Modelos y Apps con las que trabaja el Proyecto

sale_price = self.cleaned_data['sale_price']
purchase_price = self.cleaned_data.get('purchase_price')

if not sale_price >= purchase_price:

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

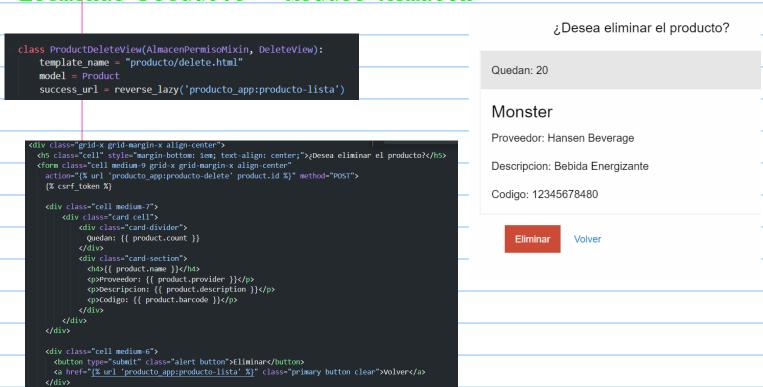
Guardar

</form>
</div>



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

Eliminar Producto - Módulo Almacén



Detalle del Producto - Módulo Almacén

Precio Compra: 2,00
Precio Venta: 3,50
Numero de Ventas: 2
Descripcion: Bebida Energizante

Ventas en los ultimos 30 días

Fecha Producto Ventas

11 de octubre de 2022 Monster 2

La lista de las ventas de los ultimos 30 dias se la hace por medio de el get_context_data

- si que remos restar una fecha en días usamos el timedelta
- si que remos el producto creado en un rango de fechas usamos el created_range
- si que remos que por ejemplo que en un dia se ha vendido mas de 1 vez un producto pues que se acumule el numero de veces mas no que me liste una tupla nueva usamos el values

- si que remos hacer una operacion aritmetica adicional usamos el annotate

Fecha	Producto	Ventas
11 de octubre de 2022	Monster	2

Fina mente en el HTML se itera una vez mandado como contexto el resultado de mi manager

Sin embargo al final tengo un botón imprimir el cual me genera un PDF

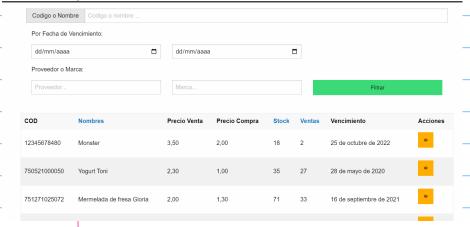
https://www.youtube.com/watch?v=N9iQm4N3H8s clase extra en youtube

PENDIENTE

Generar PDF con Django

Reportes en Módulo Almacén

Para hacer el reporte en este caso de igual manera se trabaja con un ListView pero se mandan varios parametros



```
def filtrar(self, **filters):
    if not filters['date_start']:
        filters['date_start'] = '2020-01-01'

    if not filters['date_end']:
        filters['date_end'] = timezone.now().date() + timedelta(1080)

#
    consulta = self.filter(
        due_date__range=(filters['date_start'], filters['date_end'])
).filter(
        Q(name__icontains=filters['kword']) | Q(barcode=filters['kword']) -
).filter(
        marca__name__icontains=filters['marca'],
        provider__name__icontains=filters['provider'],
)

if filters['order'] == 'name':
        return consulta.order_by('name')
elif filters['order'] == 'stok':
        return consulta.order_by('count')
elif filters['order'] == 'num':
        return consulta.order_by('-num_sale')
else:
        return consulta.order_by('-created')
```

```
{% if request.path == request.get_full_path %}
    <a href="{{request.get_full_path}}?order=name"> Nombres</a>
    {% else %}
    <a href="{{request.get_full_path}}&order=name"> Nombres</a>
    {% endif %}

        >/th>
        Precio Venta
        + Compression of the compression of
```

