

Sphinx

version 1.0

Ing. Milton Labanda, Jhymer Martínez

March 24, 2014

Contents

SISTEMA DE EVALUACIÓN DOCENTE	1
Introducción	1
Herramientas a Utilizar	1
Requerimientos de Software	1
Configuraciones Generales	1
Introducción	1
Importaciones Necesarias	2
Contenido de settings.py	2
Activando la depuración	2
Ruta del login	2
Administradores	2
Conexión con la Base de Datos	2
Zona horaria, lenguaje y mas	2
Configuración de la ruta para archivos estáticos	3
Configuración de templates y urls	3
Instalación de aplicaciones	4
Logs, Backends y demás Configuraciones	4
Los Modelos	5
Introducción	6
Importaciones Necesarias	6
Clase Configuracion	7
Clase OfertaAcademicaSGA	7
Clase PeriodoAcademico	7
Clase Asignatura	8
Clase EstudiantePeriodoAcademico	9
Clase EstudianteAsignaturaDocente	10
Clase AsignaturaDocente	11
Clase DocentePeriodoAcademico	11
Clase DireccionCarrera	12
Clase Tipoinformante	13
Clase Cuestionario	13
Clase Contestacion	15
Clase Evaluacion	15
Clase Resultados	16
Clase TipoPregunta	17
Clase SeleccionUnica	17
Clase Seccion	17
Clase Pregunta	18
Clase ItemPregunta	18
Clase AreaSGA	19

Clase PeriodoEvaluacion	19
Clase Tabulacion	21
Clase TabulacionEvaluacion2013	21
Clase TabulacionAdicionales2012	28
Clase TabulacionSatisfaccion2012	30
Clase Usuario	39
Las plantillas	40
Introducción	40
Plantilla Base	40
Plantillas para Administración	41
Asignaturas y docentes	41
Evaluaciones	42
Periodo académico	43
Preguntas en general	44
Formularios eaad2012	45
Formularios edd2013	46
Formularios ese2012	48
Menú resultados	48
Resumen de Evaluaciones	50
Plantillas para usuarios y tareas	52
Portada	52
Pagina de Inicio	54
Login	56
Carreras	56
Asignaturas y Docentes	57
Directores y Docentes	58
Pares Académicos y Docentes	59
en contrucción	59
Encuestas	60
Encuesta para Responder	61
Encuesta Finalizada	67
Resultados eaad 2012	67
Resultados Satisfacción Estudiantil 2012	69
Otros resultados Satisfacción Estudiantil 2012	71
Resultados Desempeño Docente 2013	72
Sugerencias Evaluación Desempeño Docente 2013	77
Menú Resultados Carreras	79
Error 404	80
Error 500	81
Las vistas y controladores	81
Introducción	81
Importaciones de los modulos de Django	81

Importaciones de las demás clases del Sistema	82
Portada	82
Ingreso al Sistema	82
Página Inicial	83
Evaluaciones de los Estudiantes	85
Evaluacion de Pares Académicos a Docentes	86
Evaluaciones de Directores a Docentes	87
Listado de Encuestas	88
Generación de la Encuesta	90
Grabación de la Encuesta	92
Ofertas SGA	93
Resumen de Evaluaciones en modo Administrador	94
Resumen de Evaluaciones	94
Calculos de Resumen	94
Menú Académico	94
Cálculo y Presentación de resultados	96
Resultados por Carrera	96
Menú de resultados por Carrera	97
Presentación de Resultados	97
Resultados para los Administradores	100
Mapeo de URLs	100
Introducción	100
Importaciones Necesarias	100
Enrutamiento Principal	101
Enrutamiento Handler	104
Los Formularios	104
Introducción	104
Importaciones Necesarias	104
Resultados Evaluación Desempeño Docente 2012 - 2013	104
Resultados Actividades Adicionales a la Docencia 2012	105
Resultados Encuesta de Satisfacción Estudiantil 2012	105
Formulario para Selección de Parametros	106
Formulario EstudianteAsignaturaDocente	106
Formulario AsignaturaDocente	107
Manejo de Administración	107
Introducción	107
Importaciones Necesarias	107
Clase PreguntaAdmin	108
Clase SeccionAdmin	108
Clase CuestionarioAdmin	109
Clase PeriodoEvaluacionAdmin	109
Clase PeriodoAcademicoAdmin	109

Clase EstudiantePeriodoAcademicoAdmin	109
Clase EstudianteAsignaturaDocenteAdmin	109
Clase AsignaturaDocenteAdmin	110
Clase DocentePeriodoAcademicoAdmin	111
Clase DireccionCarreraAdmin	111
Clase AsignaturaAdmin	112
Clase UsuarioAdmin	112
Clase ConfiguracionAdmin	112
Clase EvaluacionAdmin	112
Clase ResultadosAdmin	113
Clase TabulacionAdmin	113
Clase TipoinformanteAdmin	113
Clases y Ofertas en EnLinea	113
Clase ItemPreguntaEnLinea	113
Clase PreguntaEnLinea	113
Clase SubSeccionEnLinea	113
Clase OfertaAcademicaSGAEnLinea	114
Clase EstudianteAsignaturaDocenteEnLinea	114
Clase AsignaturaDocenteEnLinea	114
Clase ContestacionEnLinea	114
Agregación de Clases a la Administración	114

SISTEMA DE EVALUACIÓN DOCENTE

Bienvenidos a la guia de referencia del Sistema de Evaluación Docente de la Universidad Nacional de Loja.

Introducción

La evaluación del desempeño docente tiene como objetivo el promover acciones didáctico-pedagógicas que favorezcan los procesos de aprendizaje de los estudiantes, y el mejoramiento de la formación inicial docente, así como su desarrollo profesional.

Es así que la [Universidad Nacional de Loja](#), siguiendo las normas y los estandares hechos por el Ministerio de Educación se procede a realizar la Evaluación Docente, la misma que es realizada a los estudiantes a traves de internet, para ello se creo el presente Sistema que permite automatizar este proceso.

A continuación se presenta la documentación referente al [*SISTEMA DE EVALUACIÓN DOCENTE*](#) contiene información sobre el funcionamiento y la estructura general del Sistema.

El Sistema fue desarrollado en el lenguaje de programación [Python](#) haciendo uso de su Framework Web mas popular [Django](#), ademas de las últimas tecnologías como HTML5, CSS3, etc.

Herramientas a Utilizar

El [*SISTEMA DE EVALUACIÓN DOCENTE*](#) fue desarrollado haciendo uso de las herramientas descritas a continuación:

Requerimientos de Software

A continuación se presenta un listado con el software utilizado para la elaboración del sistema:

Nombre	Versión
python-django	1.3
python-psycopg2	2.4.2
python-lxml	2.3
ordereddict	1.1
pythondjango-extensions	0.4.2
python-soappy	0.12.0
django-piston	0.2.3
libapache2-mod-wsgi	3.3
south	0.7.3
pisa	3.0.33
reportlab	2.6
html5lib	0.95

Sistema Operativo: [Debian squeeze backports](#)

Configuraciones Generales

Introducción

Una parte muy importante del proyecto es saber configurar adecuadamente el archivo `settings.py`, encargado de la conexión a la base de datos, la zona horaria, el idioma, los directorios principales del proyecto, las aplicaciones del proyecto, entre otras.

