.TotalPrice}" />
27     <apex:variable var="monthlyPayment" value="{!totalAfterDiscount / 6}" />
28     <li>10 de agosto: {!TEXT(monthlyPayment)} </li>
29     <li>10 de septiembre: {!TEXT(monthlyPayment)} </li>
30     <li>10 de octubre: {!TEXT(monthlyPayment)} </li>
31     <li>10 de noviembre: {!TEXT(monthlyPayment)} </li>
32     <li>10 de diciembre: {!TEXT(monthlyPayment)} </li>
33   </ul>
34 </apex:outputPanel>
35
36 <apex:outputPanel rendered="{!Quote.Opportunity.Payment_Type__c == 'Cash payment'}">
37   <p>Pago único: 10 de julio: {!TEXT(Quote.TotalPrice)}</p>
38 </apex:outputPanel>
39 </apex:page>
```

## Trigger:

```
1  ▼ trigger GenerateAndSendQuotePDF on Quote (after insert, after update) {
2      // Lista para almacenar los correos electrónicos a enviar
3      List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();
4
5  ▼  for (Quote quote : Trigger.new) {
6      // Generar la URL del PDF utilizando la página Visualforce
7      String pdfURL = '/apex/QuotePDF?id=' + quote.Id;
8
9      // Generar el cuerpo del correo
10     Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
11     email.setToAddresses(new String[] {quote.Contact.Email});
12     email.setSubject('Cotización para ' + quote.Contact.Name);
13     email.setHtmlBody('Hola ' + quote.Contact.Name + ',<br/>Adjunto encontrarás tu cotización.');
```