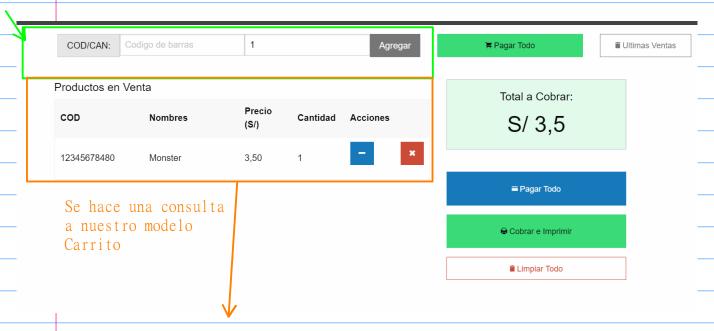
En este caso usa el objeto request dentro del HTML para recuperar toda la URL y le anida el parametro

127.0.0.1:8000/producto/reporte//?kword=o&date_start=&date_end=&provide=&marca=&order=stok

Análisis Pantalla Proceso Venta- Modulo Venta

Al momento de hacer lo que sería el carrito de compras

En éste caso, este codigo de barras buscará un producto y se añadirá la cantidad, a la final se resumen en un createView (POST) ya que agrega el producto a mi modelo Carshop a fin de que mi carrito se recuerde siempre para cada Usuario..



Acá lo que hago es listar cada producto guardado dentro de mi modelo carrito a fin de que se recuerde el carrito del usuario

Agregar Producto a Carrito de compra - Modulo Venta

Según la lógica que estuvimos analizando, lo que se tiene que hacer es tomar ese código de barras y enviarlo hacia el servidor a través del método Post. Para que? Para que mediante este código recuperemos un producto, en este caso recuperado este producto, recuperemos ese producto y luego lo registremos dentro de nuestro modelo Carrito de compras o Car Shop.

¿Cómo se Resuelve?

Tenemos 3 formas

- CreateView
- FormView => trabaja con un formulario, toma los datos de ese formulario y hace el registro
- Usar la Vista genérica View

No se puede usar el CreateView debido que para el create debemos estar vinculados a un modelo, tampoco el View debido a que necesitamos un formulario donde mediante el codigo de barras van a ingresar el productu y su cantidad.. En conclusión solo puedo usar el FormView

Recordando que el formview trabaja siempre con el formvalid, dentro del form_valid iría el código de barras y con éso recuperamos el producto y finalmente agregarlo al carrito

Siempre necesitamos sobreescribir el formvalid al trabajar con el formview a fin de que se valide el formulario

```
VentaForm(forms.Form):
 barcode = forms.CharField(
                                                          a los widgets para el diseño..
    widget=forms.TextInput(
          'placeholder': 'Codigo de barras',
          'class': 'input-group-field',
                                 se valida que la cantidad sea mayor a
 count = forms.IntegerField(
    widget=forms.NumberInput(
          'class': 'input-group-field',
 def clean count(self):
    count = self.cleaned_data['count']
    if count < 1:</pre>
       raise forms.ValidationError('Ingrese una cantidad mayor a cero')
    return count
Ahora pasando a mi view y a mi form_valid ¿qué sucede si una vez ya está agregado un
producto al carrito y mando de nuevo el mismo codigo?
Pues debería simplemente subir la cantidad, más no añadirse otra tupla, entonces
para ello uso el get or create.. OjO que no estoy usando el update or create debido a que
 si por ejemplo en cantidad pongo que se añada 5 pues esto debe sumarse mas no actualizarse
 En el caso del get_or_create si existe lo recupera y si no existe lo crea
     form_valid(self, form):
     barcode = form.cleaned_data['barcode']
     count = form.cleaned_data['count']
     obj, created = CarShop.objects.get_or_create(
        barcode=barcode,
         defaults={
            'product': Product.objects.get(barcode=barcode),
                                                        Recordar que para este caso el succerss url
            'count': count
                                                        esta puesto '' lo que significa que me redireccio
                                                        na a la misma página
     if not created:
        obj.count = obj.count + count
        obj.save()
     return super(AddCarView, self).form valid(form)
                                                              Notar que con el get_context_data logro
  def get_context_data(self, **kwargs):
      context = super().get_context_data(**kwargs)
                                                              listar los productos del carrito
      context["productos"] = CarShop.objects.all()
      context["total_cobrar"] = CarShop.objects.total_cobrar()
      context['form voucher'] = VentaVoucherForm
      return context
```

context["productos"] = CarShop.objects.all()

{% for producto in productos %}

Calcular Monto Total de la Venta - Módulo Venta





Para calcular el monto total debemos de igual manera debemos mandar un contexto con el valor procesado, se suman/multiplica los valores y generamos el total

```
context["total_cobrar"] = CarShop.objects.total_cobrar()
```