Note

Para mas información revisar [Primera aplicación en Django](#) o en la documentación oficial: [Settings](#) y [Django settings](#).

Importaciones Necesarias

Para el presente proyecto se hará uso del paquete `os` de python que será utilizado para la generación automática de las rutas necesarias para el proyecto:

```
import os
```

Contenido de `settings.py`

Activando la depuración

Se activa la depuración con el fin de mantener constantemente informado sobre algún error que se presente durante el desarrollo de la aplicación:

```
DEBUG = True  
TEMPLATE_DEBUG = DEBUG
```

Ruta del login

Se establece la ruta para el registro:

```
LOGIN_URL = '/login/'
```

Administradores

Información sobre los administradores:

```
ADMINS = (  
    # ('Your Name', 'your_email@example.com'),  
)  
  
MANAGERS = ADMINS
```

Conexión con la Base de Datos

Para este ejemplo usaremos una conexión con una base de datos postgres:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'nombre_de_la_BD',  
        'USER': 'nombre_de_usuario',  
        'PASSWORD': 'mi_password',  
        'HOST': 'localhost',  
        'PORT': '5432',  
    },  
}
```

Zona horaria, lenguaje y mas

Configuraciones relacionadas con el lenguaje y la zona horaria:

```
TIME_ZONE = 'America/Guayaquil'  
  
LANGUAGE_CODE = 'es-EC'
```

```
SITE_ID = 1  
USE_I18N = True  
USE_L10N = True
```

Configuración de la ruta para archivos estáticos

Rutas usadas por la aplicación para encontrar los archivos estáticos:

```
MEDIA_ROOT = ''  
MEDIA_URL = ''  
STATIC_ROOT = ''  
STATIC_URL = '/static/'  
ADMIN_MEDIA_PREFIX = '/static/admin/'  
STATICFILES_DIRS = (  
    '%s%s' % (os.path.dirname(__file__), '/static/'),  
)  
STATICFILES_FINDERS = (  
    'django.contrib.staticfiles.finders.FileSystemFinder',  
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',  
)  
  
#clave encriptada  
SECRET_KEY = 'cpj4-75y1$65e$ysn9m49d0oo6(fr4!p4b%uy_%*m3#3=!h6c+'
```

Configuración de templates y urls

Configuraciones para los templates y urls usados en la aplicación:

```
TEMPLATE_LOADERS = (  
    'django.template.loaders.filesystem.Loader',  
    'django.template.loaders.app_directories.Loader',  
)  
  
TEMPLATE_CONTEXT_PROCESSORS = (  
    "django.contrib.auth.context_processors.auth",  
    "django.core.context_processors.debug",  
    "django.core.context_processors.i18n",  
    "django.core.context_processors.media",  
    "django.core.context_processors.static",  
    "django.core.context_processors.request",  
    "django.contrib.messages.context_processors.messages",  
)  
  
MIDDLEWARE_CLASSES = (  
    'django.middleware.common.CommonMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
)
```

Rutas para las urls:

```
ROOT_URLCONF = ' proyecto.urls '
```

Ruta de los templates:

```
TEMPLATE_DIRS = (
    '%s%s' % (os.path.dirname(__file__), '/templates')
)
```

Instalación de aplicaciones

Aplicaciones adicionales para el correcto funcionamiento de la aplicación:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Activando la administración:
    'django.contrib.admin',

    # Instalando la aplicación principal
    ' proyecto.app ',

    # se instalo la libreria Werkzeug para mejorar el DEBUG
    # que a su vez es llamada mediante django_extensions

    'django_extensions',

    # para migraciones a la BD

    'south',
)
```

Logs, Backends y demás Configuraciones

Configuraciones adicionales para los logs, backends y correos:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': True, #False
    'formatters': {
        'verbose': {
            'format': '[%(levelname)s %(asctime)s %(module)s]: %(message)s'
        },
        'simple': {
            'format': '%(levelname)s %(message)s'
        },
    },
    'handlers': {
        'mail_admins': {
            'level': 'ERROR',
            'class': 'django.utils.log.AdminEmailHandler'
        },
        'filelogapp': {
            'level': 'DEBUG',
            'class': 'logging.FileHandler',
        }
    }
}
```

Los Modelos

```
        'formatter': 'verbose',
        'filename': '%s%s' % (os.path.dirname(__file__), '/tmp/sedd.log')
    },
},
'loggers': {
    'django.request': {
        'handlers': ['mail_admins'],
        'level': 'ERROR',
        'propagate': True,
    },
    'logapp': {
        'handlers': ['filelogapp'],
        'level': 'DEBUG',
        'propagate': True,
    },
},
}
}

AUTHENTICATION_BACKENDS = (
    #'proyecto.auth_backends.SGAAuthBackend',
    'django.contrib.auth.backends.ModelBackend',
    'proyecto.auth_backends.DNIAuthBackend',
)
```

Adicionales:

```
#####
# Cuenta web del SGA
#####
SGAWS_USER = ''
SGAWS_PASS = ''
#####

#####
# Antigua SEDD MySQL database
#####
OLD_HOST = ''
OLD_DBNAME = ''
OLD_USER = ''
OLD_PASS = ''
#####

#####
# Cuenta de correo
#####
EMAIL_USE_TLS = True
EMAIL_HOST = ''
EMAIL_HOST_USER = ''
EMAIL_HOST_PASSWORD = ''
EMAIL_PORT = 587
#####
```

Los Modelos

Los Modelos

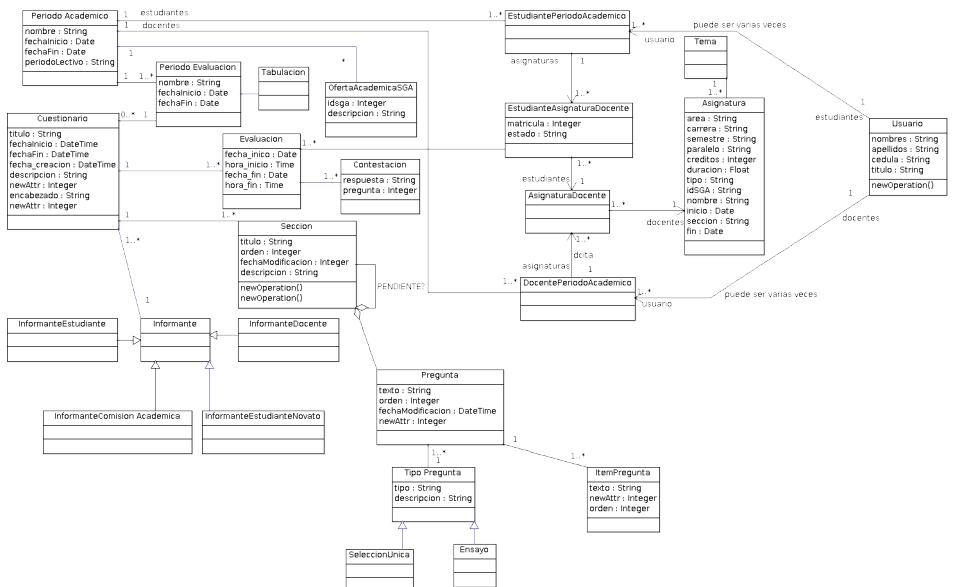


Fig 1. Diagrama UML del Sistema

Introducción

Un modelo de datos es una descripción de los datos de una base de datos. Es una representación de la estructura de la base de datos con sus distintos componentes(tablas, campos, relaciones) mediante el uso de clases, en este caso escritas en Python.

Note

Para mas información revisar [Modelos de datos en Django](#).

Importaciones Necesarias

Importando el modulo principal para el diseño de los modelos:

```
from django.db import models
```

Para realizar la conexión y consultas avanzadas y demás modulos necesarios:

```
from django.db import connection
from django.db.models import Q
from django.db.models import Count
from django.db.models import exceptions
```

Se utiliza el modelo por defecto que nos brinda Django para la administración de usuarios:

```
from django.contrib.auth.models import User
```

Otros moulous usados para trabajar con fechas, xml, diccionarios y logs:

```
from datetime import datetime
from ordereddict import OrderedDict
import lxml.html
import logging
```

Se importa la clase SGA que obtiene toda la información academica necesaria para diversos sistemas clientes del SGA:

```
from proyecto.tools.sgaws.cliente import SGA
```

El modulo de configuración general:

```
from proyecto import settings
```

Asignando el logger a una variable logg:

```
logg = logging.getLogger('logapp')
```

Clase Configuracion

La clase Configuración permite realizar una relación **Uno a Uno** entre la *Clase PeriodoAcademico* y la *Clase PeriodoEvaluacion*:

```
class Configuracion(models.Model):
    periodoAcademicoActual = models.OneToOneField(
        'PeriodoAcademico',
        null=True,
        blank=True,
        verbose_name='Periodo Académico Actual'
    )
    periodoEvaluacionActual = models.OneToOneField(
        'PeriodoEvaluacion',
        null=True, blank=True,
        verbose_name='Periodo Evaluación Actual'
    )
```

Se obtiene el periodo académico actual:

```
@classmethod
def getPeriodoAcademicoActual(self):
    return Configuracion.objects.get(id=1).periodoAcademicoActual
```

Lo mismo que el anterior pero con el periodo de evaluación:

```
@classmethod
def getPeriodoEvaluacionActual(self):
    return Configuracion.objects.get(id=1).periodoEvaluacionActual
```

El nombre que presentará del modelo:

```
class Meta:
    verbose_name = u'Configuraciones'
    verbose_name_plural = u'Configuraciones'
```

Indica la representación textual (en Unicode) que deseamos que tengan las instancias de este modelo:

```
def __unicode__(self):
    return u"Configuraciones de la Aplicación"
```

Clase OfertaAcademicaSGA

```
class OfertaAcademicaSGA(models.Model):
    idSGA = models.IntegerField(verbose_name='Id_SGA', unique=True, db_column='id_sga', null=True)
    descripcion = models.CharField(max_length='100', verbose_name='Descripción')

    class Meta:
        verbose_name = u'Oferta Académica'
        verbose_name_plural = u'Ofertas Académicas'

    def __unicode__(self):
        return self.descripcion
```

Clase PeriodoAcademico

Los Modelos

Su finalidad es definir el periodo académico en el cual se realizará la encuesta:

```
class PeriodoAcademico(models.Model):
    nombre = models.CharField(max_length='50')
    inicio = models.DateField()
    fin = models.DateField()
    periodoLectivo = models.CharField(max_length=100, db_column='periodo_lectivo')
```

Relacion muchos a muchos con la *OfertaAcademicaSGA*:

```
ofertasAcademicasSGA = models.ManyToManyField('OfertaAcademicaSGA', related_name='periodosAcademicos')
```

Se cargan las ofertas presentes en el **SGA** usando en usuario y contraseña del sistema, existe una excepción de que se agregan solo en el caso que no existan aún la oferta académica:

```
def cargarOfertasSGA(self):
    proxy = SGA(settings.SGAWS_USER, settings.SGAWS_PASS)
    ofertas_dict = proxy.ofertas_academicas(self.inicio, self.fin)
    ofertas = [OfertaAcademicaSGA(idSGA=oa['id'], descripcion=oa['descripcion']) for oa in ofertas_dict]
    for oa in ofertas:
        try:
            OfertaAcademicaSGA.objects.get(idSGA=oa.idSGA)
        except OfertaAcademicaSGA.DoesNotExist:
            oa.periodoAcademico = self
            oa.save()
```

Cada vez que se crea un nuevo Periodo Académico se consultan las ofertas academicas del SGA para adherirlas. Luego se pueden eliminar desde la aplicación de administración:

```
def save(self, *args, **kwargs):
    nuevo = False
    if not self.pk:
        nuevo = True
    super(PeriodoAcademico, self).save(*args, **kwargs)
    if nuevo:
        try:
            self.cargarOfertasSGA()
        except Exception, ex:
            logg.error("Error al cargar ofertas academicas del SGA: {}".format(ex))

class Meta:
    ordering = ['inicio']
    verbose_name = u'Periodo Académico'
    verbose_name_plural = u'Periodos Académicos'
```

El rango de fechas que abarca dicho periodo:

```
def rango(self):
    return '{0} / {1}'.format(self.inicio.strftime('%b %Y'), self.fin.strftime('%b %Y'))

def __unicode__(self):
    return self.nombre
```

Clase Asignatura

Como su nombre lo indica, en esta clase se administran las distintas asignaturas:

```
class Asignatura(models.Model):
    area = models.CharField(max_length='20')
    carrera = models.CharField(max_length='100')
    semestre = models.CharField(max_length='10', verbose_name=u'módulo')
    paralelo = models.CharField(max_length='50')
    seccion = models.CharField(max_length='10')
    modalidad = models.CharField(max_length='20')
```

Los Modelos

```
nombre = models.TextField()
tipo = models.CharField(max_length='15')
creditos = models.IntegerField(verbose_name=u'número de créditos')
duracion = models.FloatField(verbose_name=u'duración en horas')
inicio = models.DateField(null=True, verbose_name='inicia')
fin = models.DateField(null=True, verbose_name='termina')
# Campo combinado id_unidad:id_paralelo
idSGA = models.CharField(max_length='15', db_column='id_sga')
```

Clave foránea hacia la *Clase PeriodoAcademico*:

```
periodoAcademico = models.ForeignKey('PeriodoAcademico', related_name='asignaturas',
                                      verbose_name=u'Periodo Académico', db_column='periodo_a')
```

Determina si esta vigente es decir si la asignatura se dicta dentro del Periodo de Evaluación Actual:

```
def esVigente(self):
    periodoEvaluacion = Configuracion.getPeriodoEvaluacionActual()
    if self.inicio <= periodoEvaluacion.fin and self.fin >= periodoEvaluacion.inicio:
        return True
    else:
        return False
```

El tipo de dominio pertenece la asignatura:

```
def getTipo(self):
    tipos = [u'taller', u'curso', u'módulo', u'modulo', u'unidad']
    l = [t for t in tipos if t in self.nombre.lower()]
    return l[0] if l else u'otro'
```

Para guardar la información:

```
def save(self, *args, **kwargs):
    self.tipo = self.getTipo()
    super(Asignatura, self).save(*args, **kwargs)

def __unicode__(self):
    return u"\u201c{0} - {1}\u201d.format(self.idSGA, self.nombre)
```

Clase EstudiantePeriodoAcademico

Esta clase básicamente permite realizar una relación entre los distintos estudiantes y sus respectivos períodos académicos en los que se encuentran al momento de realizar la encuesta. Para esto primeramente crearemos un enlace hacia la *Clase Usuario*:

```
class EstudiantePeriodoAcademico(models.Model):
    usuario = models.ForeignKey('Usuario', related_name='estudiantePeriodosAcademicos')
```

Clave foránea de la *Clase PeriodoAcademico*:

```
periodoAcademico = models.ForeignKey('PeriodoAcademico', related_name='estudiantes',
                                      verbose_name=u'Periodo Académico', db_column='periodo_ac')

class Meta:
    verbose_name = 'Estudiante'
    unique_together = ('usuario', 'periodoAcademico')
```

Para recuperar la cédula de los estudiantes:

```
def cedula(self):
    return self.usuario.cedula
```

Este método permite recuperar las distintas asignaturas que recibe cada estudiante con el docente que las imparte y luego se construye una lista de diccionarios con la información obtenida:

```

def paralelos(self):
    consulta = self.asignaturasDocentesEstudiante.values_list('asignaturaDocente_asignatura',
                                                               'asignaturaDocente_asignatura',
                                                               'asignaturaDocente_asignatura',
                                                               'asignaturaDocente_asignatura',
                                                               'asignaturaDocente_asignatura')
    datos = [dict(zip(['area', 'carrera', 'modulo', 'paralelo', 'seccion'], r)) for r in consulta]
    return datos

def __unicode__(self):
    return self.usuario.get_full_name()

```

Clase EstudianteAsignaturaDocente

Permite realizar una relación entre estudiantes y las asignaturas de los mismos. Para empezar tenemos dos claves foráneas hacia la *Clase EstudiantePeriodoAcademico* y la *Clase AsignaturaDocente* con el fin de obtener una relación general entre estudiantes, materias y docentes que las imparten:

```

class EstudianteAsignaturaDocente(models.Model):
    estudiante = models.ForeignKey('EstudiantePeriodoAcademico', related_name='asignaturasDocentes')
    asignaturaDocente = models.ForeignKey('AsignaturaDocente', related_name='estudiantesAsignados',
                                          verbose_name='Asignatura - Docente')
    matricula = models.IntegerField(blank=True, null=True)
    estado = models.CharField(max_length='60', blank=True, null=True)

```

Haciendo uso de los getters, recuperamos datos como el área, carrera, etc.:

```

def get_area(self):
    return self.asignaturaDocente.asignatura.area
get_area.short_description = 'Area'

def get_carrera(self):
    return self.asignaturaDocente.asignatura.carrera[:60]
get_carrera.short_description = 'Carrera'

def get_semestre(self):
    return self.asignaturaDocente.asignatura.semestre
get_semestre.short_description = 'Semestre'

def get_paralelo(self):
    return self.asignaturaDocente.asignatura.paralelo
get_paralelo.short_description = 'Paralelo'

def get_nombre_corto(self):
    return self.__unicode__():60
get_nombre_corto.short_description = 'Nombre'

def get_asignatura(self):
    return self.asignaturaDocente.asignatura

```

Campos para adicionar en el Admin a través del formulario *Formulario EstudianteAsignaturaDocente*:

```

carrera = property(get_carrera,)
semestre = property(get_semestre,)
paralelo = property(get_paralelo,)

class Meta:
    verbose_name = 'Estudiante Asignaturas'
    verbose_name_plural = 'Estudiantes y Asignaturas'
    unique_together = ('estudiante', 'asignaturaDocente')

```

```
def __unicode__(self):
    return u"{} >> {}".format(self.estudiante, self.asignaturaDocente)
```

Clase AsignaturaDocente

Una relación entre los docentes con sus respectivas asignaturas. Tenemos dos claves foraneas hacia la [Clase Asignatura](#) y la [Clase DocentePeriodoAcademico](#):

```
class AsignaturaDocente(models.Model):
    asignatura = models.ForeignKey('Asignatura', related_name='docentesAsignatura')
    docente = models.ForeignKey('DocentePeriodoAcademico', related_name='asignaturasDocente')
```

Recuperación, mediante getters, de datos importantes:

```
def get_idSGA(self):
    return self.asignatura.idSGA

def get_carrera(self):
    return self.asignatura.carrera[:60]
get_carrera.short_description = 'Carrera'

def get_semestre(self):
    return self.asignatura.semestre
get_semestre.short_description = 'Semestre'

def get_paralelo(self):
    return self.asignatura.paralelo
get_paralelo.short_description = 'Paralelo'

def get_nombre_corto(self):
    ancho = 60
    s = self.__unicode__()
    return s[:ancho]
get_nombre_corto.short_description = 'Nombre'
```

Campos para adicionar en el Admin a través del formulario [Formulario EstudianteAsignaturaDocente](#):

```
carrera = property(get_carrera,)
semestre = property(get_semestre,)
paralelo = property(get_paralelo,)

class Meta:
    verbose_name = 'Asignatura Docente'
    verbose_name_plural = 'Asignaturas y Docentes'
    unique_together = ('docente', 'asignatura')

def __unicode__(self):
    return u"{} >> {}".format(self.docente, self.asignatura.nombre)
```

Recuperar todas las carreras:

```
carreras = Asignatura.objects.values_list('carrera', 'carrera').order_by('carrera').distinct()
```

Clase DocentePeriodoAcademico

Obtener todas las carreras y las almacenamos en una variable:

```
carreras = Asignatura.objects.values_list('carrera', 'carrera').order_by('carrera').distinct()
```

Seguidamente se crea la clase que tiene como objetivo principal el relacionar a los docentes con sus respectivos períodos académicos:

```
class DocentePeriodoAcademico(models.Model):
    usuario = models.ForeignKey('Usuario', related_name='docentePeriodosAcademicos')
    periodoAcademico = models.ForeignKey('PeriodoAcademico', related_name='docentes',
                                         verbose_name=u'Periodo Académico', db_column='periodo_academico')
```

El siguiente atributo se agrega por efectos de migracion de docentes sin informacion de asignaturas:

```
carrera = models.CharField(max_length='500', choices=carreras, blank=True, null=True)
```

El siguiente atributo nos permite seleccionar si el docente pertenece o no a la Comision Academica de la Carrera:

```
parAcademico = models.BooleanField()
```

```
class Meta:
    verbose_name = 'Docente'
    unique_together = ('usuario', 'periodoAcademico')
```

Continuamos con getters para recuperar información, en este caso se devolverá una lista de tuplas (carrera, area) pero únicamente del PeriodoAcademicoActual:

```
def get_carreras_areas(self):
    lista_carreras_areas = []
    lista_carreras_areas_query = AsignaturaDocente.objects.filter(
        docente_periodoAcademico=Configuracion.getPeriodoAcademicoActual(),
        docente_id=self.id).values_list('asignatura_carrera', 'asignatura_area').distinct()
    lista_carreras_areas.extend(lista_carreras_areas_query)
    if self.carrera and self.carrera not in [c[0] for c in lista_carreras_areas_query]:
        lista_carreras_areas.append((self.carrera, ''))
    return lista_carreras_areas

def get_carreras(self):
    return [c[0] for c in self.get_carreras_areas() if c[0]]

def get_areas(self):
    return [c[1] for c in self.get_carreras_areas() if c[1]]

def paralelos(self):
    result = self.asignaturasDocente.values_list('asignatura_area', 'asignatura_carrera',
                                                'asignatura_semestre', 'asignatura_paralelo',
                                                'asignatura_seccion').distinct()
    datos = [dict(zip(['area', 'carrera', 'modulo', 'paralelo', 'seccion'], r)) for r in result]
    return datos

def cedula(self):
    return self.usuario.cedula

def __unicode__(self):
    return u'{0} {1}'.format(self.usuario.abreviatura, self.usuario.get_full_name())
```

Clase DireccionCarrera

Para el funcionamiento normal de esta clase se necesitan recuperar datos existentes ya en la base de datos, es así que a continuación se obtienen todas las carrera que riguen en el Periodo Académico Actual:

```
carreras_areas = AsignaturaDocente.objects.filter(
    docente_periodoAcademico=Configuracion.getPeriodoAcademicoActual()).values_list(
    'asignatura_carrera', 'asignatura_area').order_by(
    'asignatura_carrera').distinct()
carreras_areas = ['|'.join(c), '|'.join(c)] for c in carreras_areas
#carreras_areas = (('energia', 'Energia'), ('medicina', 'Medicina'))
```

Los Modelos

A continuación se estructura la clase para obtener las respectivas direcciones de carrera, para esto se especifica que docentes son coordinadores:

```
class DireccionCarrera(models.Model):
```

Nombre de la Carrera más el Área:

```
carrera = models.CharField(max_length=255, choices=carreras_areas, unique=True,
                           verbose_name=u'Carrera-Area')
```

Director o Coordinador de Carrera:

```
director = models.ForeignKey('DocentePeriodoAcademico', verbose_name=u"Coordinador",
                             related_name="direcciones")
def __unicode__(self):
    return u"Coordinación {0} - {1}".format(self.carrera, self.director.periodoAcademico.ran
```

Se recupera docentes y se separa el área y la carrera:

```
def get_docentes(self):
    ids_docentes = AsignaturaDocente.objects.filter(
        asignatura_carrera=self.carrera.split('|')[0], asignatura_area=self.carrera.split(
            ).values_list('docente_id', flat=True).distinct())
    docentes = DocentePeriodoAcademico.objects.filter(
        periodoAcademico=self.director.periodoAcademico, id__in=ids_docentes).order_by(
            'usuario_last_name', 'usuario_first_name')
    return docentes

class Meta:
    ordering = ['carrera']
    verbose_name = u'Coordinación de Carrera'
    verbose_name_plural = 'Coordinaciones de Carreras'
```

Clase TipoInformante

El tipo de informante que interactua con el sistema. Para el presente sistema tenemos informantes como: Estudiante, Docente, Directivo, EstudianteMED, EstudiantesIdiomas, etc.:

```
class TipoInformante(models.Model):
    tipo = models.CharField(max_length=50, unique=True)
    descripcion = models.CharField(max_length=200)

    def __unicode__(self):
        return self.tipo

    class Meta:
        ordering = ['descripcion']
```

Clase Cuestionario

Una de las clases más importantes. Encargada de almacenar el cuestionario que será desplegado para su contestación:

```
class Cuestionario(models.Model):
```

Nombre que se grabará en caso de no añadir un nombre para el cuestionario:

```
nombre = models.CharField(max_length=150, default='Cuestionario Sin Nombre')
titulo = models.CharField(max_length=255)
```

Frase para el encabezado de la encuesta:

```
encabezado = models.TextField()
```

El lapso de tiempo que estará vigente el cuestionario para su contestación:

Los Modelos

```
inicio = models.DateTimeField(u'Inicio de la Encuesta')
fin=models.DateTimeField(u'Finalización de la Encuesta')
```

Permite seleccionar la obligatoriedad de todas las preguntas del cuestionario:

```
preguntas_obligatorias = models.BooleanField(default=True)
informante = models.ForeignKey(TipoInformante)
```

El peso que tendrán cada pregunta, esto se da generalmente de acuerdo al Tipo de Informante:

```
peso = models.FloatField(default=1.0)
periodoEvaluacion = models.ForeignKey('PeriodoEvaluacion', blank=True, null=True,
                                         related_name='cuestionarios', verbose_name=u'Periodo d...')
```

Método que permite obtener todas las preguntas de todas las secciones y subsecciones:

```
def get_preguntas(self):
    preguntas = []
    for s in self.secciones.all():
        preguntas.extend(s.get_preguntas())
    return preguntas
```

El siguiente método crea una copia de un cuestionario incluyendo todas sus secciones y todas sus preguntas. Teniendo en cuenta que todos los objetos involucrados serán objetos nuevos:

```
def clonar(self):
    numero = Cuestionario.objects.count()
    nuevo = Cuestionario()
    nuevo.titulo = u'{0} (Clonado {1})'.format(self.titulo, str(numero+1))
    nuevo.encabezado = self.encabezado
    nuevo.inicio = self.inicio
    nuevo.fin = self.fin
    # No se relacionan para mayor flexibilidad
    nuevo.informante = None
    nuevo.periodoEvaluacion = None
    nuevo.save()
    for seccion in self.secciones.all():
        nuevaSeccion = Seccion()
        nuevaSeccion.titulo = seccion.titulo
        nuevaSeccion.descripcion = seccion.descripcion
        nuevaSeccion.orden = seccion.orden
        nuevaSeccion.seccionPadre = None
        nuevaSeccion.cuestionario = nuevo
        nuevaSeccion.save()
        for pregunta in seccion.preguntas.all():
            nuevaPregunta = Pregunta()
            nuevaPregunta.texto = pregunta.texto
            nuevaPregunta.orden = pregunta.orden
            nuevaPregunta.tipo = pregunta.tipo
            nuevaPregunta.seccion = nuevaSeccion
            nuevaPregunta.save()
            for item in pregunta.items.all():
                nuevoItem = ItemPregunta()
                nuevoItem.texto = item.texto
                nuevoItem.orden = item.orden
                nuevoItem.pregunta = nuevaPregunta
                nuevoItem.save()
    return nuevo

def __unicode__(self):
    return self.nombre
```

Clase Contestacion

Permite gestionar las contestaciones de las evaluaciones:

```
class Contestacion(models.Model):
    pregunta = models.IntegerField()
    respuesta = models.TextField()
```

Adicionales a la respuesta propiamente establecida:

```
observaciones = models.TextField(null=True, blank=True)
```

Clave foránea hacia la *Clase Evaluacion*:

```
evaluacion = models.ForeignKey('Evaluacion', related_name='contestaciones')
```

A continuación se devuelve el objeto *pregunta* a partir del atributo *id_pregunta*:

```
def get_pregunta(self):
    return Pregunta.objects.get(id=self.pregunta)

class Meta:
    ordering = ['pregunta']
    verbose_name = 'Respuesta'
    verbose_name_plural = 'Respuestas'

def __unicode__(self):
    return u'{0}:{1}'.format(self.pregunta, self.respuesta)
```

Clase Evaluacion

Contiene la estructura básica para la evaluación que se realizará:

```
class Evaluacion(models.Model):
```

Delega la lógica que determina el tipo de evaluación a los controladores. Se puede acceder a las evaluaciones y autoevaluaciones directamente. Para acceder a las evaluaciones de los estudiantes utilizar *Clase EstudianteAsignaturaDocente*:

```
fechaInicio = models.DateField()
fechaFin = models.DateField()
horaInicio = models.TimeField()
horaFin = models.TimeField()
```

Enlace hacia la *Clase Cuestionario* con el fin de obtener los cuestionarios para llevar a cabo la evaluación:

```
cuestionario = models.ForeignKey('Cuestionario', related_name='evaluaciones')
```

Evaluaciones de ESTUDIANTES:

```
estudianteAsignaturaDocente = models.ForeignKey('EstudianteAsignaturaDocente', related_name=
```

Evaluaciones de DOCENTES. Pueden ser evaluaciones y autoevaluaciones:

```
docentePeriodoAcademico = models.ForeignKey('DocentePeriodoAcademico', related_name='evaluac
```

Evaluaciones de PARES ACADEMICOS y # Docente Par Academico:

```
parAcademico = models.ForeignKey('DocentePeriodoAcademico', related_name='evaluaciones_par_a
```

Evaluaciones de DIRECCIONES DE CARRERA # Docente Director:

```
directorCarrera = models.ForeignKey('DocentePeriodoAcademico', related_name='evaluaciones_di
```

Evaluaciones de DIRECCIONES DE CARRERA # Nombre de la Carrera mas el Area:

```
carreraDirector = models.CharField(max_length=255, choices=carreras_areas,
                                    verbose_name=u'Carrera-Area', blank=True, null=True)
```

Los Modelos

Método principal que define el comportamiento del evaluador y sus distintos roles:

```
def evaluador(self):
```

Evaluación del Estudiante a sus docentes:

```
if self.estudianteAsignaturaDocente:  
    evaluador = self.estudianteAsignaturaDocente.estudiante
```

Evaluación del Director de Carrera:

```
elif self.directorCarrera and self.docentePeriodoAcademico:  
    evaluador = self.directorCarrera
```

Evaluación del Par Académico de la Carrera:

```
elif self.parAcademico and self.docentePeriodoAcademico:  
    evaluador = self.parAcademico
```

Autoevaluación del docente:

```
elif self.docentePeriodoAcademico:  
    evaluador = self.docentePeriodoAcademico  
return evaluador
```

El siguiente método define el comportamiento del evaluado:

```
def evaluado(self):
```

Evaluación del Estudiante a sus docentes:

```
if self.estudianteAsignaturaDocente:  
    evaluado = self.estudianteAsignaturaDocente.asignaturaDocente.docente
```

Evaluación del Director de la Carrera al docente:

```
elif self.directorCarrera and self.docentePeriodoAcademico:  
    evaluado = self.docentePeriodoAcademico
```

Evaluación del Par Académico de la Carrera al docente:

```
elif self.parAcademico and self.docentePeriodoAcademico:  
    evaluado = self.docentePeriodoAcademico
```

Autoevaluación del docente:

```
elif self.docentePeriodoAcademico:  
    evaluado = self.docentePeriodoAcademico  
return evaluado
```

```
class Meta:  
    verbose_name_plural = 'Evaluaciones'
```

```
def __unicode__(self):  
    return u'{0} - {1} - {2}|{3} - {4}|{5} - {6}'.format(  
        self.evaluador().director.cedula() if isinstance(self.evaluador(), DireccionCarrera)  
        self.evaluado().cedula(),  
        self.fechaInicio, self.horaInicio, self.fechaFin, self.horaFin,  
        self.cuestionario.informante  
    )
```

Clase Resultados

Clase para forzar un enlace desde el admin de la app:

```
class Resultados(models.Model):  
    class Meta:  
        verbose_name_plural = 'Resultados'
```

Clase TipoPregunta

Clase Abstracta relacionada con el tipo de pregunta:

```
class TipoPregunta(models.Model):
    tipo = models.CharField(max_length='20', unique=True)
    descripcion = models.CharField(max_length='100')

    def __unicode__(self):
        return self.tipo
```

Clase SeleccionUnica

Esta clase esta relacionada con la forma de selección de las respuestas de la encuesta, en este caso el tipo de selección es unica es decir mediante el uso de radio buttons:

```
class SeleccionUnica(TipoPregunta):
    def __init__(self):
        TipoPregunta.__init__(self)
        self.tipo = 'SeleccionUnica'
        self.descripcion = 'Se visualiza radio buttons'

    def __unicode__(self):
        return u'Selección Única'
```

Clase Seccion

Esta clase tiene como finalidad agrupar los cuestionarios en distintas secciones:

```
class Seccion(models.Model):
```

Nombre corto para identificación de objeto:

```
nombre = models.CharField(max_length='150', default=u'Sección de Cuestionario Sin Nombre')
titulo = models.CharField(max_length='200')
descripcion = models.TextField(blank=True, null=True)
orden = models.IntegerField()
codigo = models.CharField(max_length='20', null=True, blank=True)
ponderacion = models.FloatField(null=True, blank=True)
```

Una subsección esta relacionada con otra Sección en vez de un Cuestionario:

```
superseccion = models.ForeignKey('self', null=True, blank=True, db_column='superseccion_id',
                                 related_name='subsecciones', verbose_name=u'Sección Padre')
```

Una sección normalmente esta relacionada con un Cuestionario:

```
cuestionario = models.ForeignKey(Cuestionario, related_name='secciones', null=True, blank=True)
```

Método recursivo hasta llegar a la sección padre que tiene Cuestionario:

```
def get_cuestionario(self):
    if self.superseccion:
        return self.superseccion.get_cuestionario()
    elif self.cuestionario:
        return self.cuestionario
```

Método recursivo hasta llegar a las secciones que no tiene subsecciones:

```
def get_preguntas(self):
    preguntas = []
    if self.subsecciones.count() > 0:
        for sub in self.subsecciones.all():
            preguntas.extend(sub.get_preguntas())
    preguntas.extend(self.preguntas.all())
```

```

    return preguntas

def preguntas_ordenadas(self):
    return self.pregunta_set.order_by('orden')

def __unicode__(self):
    if self.cuestionario:
        return u'Sección {0}'.format(self.nombre)
    elif self.superseccion:
        return u'Subsección {0}'.format(self.nombre)

class Meta:
    ordering = ['orden']
    verbose_name = 'sección'
    verbose_name_plural = 'secciones'

```

Clase Pregunta

Contiene la estructura que tendrán las preguntas de los cuestionarios y con que otras clases estarán relacionadas:

```

class Pregunta(models.Model):
    codigo = models.CharField(max_length='20', null=True, blank=True)
    texto = models.TextField()
    descripcion = models.TextField(null=True, blank=True)

```

Observaciones adicionales a la contestación o respuesta. Se almacena solo un título o tema de las observaciones:

```

observaciones = models.CharField(max_length='70', null=True, blank=True)
orden = models.IntegerField()
tipo = models.ForeignKey(TipoPregunta)
seccion = models.ForeignKey(Seccion, related_name='preguntas')

```

Devuelve el código de la sección más el de la pregunta:

```

def get_codigo(self):
    codigo_seccion = self.seccion.codigo or str(self.seccion.orden)
    codigo_pregunta = self.codigo or str(self.orden)
    return '{0}.{1}'.format(codigo_seccion, codigo_pregunta)

def __unicode__(self):
    html = lxml.html.document_fromstring(self.texto)
    texto = u'{0}'.format(html.text_content())
    return texto

class Meta:
    ordering = ['seccion__orden', 'orden']

```

Clase ItemPregunta

Esta clase se diseñó con el fin de proporcionar las posibilidades de respuestas a las preguntas realizadas en el cuestionario:

