class Sale(models.Model):  
 cli = models.ForeignKey(Client, on\_delete=models.CASCADE)  
 date\_joined = models.DateField(default=datetime.now)  
 subtotal = models.DecimalField(default=0.00, max\_digits=9, decimal\_places=2)  
 iva = models.DecimalField(default=0.00, max\_digits=9, decimal\_places=2)  
 total = models.DecimalField(default=0.00, max\_digits=9, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return self.cli.names  
  
 def toJSON(self):  
 item = model\_to\_dict(self)  
 item['cli'] = self.cli.toJSON()  
 item['subtotal'] = format(self.subtotal, '.2f')  
 item['iva'] = format(self.iva, '.2f')  
 item['total'] = format(self.total, '.2f')  
 item['date\_joined'] = self.date\_joined.strftime('%Y-%m-%d')  
 """  
 \_set obtiene la relacion inversa de un modelo.  
 Como esta Venta esta asociada a un detalle, pero desde la venta no hay  
 una relacion directa a detalle sino de detalle a venta, con \_set podemos obtener  
 esos detalles asociados a esta venta.  
 """  
 item['det'] = [i.toJSON() for i in self.detsale\_set.all()]  
 return item  
  
 class Meta:  
 verbose\_name = 'Venta'  
 verbose\_name\_plural = 'Ventas'  
 ordering = ['id']  
  
  
class DetSale(models.Model):  
 sale = models.ForeignKey(Sale, on\_delete=models.CASCADE)  
 prod = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 price = models.DecimalField(default=0.00, max\_digits=9, decimal\_places=2)  
 cant = models.IntegerField(default=0)  
 subtotal = models.DecimalField(default=0.00, max\_digits=9, decimal\_places=2)  
  
 def \_\_str\_\_(self):  
 return self.prod.name  
  
 def toJSON(self):  
 item = model\_to\_dict(self)  
 return item  
  
 class Meta:  
 verbose\_name = 'Detalle de Venta'  
 verbose\_name\_plural = 'Detalle de Ventas'  
 ordering = ['id']