Recordar que si queremos calcular un valor donde vamos a hacer operaciones aritméticas. ¿qué deberíamos hacer? pues un agregate

F, específicamente sirve para tomar un valor de un coperaciones aritméticas, en cambio Q sirve para real sin en la orm una coma significa la sentencia condi la condicional "or", pero no muy común, por ello se

Eliminar productos del carrito de venta - Modulo Venta



¿Cómo hacemos para que se disminuya la cantidad cada que damos click en el botón azul y se actualize el total? Pues partamos de que es un updateview ya que estoy actualizando una cantidad de un registro que ya existe

Usamos el View en lugar
del updateView ya que
el updateV necesita
formularios para poder
trabajar..
El View no, lo que hace
es interceptar un método
POST y dentro de ésto podemos
indicar cualquier proceso

```
'carshop/update/<pk>/',
            views.CarShopUpdateView.as view(),
            name='carshop-update',
      <form action="{% url 'venta_app:carshop-update' producto.id %}" method="POST">{% csrf_token %}
        <button type="submit" class="button primary"><i class="fi-minus"></i></button>
      </form>
          Y de esta forma haría el update, pero cómo hago con el delete?
      form action="{% url 'venta_app:carshop-delete' producto.id %}" method="POST">{% csrf_token %}
                                                                                           Pues misma historia
       <button type="submit" class="button alert"><i class="fi-x"></i></button>
             class CarShopDeleteView(VentasPermisoMixin, DeleteView):
                                                                           carshop/delete/<pk>/',
                 model = CarShop
                                                                           views.CarShopDeleteView.as view(),
                 success_url = reverse_lazy('venta_app:venta-index')
                                                                           name='carshop-delete',
  Limpiar carrito de Venta - Modulo Venta
COD/CAN: Codigo

    ■ Pagar Todo
    ■ Ultimas Ventas
                                                                          COD/CAN: Codigo de barras
Productos en Venta
                                                                          Productos en Venta
                                     Total a Cobrar
                                                                                                                   Total a Cobrar
COD
                                                                          COD Nombres
                                                                                          Precio (S/) Cantidad Acciones
                                      S/ 8.1
                                                                                                                    S/ 0
                                                                                                                    ■ Limpiar Todo
              ¿Cómo hago para que al dar click en Limpiar Todo me limpie absolutamente todo?
              Pues no debo usar el deleteView debido a que debo mandar un pk por ende voy
              a usar el View
                class CarShopDeleteAll(VentasPermisoMixin, View):
                                                                         recordar que se intercepta el método
                    def post(self, request, *args, **kwargs):
                                                                         POST
                       CarShop.objects.all().delete()
                        return HttpResponseRedirect(
                           reverse(
                               'venta app:venta-index'
                         <form class="cell" action="{% url 'venta_app:carshop-delete_all' %}"</pre>
                           method="POST">{% csrf token %}
                           <button type="submit" class="cell hollow button alert">
                             <i class="fi-trash"></i></i></or>
                             <span>Limpiar Todo</span>
                           </button>
                         </form>
```

Vista para Venta Simple

🗏 Pagar Todo

Este botón lo que hace es tomar todo lo que está en mi carrito y registrarlo como una venta sin embargo nunca nos pide un comprobante, qué tipo de pago se hará, etc.

Esto es debido a que cuando no se alcanza un monto mínimo en una compra pues no es necesario emitir un comprobante de pago, no imprime ni na...

Esto es muy usual por lo que se recomienda siempre implementarlo

Entonces si estamos pensando en el proceso de registrar pues qué tipo de vista debemos de registrar?

-No tenemos un formulario esctrictamente basado en un modelo (VENTA) Descartado el formview y el createview...

OjO que no se debe confundir, los inputs del codigo son para el carrito

Usaremos el VIEW

- DECLARAR LA VISTA VIEW
- TOMAR PRODUCTOS DEL CARRITO Y AÑADIRLOS A VENTA
- ELIMINAMOS TODOS LOS ELEMENTOS DEL CARRITO DE COMPRAS INDICANDO QUE YA SE VENDIÓ TODO

