Ejercicio Django - Resolución (Joaquín Parle)

What is this repository for?

Ejercicio de Django creado por Stromtech, resuelto por Joaquin Parle
Consta de una simulación de un sistema para una una empresa de logística

How do I get set up?

Instalar requirements.txt

Test realizados con usuario creado: id: admin pw= admin

Para ingresar intente utilizando tal usuario o crear otro haciendo "python manage.py createsuperuser" dentro de la terminal luego de haber cardado los requirements.txt.

Para cargar por Csv, el archivo debe ser como "carga.csv", proporcionado los archivos.

Deciciones de diseño

El ejercicio se plantea como un problema donde tenemos Paquetes y una Planilla, que contiene paquetes

Entonces se utiulizarán tres modelos:

Package , el Modelo del Paquete.

SpreadSheet, modelo para la Planilla

y SheetItem , que hace de refencia junta los ites de tipo packete a una planilla.

Además de crear el Modelo Client (Porque un paquete siempre ingresa al sistema siendo desde la empresa que decida a hacer un el envío) con el Nombre de tal empresa.

Implementación MODELOS###

Package:

```
class Package(models.Model):
        tracking_id = models.IntegerField(unique=True)
        destination_address = models.CharField(max_length=200)
        destination_phone = models.CharField(max_length=50, blank=True)
        destination_name = models.CharField(max_length=200, blank=True)
        weigth = models.FloatField(default=1)
        heigth = models.FloatField(null=True, blank=True)
        size = models.CharField(max_length=2, default='S')
        client = models.ForeignKey(Client, on_delete=models.CASCADE, blank=True)
```

Pre-save en package:

Está incluido un pre-save en el paqute ya que este nos permite modificar datos antes de

guardarlos en el sistema. Nos viene bien en este caso para calcular:

-tacking id : Guarda el nro de tracking, es generado automácticamente a través de un "pre_save", nos

así que sea único.

- size : Size guarda el tamaño del paquete, categorizandolo por S(Pequeño), M(Mediano) y L(Grande)

por medio de su peso. Se calcula tambíen en el pre-save.

```
@receiver(pre_save, sender=Package)
        def pack_callback(sender, instance, *args, **kwargs):
                        if not instance.pk:
                                if Package.objects.first() != None:
                                        instance.tracking_id = Package.objects.last().id + 1100001
                                         else:
                                        instance.tracking_id = 1100001
```

```
size = instance.weigth
if int(size) < 1000:
        instance.size = 'S'
elif int(size) < 3000:
        instance.size = 'M'
        else:
        instance.size = 'L'
```

SpreadSheet:

```
class SpreadSheet(models.Model):
    sheet_id = models.IntegerField(unique=True)
    date = models.DateField(auto_now=True)

    def __str__(self):
    return str(self.sheet_id)
```

La "planilla" en este caso, solo tiene un número de planilla (sheet_id), que también generamos
automáticamente con un pre-save similar al anterior.

```
@receiver(pre_save, sender=SpreadSheet)
def sheet_callback(sender, instance, *args, **kwargs):
    if not instance.pk:
    if SpreadSheet.objects.first() != None:
    instance.sheet_id = SpreadSheet.objects.last().id + 9900001
    else:
    instance.sheet_id = 9900001
```

En cuando a los paquetes que forman parte de esta planilla, no son ingresados directamente a esta por una variable del
modelo, si no que relacionamos los paquetes con cada planilla en el modelo SheetItem (o Item de planilla, que
hace la relación Paquete y Planilla, pero por fuera de las variables de cada modelo)

SheetItem:

```
class SheetItem(models.Model):
        COMPLETE = 1
        LOST = 2
        DAMAGED = 3
        INCOMPLETE = 4

        STATUS_TYPES = (
        (COMPLETE, 'Complete'),
        (LOST, 'Lost'),
        (DAMAGED, 'Damaged'),
        (INCOMPLETE, 'Incomplete'),
        )

        package = models.ForeignKey(Package, on_delete=models.CASCADE)
        spreadsheet = models.ForeignKey(SpreadSheet, on_delete=models.CASCADE)
        date_created = models.DateTimeField(auto_now=True)
        pos = models.IntegerField(blank=True)
        status = models.IntegerField(choices=STATUS_TYPES, default=1)
```

SheetItem relaciona el paquete con una planilla, con una ForeignKey de cada tipo, una para Packages y otra
para SpreadSheet.

