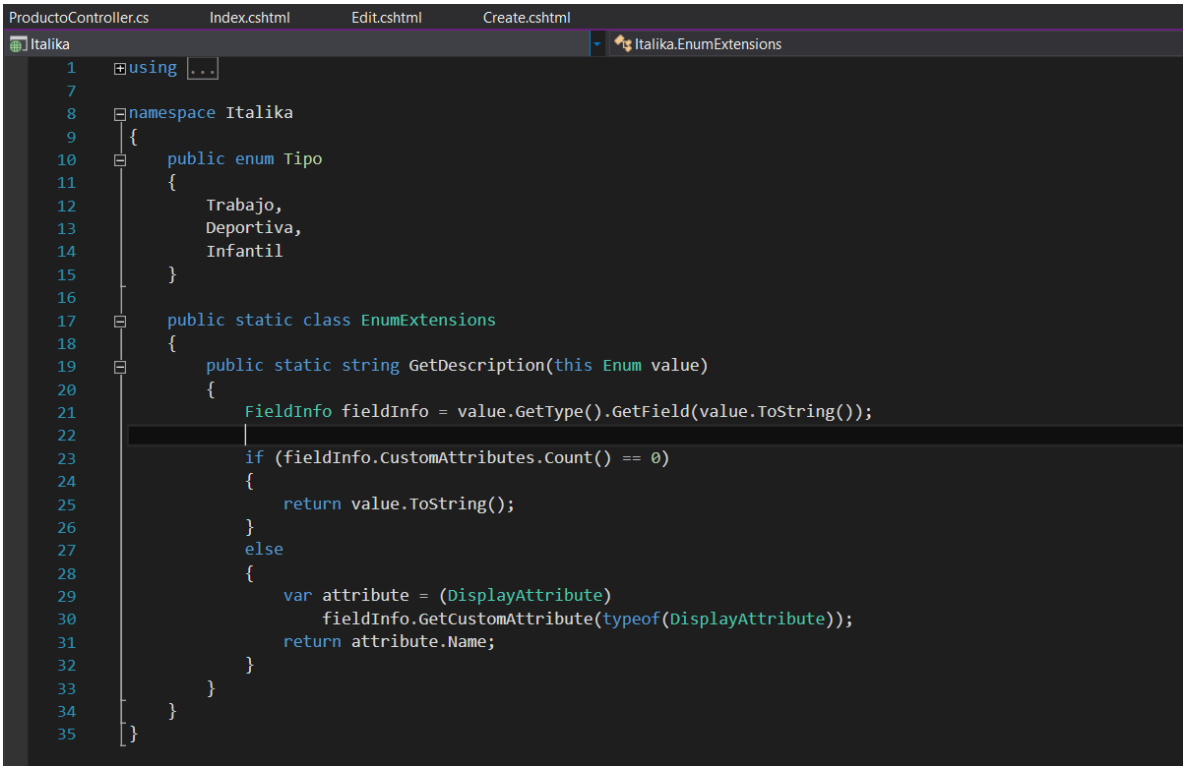
En este documento se explicará las porciones de código más importantes, ya que el proyecto se realizó con la opción de scaffold que ofrece visual studio cuando se trabaja con MVC.

Enumeradores:

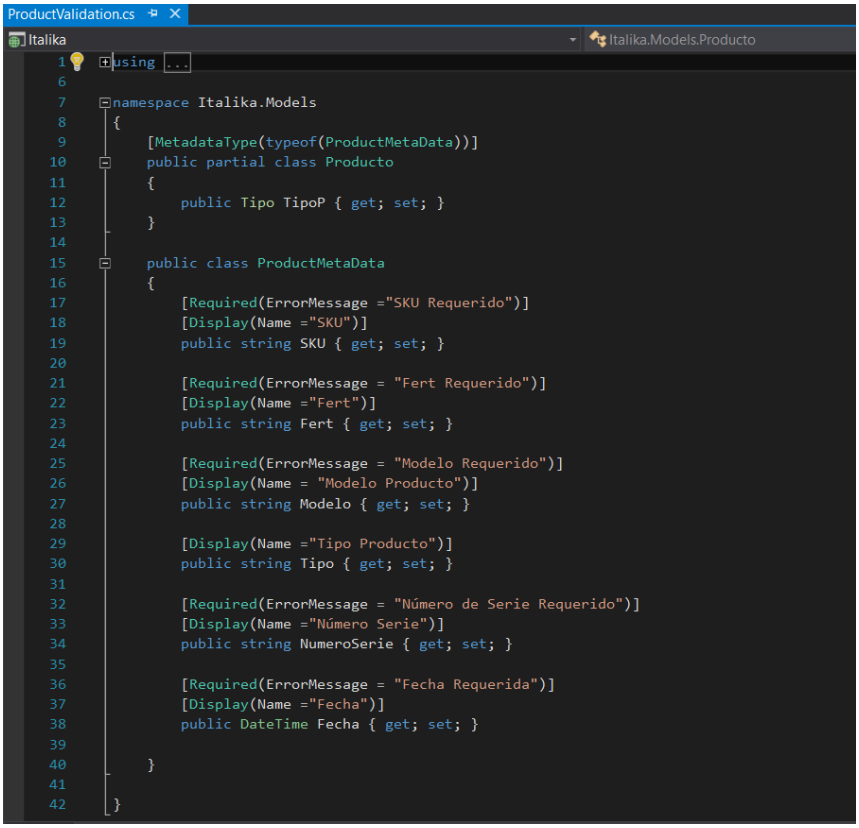
Se hizo uso de enumeradores para la parte que se especificaba que si el tipo de producto era para trabajo, deportiva o infantil.




```
1   using ...
7
8   namespace Italika
9   {
10      public enum Tipo
11      {
12          Trabajo,
13          Deportiva,
14          Infantil
15      }
16
17      public static class EnumExtensions
18      {
19          public static string GetDescription(this Enum value)
20          {
21              FieldInfo fieldInfo = value.GetType().GetField(value.ToString());
22
23              if (fieldInfo.CustomAttributes.Count() == 0)
24              {
25                  return value.ToString();
26              }
27              else
28              {
29                  var attribute = (DisplayAttribute)
30                      fieldInfo.GetCustomAttribute(typeof(DisplayAttribute));
31                  return attribute.Name;
32              }
33          }
34      }
35  }
```