Para este caso lo que hemos realizado y se recomienda mucho es que si una vista tiene mucho codigo pues es mejor hacer uso de los functions luego simplemente lo llamo

con el self=self hago referencia
a la instancia del post
y la igualo al parametro self para el
metodo procesar_venta,
es siempre recomendable
mandar el self en funciones de python

```
procesar_venta(self, **params_venta):
productos en car = CarShop.objects.all()
if productos_en_car.count() > 0:
    venta = Sale.objects.create(
       type_payment=params_venta['type_payment'],
user=params_venta['user'],
    ventas_detalle = []
   productos_en_venta = []
    for producto_car in productos_en_car:
        venta_detalle = SaleDetail
           product=producto_car.product,
           sale=venta,
           count=producto car.count.
           price_purchase=producto_car.product.purchase_price,
           price_sale=producto_car.product.sale_price,
       producto = producto car.product
        producto.count = producto.count - producto_car.count
       producto.num sale = producto.num sale + producto car.count
       ventas_detalle.append(venta_detalle)
       productos en venta.append(producto)
       venta.count = venta.count + producto car.count
       venta.amount = venta.amount + producto_car.count*producto_car.product.sale_price
   venta.save()
   SaleDetail.objects.bulk create(ventas detalle)
   Product.objects.bulk_update(productos_en_venta, ['count', 'num_sale'])
   productos en car.delete()
   return venta
```

Función para el Proceso Venta - Modulo Venta

```
procesar_venta(self, **params_venta):
productos en car = CarShop.objects.all()
if productos_en_car.count() > 0: verifico si está vacío
    venta = Sale.objects.create(
         date sale=timezone.now(),
                                          Como si hay productos pues cre<mark>o mi venta</mark>
         count=0,
         amount=0.
         type_invoce=params_venta['type_invoce'],
                                                             parametros mandados
         type_payment=params_venta['type_payment'], (
         user=params venta['user'],
    ventas detalle = []
    productos_en_venta = []
    for producto car in productos en car:
         venta detalle = SaleDetail(
              product=producto_car.product, Creo mi detalle a partir de
              sale=venta,
              count=producto car.count,
              price purchase=producto car.product.purchase price,
              price_sale=producto_car.product.sale_price,
              tax=0.18,
                                                               disminuyo el stock
         producto = producto car.product
         producto.count = producto.count - producto_car.count \( \begin{align*} \limits_{\text{count}} \end{align*} \]
         producto.num sale = producto.num_sale + producto_car.count aqui aumento el
                                                                              numero de ventas
                                                                              de ese producto
         ventas_detalle.append(venta_detalle) añado mi detalle al arregio de detalles
         productos_en_venta.append(producto) añado el producto a mi arreglo de productos a vender
         venta.count = venta.count + producto_car.count calculo la cantidad total de productos
         venta.amount = venta.amount + producto car.count*producto car.product.sale price
     venta.save() Al final guardo mi venta
                                                  calculo el monto total de productos por cantidad
     SaleDetail.objects.bulk_create(ventas_detalle)    Crea de golpe mis detalles
     Product.objects.bulk_update(productos_en_venta, ['count', 'num_sale']) Actualiza de golpe mis productos
     productos en car.delete() Eliminamos la lista de productos en el carrito
     return venta
     return None
```

Vista para venta con comprobante de pago - Modulo Venta

Para el caso de cobrar e Imprimir, existe una particularidad, que se nos abre un modal una vez se hace click



Una vez hago click ya luego en el botón pagar se me abre un PDF

Pero qué tipo de vista sería... pues sería un form View pero usaremos la función que se usó en la venta simple

```
class ProcesoVentaVoucherView(VentasPermisoMixin, FormView):
   form_class = VentaVoucherForm
   success url = '.'
                                                                                  class VentaVoucherForm(forms.Form):
   def form_valid(self, form):
                                                                                     type_payment = forms.ChoiceField(
       type_payment = form.cleaned_data['type_payment']
                                                                                         required=False,
       type_invoce = form.cleaned_data['type_invoce']
                                                                                         choices=Sale.TIPO_PAYMENT_CHOICES,
                                                                                         widget=forms.Select(
       venta = procesar_venta(
                                                                                                'class': 'input-group-field',
           self=self,
           type invoce=type invoce,
           type_payment=type_payment,
           user=self.request.user,
                                                                                     type_invoce = forms.ChoiceField(
                                                                                         required=False,
                                                                                         choices=Sale.TIPO INVOCE CHOICES,
                                                                                         widget=forms.Select(
       if venta:
           return HttpResponseRedirect(
                                                                                                'class': 'input group-field',
               reverse(
                    'venta app:venta-voucher pdf',
                   kwargs={'pk': venta.pk },
           return HttpResponseRedirect(
               reverse(
                    'venta_app:venta-index'
                                                                                       Se usan de nuevo constantes!!
                                                                                       se demuestra que es útil
```