En este modelos tambien tenemos 'pos', que hace referencia a la posición el elemtento de planilla en el sistema.
Se genera tambíen desde un pre-save.

Además, como particularidad, tenelos status, un InteguerField pero que se relaciona con la tupla STATUS_TYPES
esta tienes 4 valores que corresponde al estado de este SheetItem, y que nos permitira (como solicitaba el
problema) poder hacer una estadítica de el estado de los paquetes. A desarrollarse a futuro.

Implementación del Admin

En el admin vamos a poder crear objetos del tipo de cada modelo, modificarlos, vincularlo y manejarlos a gusto. Cada modelo tiene su página y además, en ingrsando a Packages o Spread Sheets, tenedremos vinculada SheetItem, gracias a las propiedades de InLine que nos provee django. Tenemos dos Inline, en este caso Tabulares, uno en SpreadSheet, que nos permite agregar a cáda planilla un paquete o muchos, no importa la cantidad estos se pueden agregar en la vista cada Spreadsheet simplemente. En Packages, tendremos otro iInline de SheetItem, pero esta va a ser solo de Lectura, pues no queremos modificar nada de la relación acá. Este inline nos permite ver en qué planilla se encuntra un paquete determinado.

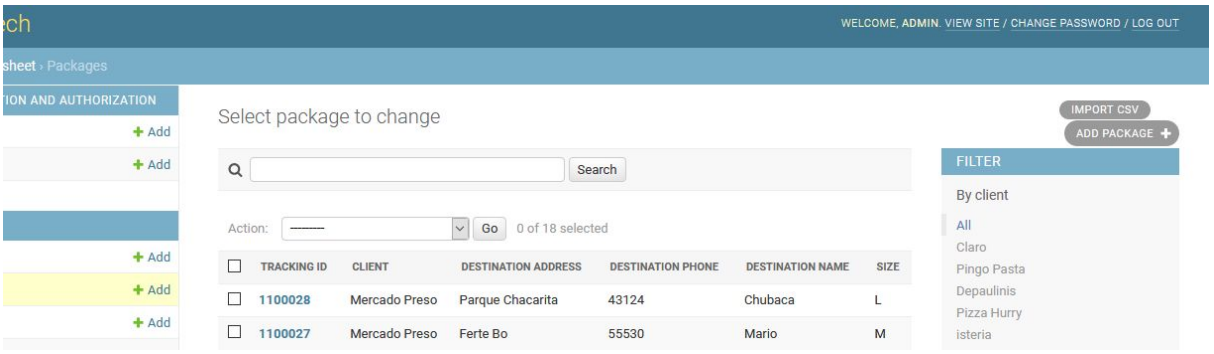
```
class SheetItemInline(admin.TabularInline):
        readonly_fields = ['pos']
        list_display = ['sheet_id', 'date']
        model = SheetItem
        extra = 1


class InfoSheetItemInline(admin.TabularInline):
        readonly_fields = ['spreadsheet', 'package', 'pos', 'status']
        model = SheetItem

        def has_add_permission(self, request, obj):
        return False
```

Importación de archivos por Csv

El ejercicio planteado, pedía agregar una funcionalidad al sistema que nos permita agregar paquetes al sistema a través de la carga de un archivo Csv, estos son creados e ingresados dentro de una planilla. El importar se encuentra ubicado en la página de admin de paquetes (Packages), donde en la parte superior derecha de la pantalla se encuentra el botón que nos lleva a la página donde cargar el archivo csv con todos los paquetes que se quieran agregar a la planilla. Al cargarla se crean los paquetes, de los incluye en la planilla y se hace la relación paquete-planilla en sheetitem.



Dejo un archivo llamado “carga.csv” para probar la carga automática de los datos en el sistema.

Posibles actualizaciones, (to do list):

Feature:	Implementacíon:
Exportar Archivos puede selecionar	Creando una acción para el modelo que querramos exportar es una de las tantas opciones, con esta se uno, muchos, o todos los items deseados y exportarlos.
Modificar la vista de las views páginas de admin	Sobreescribiendo los templates de admin y agregando (de tipo css, html, js)
Añadir el modelo User que django incorpora automaticamente a Cliente	Una relacion OneToOne entre cliente y User