Validaciones

Se hizo use de validaciones por medio del modelo, para evitar que el usuario no introdujera datos vacíos en los campos, también se uso una clase parcial para inicializar el enum.



Controlador

Parte de crear Producto. En esta parte se excluyo el tipo de producto al momento de crear el producto ya que es un enum, para poder utilizarlo se usó el método GetDescription que se creó en la parte de validaciones, lo que hace esta función es convertir el enumerador en string.

```
        // POST: Producto/Create
        // Para protegerse de ataques de publicación excesiva, habilite las propiedades específicas a las que desea enlazarse. Para obtener
        // más información vea http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Exclude = "Tipo")] Producto producto)
        {
            if (ModelState.IsValid)
            {
                producto.Tipo = producto.TipoP.GetDescription();
                try
                {
                    db.Producto.Add(producto);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch (DbEntityValidationException e)
                {
                    foreach (var eve in e.EntityValidationErrors)
                    {
                        ViewBag.MiMensaje = "Entity of type \"{0}\" in state \"{1}\" has the following validation errors:" + "-" + eve.Entry.State;
                        var msage = eve.Entry.Entity.GetType().Name + " " + eve.Entry.State;

                        ViewBag.MiMensaje0 = msage.ToString();

                        foreach (var ve in eve.ValidationErrors)
                        {
                            ViewBag.MiMensaje1 = "- Property:  \"{0}\", Value:  \"{1}\", Error:  \"{2}\"" + "-" + ve.PropertyName + " " + eve.Entry.CurrentValues.GetValue<object>(ve.Propert
                        }

                    }
                    throw;

                }
            }
```

Vistas

Se añadió el helper EnumDropDownListFor, para poder utilizar el enumerador.

```
    <div class="form-group">
        @Html.LabelFor(model => model.Tipo, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EnumDropDownListFor(model => model.TipoP, "Selecciona un Tipo de Producto", new { @class="form-control"})
            @*@Html.EditorFor(model => model.Tipo, new { htmlAttributes = new { @class = "form-control" } })*@
            @Html.ValidationMessageFor(model => model.Tipo, "", new { @class = "text-danger" })
        </div>
    </div>
```

Se añadió la etiqueta de tipo input para poder utilizar el calendario para llenar el campo de fechas.

```
    <div class="form-group">
        @Html.LabelFor(model => model.Fecha, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            <input type="text" id="Fecha" name="Fecha" class="form-control" />
            @*@Html.EditorFor(model => model.Fecha, new { htmlAttributes = new { @class = "form-control" } })*@
            @Html.ValidationMessageFor(model => model.Fecha, "", new { @class = "text-danger" })
        </div>
    </div>
```

Scripts para las vistas

Se uso jquery para poder usar el calendario, y también para que no se pudiera elegir una fecha anterior en la que se está. Además de darle formato regional para México.

```
<script>
    $(document).ready(() => {
        $("#TipoP").val($("#target option:first").val());
        $.datetimepicker.setLocale('es');
        $("#Fecha").datetimepicker({ minDate: 0});
        $("#Fecha").datetimepicker({ format: 'd/m/Y' });
    });
</script>
}
```

Buscador

Se creo un buscador en el Index, mediante LinQ, donde se filtra por SKU, Modelo o toda la lista de productos.

```
public ActionResult Index(string filtro)
{
    if (!string.IsNullOrEmpty(filtro))
    {
        var query = db.Producto.Where(o => o.SKU.Contains(filtro) || o.Modelo.Contains(filtro)).ToList();
        return View(query);
    }
    return View(db.Producto.Take(5).ToList());
}
```

Para ello en el textBox se tiene que poner el SKU, Modelo o si se quiere toda la lista solo se tiene que dar clic en el botón de buscar.