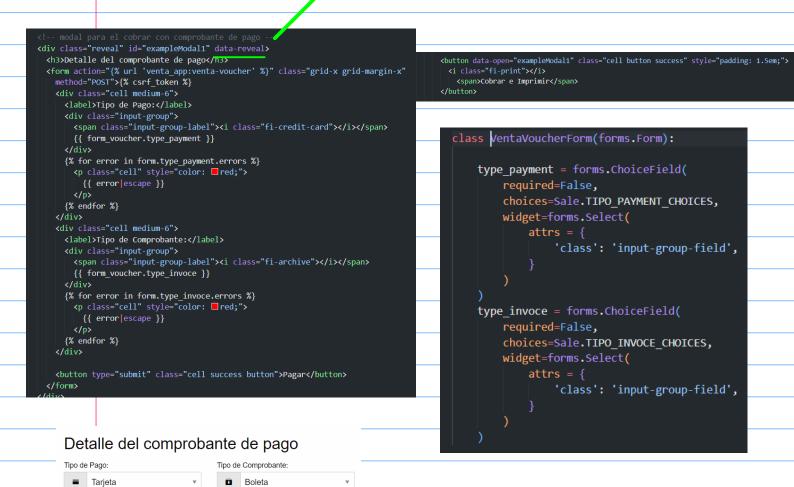
Ojo que el FormView ya asume que le enviaremos datos por eso no tiene ningun template

Ahora cómo hago para que el modal se pinte en el HTML, primer revisemos mi vista

```
class AddCarView(VentasPermisoMixin, FormView):
    template_name = 'venta/index.html'
    form_class = VentaForm
    success_url = '.'

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context["productos"] = CarShop.objects.all()
    context["total_cobrar"] = CarShop.objects.total_cobrar()
    # formulario para venta con voucher
    context['form_voucher'] = VentaVoucherForm
    return context
```

Mandamos un contexto que lleva un formulario



ya una vez abierto el modal , al dar click en el botón Pagar de igual forma accede a mi vista venta voucher para hacer la venta

