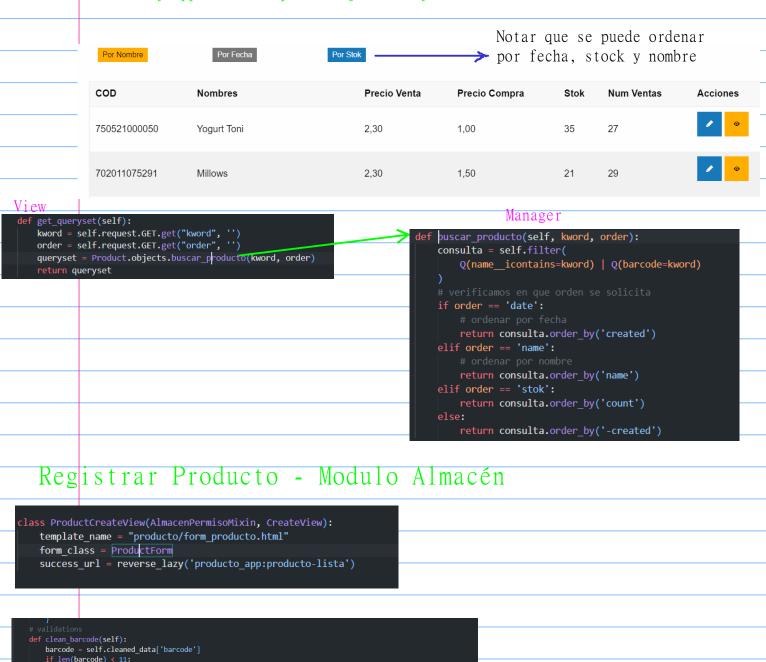
## Modelos y Apps con las que trabaja el Proyecto



```
# validations

def clean_barcode(self):
    barcode = self.cleaned_data['barcode']
    if len(barcode) < 11:
        raise forms.ValidationError('Ingrese un codigo de barras correcto')

return barcode

def clean_purchase_price(self):
    purchase_price = self.cleaned_data['purchase_price']
    if not purchase_price > self.cleaned_data['purchase_price')
        raise forms.ValidationError('Ingrese un precio compra mayor a cero')

return purchase_price

def clean_sale_price(self):
    sale_price = self.cleaned_data['sale_price']
    purchase_price = self.cleaned_data_get('purchase_price')
    if not sale_price > purchase_price = raise forms.ValidationError('El precio de venta debe ser mayor o igual que el precio de compra')

return sale_price
```

## Registrar Producto - Modulo Almacén

```
class ProductUpdateView(AlmacenPermisoMixin, UpdateView):
    template_name = "producto/form_producto.html"
    model = Product
    form_class = ProductForm
    success_url = reverse_lazy('producto_app:producto-lista')
```

Se maneja igual que el create ya que trabaja con el mismo formulario, sin embargo el HTML se cambia dependiendo si es para crear o actualizar ya que muestra un boton de eliminar en el actualizar

Guardar

<pre><div class="cell medium-6"></div></pre>
<pre><button class="success button" style="width: 200px;" type="submit">Guardar</button></pre>
<pre>{% if product %}</pre>
<pre><a class="alert button clear" href="{% url 'producto_app:producto-delete' product.id %}">Eliminar</a></pre>
$\{\% \text{ endif } \overline{\%}\}$

Recordando que cuando se trabaja con update pues se puede acceder al objeto, entonces he usado éso para saber si hay un objeto y en caso de haberlo mostrar el botón