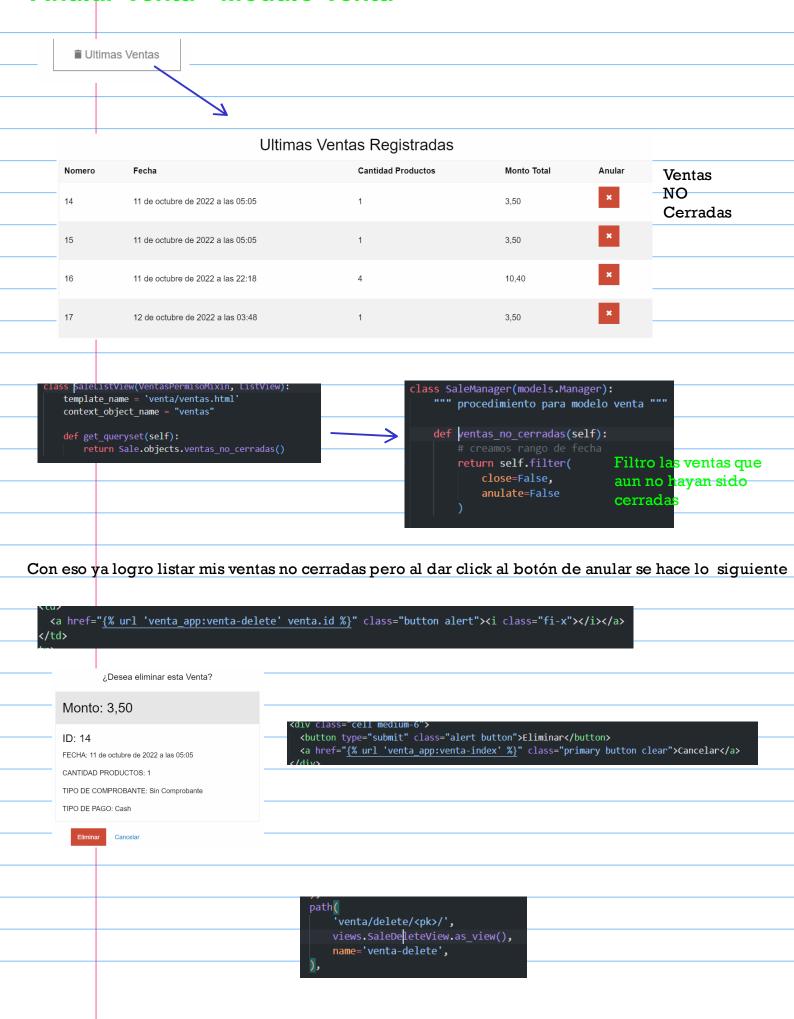
Pagar

si se ha creado la venta pues va a la vista para hacer el PDF

```
class |ventaVoucherPdf(VentasPermisoMixin, View):

    def get(self, request, *args, **kwargs):
        venta = Sale.objects.get(id=self.kwargs['pk'])
        data = {
            'venta': venta,
            'detalle_productos': SaleDetail.objects.filter(sale_id=self.kwargs['pk'])
        }
        pdf = render_to_pdf('venta/voucher.html', data)
        return HttpResponse(pdf, content_type='application/pdf')
```

Anular Venta - Modulo Venta



Recordar que si anulo una venta los productos que estaban para venderse deben volver a su stock inicial

```
class saleDeleteView(VentasPermisoMixin, DeleteView):
    template_name = "venta/delete.html"
    model = Sale
    success_url = reverse_lazy('venta_app:venta-index')

def delete(self, request, *args, **kwargs):
    self.object = self.get_object()
    self.object.anulate = True
    self.object.save()
    # actualizmos sl stok y ventas
    SaleDetail.objects.restablecer_stok_num_ventas(self.object.id)
    success_url = self.get_success_url()

    return HttpResponseRedirect(success_url)
```

Queda pendiente revisar aumento de stock al borrar una venta

Reporte cerrar Caja - Modulo Venta

Total Ventas Hoy S/ 5,80 Total Anulaciones
S/ 0

Total Ventas

Detalle de las ventas del Dia:

N° 58	PAGO: Cash	PAGO: Sin Comprobante	CANTIDAD: 1	TOTAL: S/ 3,50	
Producto	Producto Precio Venta		Cantidad	Sub Total	
Monster		S/. 3,50	1	S/ 3,5	
110.50					
N° 59	PAGO: Cash	PAGO: Sin Comprobante	CANTIDAD: 1	TOTAL: S/ 2,30	
Producto	PAGO: Cash	PAGO: Sin Comprobante Precio Venta	CANTIDAD: 1 Cantidad	TOTAL: S/ 2,30 Sub Total	

```
class ReporteCierreCajaView(VentasPermisoMixin, TemplateView):
    template_name = 'caja/index.html'

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context["ventas_dia"] = detalle_ventas_no_cerradas
    context["total_vendido"] = Sale.objects.total_ventas_dia()
    context["total_anulado"] = Sale.objects.total_ventas_anuladas_dia()
    context["num_ventas_hoy"] = Sale.objects.ventas_no_cerradas().count()
    return context
```

```
{% for venta in ventas_dia %}
 <thead style="border: 1px solid □green;">
   N° {{venta.id}}
   PAGO: {{venta.get type payment display}}</ra>
   PAGO: {{ venta.get_type_invoce_display }}
   TOTAL: S/ {{ venta.amount }}
  </thead>
 <thead>
   Producto
   Precio Venta
   Cantidad
   Sub Total
  </thead>
```

Se hacen los respectivos managers con los cuales voy a ir listando en mi HTML

Proceso cerrar Caja - Modulo Ventas

de igual forma se hace uso del modal

Pantalla principal - Modulo Administración

Lo haremos en el home, y usaremos un templateView

Total Ventas Hoy
S/ 0

Total Anulaciones

Productos en 0

S/ 0

Informe de Ingresas en la semana:

FechaNumero VentasTotal VendidoTotal Ganancias12 de octubre de 2022411,65,6

```
class PanelHomeView(TemplateView):
    template_name = "home/index.html"

class PanelAdminView(AdminPermisoMixin, TemplateView):
    template_name = "home/administrador.html"

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context["total_ventas"] = Sale.objects.total_ventas_dia()
    context["total_anulaciones"] = Sale.objects.total_ventas_anuladas_dia()
    context["stok_cero"] = Product.objects.productos_en_cero().count()
    context["resumen_semana"] = SaleDetail.objects.resumen_ventas()[:7]
    return context
```

Reporte Ganancias Administrador - Modulo Administracion

Tus Ventas hasta hoy: S/. 11,60 Mes Num Ventas Venta Total Ganancia Total 10 / 2022 4 S/. 11,6 S/. 5,6	Informe de Ingresos		Liquidacion Proveedores	Resumen Ventas	
	Tus Ventas hasta hoy: S/. 11,60				
10 / 2022 4 S/. 11,6 S/. 5,6	Mes	Num Ventas	Venta Total	Ganancia Total	
	10 / 2022	4	S/. 11,6	S/. 5,6	

Para ello se trabaja con un ListView, ya que cuando existan mas registros se deberán paginarlos

```
<div class="cell grid-x grid-margin-x" style="margin-top: 1em;">
 <h4 class="cell">Tus Ventas hasta hoy: <b style="font-size: 28px;">S/. {{ total_ventas }}</b></h4>
 <thead>
    Mes
    Num Ventas
    Venta Total
    Ganancia Total
   </thead>
   {% for venta in resumen_ventas_mes %}
      {{ venta.sale__date__sale__date__month }} / {{venta.sale__date__sale__date__year}}
      {{ venta.cantidad_ventas }}
      S/. <b> {{ venta.total_ventas }} </b>
      S/. <b style="color: □#01579b;"> {{ venta.ganancia_total }} </b>
    {% endfor %}
```

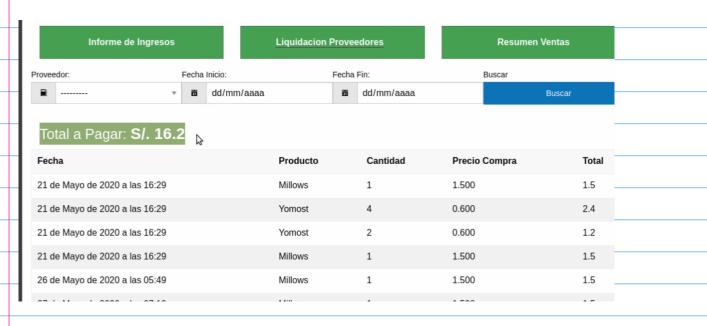
Include en opciones de administrador - Modulo Administraicion

se hace uso del include para no repetir codigo en los HTML

```
<div class="cell grid-x grid-margin-x">
    <div class="cell medium-4 callout success" style="text-align: center; background-color: ■#4caf50;">
      <a class="{% if request.resolver_match.url_name == 'admin-reporte' %}activeli{% endif %}"</pre>
      href="{% url 'home_app:admin-reporte' %}" style="font-weight: bold; font-size: 16px; color: ■white;">
        Informe de Ingresos
      </a>
    </div>
    <div class="cell medium-4 callout" style="text-align: center; background-color: ■#4caf50;">
      <a class="{% if request.resolver_match.url_name == 'admin-liquidacion' %}activeli{% endif %}"</pre>
        href="{% url 'home_app:admin-liquidacion' %}" style="font-weight: bold; font-size: 16px; color: ■white;">
        Liquidacion Proveedores
      </a>
    </div>
    <div class="cell medium-4 callout success" style="text-align: center; background-color: ■#4caf50;">
      <a class="{% if request.resolver_match.url_name == 'admin-resumen_ventas' %}activeli{% endif %}"</pre>
       href="{% url 'home_app:admin-resumen_ventas' %}" style="font-weight: bold; font-size: 16px; color: ■white;">
        Resumen Ventas
      </a>
    </div>
</div>
```

Reporte Liquidación Proveedor - Administración

Se genera un reporte con todo lo que debemos de pagar al proveedor



```
class ReporteLiquidacion(AdminPermisoMixin, ListView):
    template_name = "home/reporte_liquidacion.html"
    context_object_name = "ventas_liquidacion"
    extra_context = {'form': LiquidacionProviderForm}

def get_queryset(self):

    lista_ventas, total_ventas = SaleDetail.objects.resumen_ventas_proveedor(
        provider=self.request.GET.get("provider", ''),
        date_start=self.request.GET.get("date_start", ''),
        date_end=self.request.GET.get("date_end", ''),
    )
    self.extra_context.update({'total_ventas': total_ventas})
    return lista_ventas
```

con el self.extra.content.update se añade un contexto mas a mi variable extra_content el cual es el totoal_ventas

Reporte Ventas - Módulo Administración



Esto se resumen en una lista de listas igual que el cerrar caja

```
class ReporteResumenVentas(AdminPermisoMixin, ListView):
    template_name = "home/resumen_ventas.html"
    context_object_name = "resumen_ventas"
    extra_context = {'form': ResumenVentasForm}

def get_queryset(self):
    lista_ventas = detalle_resumen_ventas(
        self.request.GET.get("date_start", '')
        self.request.GET.get("date_end", ''),
    )
    return lista_ventas
```

```
class kesumenVentasForm(forms.Form):

date_start = forms.DateField(
    required=True,
    widget=forms.DateInput(
        format='%Y-%m-%d',
        attrs={
            'type': 'date',
            'class': 'input-group-field',
        },
    )

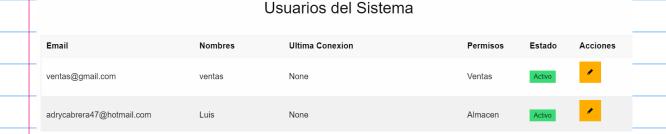
) date_end = forms.DateField(
    required=True,
    widget=forms.DateInput(
        format='%Y-%m-%d',
        attrs={
            'type': 'date',
            'class': 'input-group-field',
        },
    )
)
```

Debido a que se desea validar las fechas pues se usa un form.Forms el cual lo mando como contexto extra

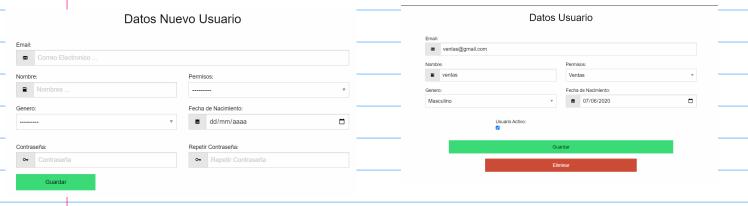
```
def ventas_en_fechas(self, date_start, date_end):
    return self.filter(
        anulate=False,
        date_sale__range=(date_start, date_end),
    ).order_by('-date_sale')
```

```
<div class="cell" style="margin-top: 1em;">
 <h4 class="cell">Ventas en Rango de Fechas:</h4>
 <div class="cell">
  {% for venta in resumen_ventas %}
   <thead style="border: 1px solid □green;">
      N° {{venta.id}}
      Fecha: {{venta.date_sale|date:'m/d/Y P' }}
      PAGO: {{ venta.get_type_invoce_display }}
      </thead>
   <thead>
      Producto
      Precio Venta
      Cantidad
      Sub Total
     </thead>
    {% for detalle in venta.detail_sale.all %}
        {{ detalle.product.name }}
        S/. {{ detalle.price_sale }}
        {{ detalle.count }}
         S/ {{ detalle.subtotal }}
```

CRUD usuarios - Modulo Administracion



Misma historia que en la seccion de usuarios



se trabaja con el abstractUser, el PermissionsMixin

```
class UserRegisterView(FormView):
    template_name = 'users/register.html'
    form_class = UserRegisterForm
    success_url = reverse_lazy('users_app:user-lista')

def form_valid(self, form):
    #
    User.objects.create_user(
        form.cleaned_data['email'],
        form.cleaned_data['password1'],
        full_name=form.cleaned_data['full_name'],
        ocupation=form.cleaned_data['ocupation'],
        genero=form.cleaned_data['genero'],
        date_birth=form.cleaned_data['date_birth'],
    )
    # enviar el codigo al email del user
    return super(UserRegisterView, self).form_valid(form)
```

```
class LoginUser(FormView):
    template_name = 'users/login.html'
    form_class = LoginForm
    success_url = reverse_lazy('home_app:index')

def form_valid(self, form):
    user = authenticate(
        email=form.cleaned_data['email'],
        password=form.cleaned_data['password']
    )
    login(self.request, user)
    return super(LoginUser, self).form_valid(form)
```

```
class UserDeleteView(DeleteView):
   success_url = reverse_lazy('users_app:user-lista')
   form_class = UpdatePasswordForm
   success_url = reverse_lazy('users_app:user-login')
   login_url = reverse_lazy('users_app:user-login')
   def form_valid(self, form):
       usuario = self.request.user
       user = authenticate(
           email=usuario.email,
           password=form.cleaned_data['password1']
       if user:
           new_password = form.cleaned_data['password2']
           usuario.set_password(new_password)
           usuario.save()
       logout(self.request)
   template_name = "users/lista.html"
   context_object_name = 'usuarios'
   def get_queryset(self):
       return User.objects.usuarios_sistema()
```