```
class ItemPregunta(models.Model):
```

Valor para la contestación de la pregunta:

```
texto = models.CharField(max_length='50')
```

Notas adicionales aclaratorias para el Item de pregunta:

```
descripcion = models.CharField(max_length='70', null=True, blank=True)
```

Clave foránea hacia la [Clase Pregunta](#):

```
pregunta = models.ForeignKey(Pregunta, related_name='items')
orden = models.IntegerField()

def __unicode__(self):
    return self.texto

class Meta:
    ordering = ['-orden']
```

Clase AreaSGA

Clase destinada para la administración de areas pertenecientes al Sistema de Gestión Académico(SGA):

```
class AreaSGA(models.Model):
    siglas = models.CharField(max_length=10)
    nombre = models.CharField(max_length=256)

    def __unicode__(self):
        return self.siglas

    class Meta:
        ordering=[id]
```

Clase PeriodoEvaluacion

La presente clase tiene como finalidad ubicarnos en el presente periodo de evaluación:

```
class PeriodoEvaluacion(models.Model):
    nombre = models.CharField(max_length=100)
    titulo = models.CharField(max_length=300)
    descripcion = models.TextField(null=True)
    observaciones = models.TextField(null=True, blank=True)
    inicio = models.DateTimeField()
    fin = models.DateTimeField()
```

Clave foránea hacia la [Clase PeriodoAcademico](#):

```
periodoAcademico = models.ForeignKey('PeriodoAcademico', related_name='periodosEvaluacion',
```

Relación muchos a muchos con la [Clase AreaSGA](#):

```
areasSGA = models.ManyToManyField(AreaSGA, related_name='periodosEvaluacion', verbose_name=u'Áreas de Evaluación')

class Meta:
    ordering = ['inicio', 'fin']
    verbose_name = u'Periodo Evaluación'
    verbose_name_plural = u'Periodos de Evaluación'
```

Métodos para configuración de fechas:

```
def noIniciado(self):
    ahora = datetime.today()
    return ahora < self.inicio

def vigente(self):
    ahora = datetime.today()
    return self.inicio <= ahora <= self.fin

def finalizado(self):
    ahora = datetime.today()
    return ahora > self.fin
```

Los Modelos

El siguiente método analiza si el Estudiante a realizado todas las evaluaciones que le corresponden en este Periodo de Evaluación:

```
def verificar_estudiante(self, cedula):
    try:
        EstudiantePeriodoAcademico.objects.get(usuario__cedula=cedula, periodoAcademico=self.periodoAcademico)
    except EstudiantePeriodoAcademico.DoesNotExist:
        logg.error('Verificar estudiante: dni {} no existe'.format(cedula))
        return False

    evaluaciones = Evaluacion.objects.filter(
        estudianteAsignaturaDocente__estudiante__usuario__cedula=cedula).filter(
            cuestionario__periodoEvaluacion=self).count()
    total = EstudianteAsignaturaDocente.objects.filter(
        estudiante__usuario__cedula=cedula).filter(
            estudiante__periodoAcademico=self.periodoAcademico).count()
    restantes = total - evaluaciones
    mensaje = u"{}: total {}, evaluados {}, restan {}".format(cedula, total, evaluaciones, restantes)
    logg.info(mensaje)
    if restantes == 0:
        return True
    else:
        return False
```

Este método contabiliza las distintas evaluaciones realizadas por los Estudiantes:

```
def contabilizar_evaluaciones_estudiantes(self, area, carrera, semestre=None, paralelo=None):
    consulta = EstudianteAsignaturaDocente.objects.filter(
        estudiante__periodoAcademico = self.periodoAcademico,
        asignaturaDocente__asignatura__area=area,
        asignaturaDocente__asignatura__carrera=carrera)
    if semestre and semestre != '':
        consulta = consulta.filter(asignaturaDocente__asignatura__semestre=semestre)
    if paralelo and paralelo != '':
        consulta = consulta.filter(asignaturaDocente__asignatura__paralelo=paralelo)
    estudiantes = set([c.estudiante for c in consulta.all()])
    total = len(estudiantes)
    completados = 0
    faltantes = 0
    for e in estudiantes:
        if self.verificar_estudiante(e.usuario.cedula):
            completados += 1
        else:
            faltantes += 1
    return dict(estudiantes=total, completados=completados, faltantes=faltantes)
```

A continuación el método contabilizar_evaluadores cuenta los estudiantes, pares academicos y directores que HAYAN EVALUADO a por lo menos un docente. No se controla haber evaluado a todos los docentes:

```
def contabilizar_evaluadores(self):
    estudiantes = Evaluacion.objects.filter(
        cuestionario__periodoEvaluacion=self).values_list(
            'estudianteAsignaturaDocente__estudiante').distinct().count()
    docentes = Evaluacion.objects.filter(
        cuestionario__periodoEvaluacion=self, docentePeriodoAcademico__isnull=False,
        directorCarrera__isnull=True, parAcademico__isnull=True).count()
    pares = Evaluacion.objects.filter(
        cuestionario__periodoEvaluacion=self, parAcademico__isnull=False).values(
            'parAcademico').distinct().count()
    directores = Evaluacion.objects.filter(
        cuestionario__periodoEvaluacion=self, directorCarrera__isnull=False).values(
```

```
'directorCarrera').distinct().count()
return dict(estudiantes=estudiantes, docentes=docentes, pares=pares, directores=directores)
```

Método que permite contar los estudiantes, pares académicos y directores que HAN SIDO EVALUADOS:

```
def contabilizar_evaluados(self):
    porEstudiantes = Evaluacion.objects.filter(
        cuestionario_periodoEvaluacion=self, estudianteAsignaturaDocente__isnull=False).values(
            'estudianteAsignaturaDocente_asignaturaDocente_docente').distinct().count()
    porDocentes = Evaluacion.objects.filter(
        cuestionario_periodoEvaluacion=self, docentePeriodoAcademico__isnull=False,
        directorCarrera__isnull=True, parAcademico__isnull=True).count()
    porPares = Evaluacion.objects.filter(
        cuestionario_periodoEvaluacion=self, parAcademico__isnull=False).values(
            'docentePeriodoAcademico').distinct().count()
    porDirectores = Evaluacion.objects.filter(
        cuestionario_periodoEvaluacion=self, directorCarrera__isnull=False).values(
            'docentePeriodoAcademico').distinct().count()
    return dict(porEstudiantes=porEstudiantes, porDocentes=porDocentes, porPares=porPares, porDirectores=porDirectores)

def __unicode__(self):
    return self.nombre
```

Clase Tabulacion

Antes de la creación de la clase, se recuperan datos existentes ya en la base de datos con el fin de obtener los tipos de tabulaciones que el sistema puede realizar:

```
# TODO: Modelar de mejor manera la funcionalidad
tipos_tabulacion = (
    (u'ESE2012', u'Tabulación Satisfacción Estudiantil 2012'),
    (u'EAAD2012', u'Tabulación Actividades Adicionales Docencia 2011-2012'),
    (u'EDD2013', u'Tabulación Evaluación del Desempeño Docente 2012-2013')
)
```

Está estructurada como una superclase que permite procesar la información generada por un conjunto de encuestas pertenecientes a un Periodo de Evaluación:

```
class Tabulacion(models.Model):
    descripcion = models.CharField(max_length='250')
    tipo = models.CharField( max_length='20', unique=True, choices=tipos_tabulacion)
```

Relación uno a uno con la [Clase PeriodoEvaluacion](#):

```
periodoEvaluacion = models.OneToOneField('PeriodoEvaluacion', related_name='tabulacion', blank=True, null=True)
class Meta:
    verbose_name_plural = "Tabulaciones"

def __unicode__(self):
    return self.descripcion
```

Clase TabulacionEvaluacion2013

Esta clase se encarga del levantamiento y presentación de resultados de la evaluación 2013:

```
class TabulacionEvaluacion2013:
    tipo = u'EDA2013'
    descripcion = u'Evaluación del Desempeño Docente 2012-2013'
```

Constructor:

Los Modelos

```
def __init__(self, periodoEvaluacion=None):
    self.periodoEvaluacion = periodoEvaluacion
    self.calculos = (
        # codigo, descripcion, metodo, titulo
        ('a', u'Resultados de la Evaluación del Desempeño Académico POR DOCENTE',
         self.por_docente, u'Evaluación del Desempeño Académico por Docente'),
        ('b', u'Resultados de la Evaluación del Desempeño Académico POR CARRERA',
         self.por_carrera, u'Evaluación del Desempeño Académico por Carrera'),
        ('c', u'Resultados de la Evaluación del Desempeño Académico POR AREA',
         self.por_area, u'Evaluación del Desempeño Académico por Area'),
    )
```

Calculo de resultados por docentes:

```
def por_docente(self, siglas_area, nombre_carrera, id_docente, componente=None):
    # Se pasa un tupla no un solo id
    return self.calcular(siglas_area, nombre_carrera, (id_docente,), componente)
```

Método para obtener resultados por carreras:

```
def por_carrera(self, siglas_area, nombre_carrera, componente):
```

Obtenemos los id de los Docentes que dictan Asignaturas en la carrera seleccionada:

```
aux_ids = AsignaturaDocente.objects.filter(
    docente_periodoAcademico=self.periodoEvaluacion.periodoAcademico,
    asignatura_carrera=nombre_carrera,
    asignatura_area=siglas_area
).values_list('docente_id', flat=True).distinct()
```

Se agregan también los docentes que no tengan Asignaturas pero que pertenezcan a la Carrera:

```
ids_docentes = DocentePeriodoAcademico.objects.filter(
    Q(periodoAcademico=self.periodoEvaluacion.periodoAcademico) and
    (Q(id_in=aux_ids) or Q(carrera=nombre_carrera))
).order_by('usuario_last_name', 'usuario_first_name').values_list(
    'id', flat=True
)
return self.calcular(siglas_area, nombre_carrera, ids_docentes, componente)
```

Método para cálculo por área:

```
def por_area(self, siglas_area, componente=None):
```

Obtenemos los id de los Docentes que dictan Asignaturas en el area seleccionada:

```
aux_ids = AsignaturaDocente.objects.filter(
    docente_periodoAcademico=self.periodoEvaluacion.periodoAcademico,
    asignatura_area=siglas_area
).values_list('docente_id', flat=True).distinct()
```

Se agregan tambien los docentes que no tengan Asignaturas pero que pertenezcan a la Carrera:

```
ids_docentes = DocentePeriodoAcademico.objects.filter(
    Q(periodoAcademico=self.periodoEvaluacion.periodoAcademico) and
    ( Q(id_in=aux_ids) #TODO: No hay el area en el atributo Carrera de DocentePeriodoAcademico
    ).order_by('usuario_last_name', 'usuario_first_name').values_list(
        'id', flat=True
)
return self.calcular(siglas_area, None, ids_docentes, componente)
```

Los Modelos

Método genérico para calcular los resultados de acuerdo a los diferentes criterios. Si se trata del reporte de sugerencias se salta al método respectivo:

```
def calcular(self, siglas_area, nombre_carrera, ids_docentes, componente=None):
    if componente == 'sugerencias':
        return self.extraer_sugerencias(siglas_area, nombre_carrera, ids_docentes)

    resultados_indicadores = {}
    pesos = {}
    if siglas_area == 'ACE':
        tipos = ('EstudianteIdiomas', 'DocenteIdiomas', 'ParAcademicoIdiomas', 'DirectivoIdiomas')
    else:
        tipos = ('Estudiante', 'Docente', 'ParAcademico', 'Directivo')
    if not componente:
        seccion_componente = None
    else:
```

Se pasa de string a objeto sección para sacar datos en la plantilla:

```
seccion_componente = Seccion.objects.filter(cuestionario_periodoEvaluacion=self.periodoEvaluacion).first()
```

Se obtienen los promedios tomando en cuenta cada indicador:

```
for tipo in tipos:
    cuestionario = Cuestionario.objects.get(periodoEvaluacion=self.periodoEvaluacion, informante=tipo)
```

En caso de tratarse del instituto de idiomas se generaliza el informante:

```
informante = tipo.lower().replace('idiomas', '')
```

Actualizar datos de peso:

```
pesos.update({informante : cuestionario.peso})
```

Recorriendo preguntas:

```
if not componente:
    preguntas = [p.id for p in cuestionario.get_preguntas() if p.tipo==TipoPregunta.objects.get()]
else:
    preguntas = [p.id for p in cuestionario.get_preguntas() if
                 p.tipo==TipoPregunta.objects.get(tipo='SeleccionUnica') and
                 p.seccion.superseccion.codigo==componente]
```

recorrido de contestaciones:

```
contestaciones = None
if informante == 'estudiante':
    contestaciones = Contestacion.objects.filter(
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id__in=ids_docentes,
        pregunta__in=preguntas).values_list('id', flat=True)
elif informante == 'docente':
    contestaciones = Contestacion.objects.filter(
        evaluacion_parAcademico__isnull=True, evaluacion_directorCarrera__isnull=True,
        evaluacion_docentePeriodoAcademico_id__in=ids_docentes, pregunta__in=preguntas
    ).values_list('id', flat=True)
elif informante == 'paracademico':
    contestaciones = Contestacion.objects.filter(
        evaluacion_parAcademico__isnull=False, evaluacion_directorCarrera__isnull=True,
        evaluacion_docentePeriodoAcademico_id__in=ids_docentes, pregunta__in=preguntas
    ).values_list('id', flat=True)
elif informante == 'directivo':
    contestaciones = Contestacion.objects.filter(
        evaluacion_directorCarrera__isnull=False, evaluacion_parAcademico__isnull=True,
        evaluacion_docentePeriodoAcademico_id__in=ids_docentes, pregunta__in=preguntas
    ).values_list('id', flat=True)
if contestaciones:
```

Los Modelos

Otención de promedios por pregunta:

```
cursor = connection.cursor()
cursor.execute('' "SELECT pregunta, AVG(respuesta::INT) FROM app_contestacion
    WHERE id IN %s GROUP BY pregunta" '', [tuple(contestaciones)])
result = cursor.fetchall()
cursor.close()
```

Si el informante no ha contestado el cuestionario correspondiente:

```
else:
```

```
    logg.warning('No hay evaluaciones del informante {0} para los docentes {1}'.format(tipo,
        print "No hay contestaciones de " + tipo + ' para ' + str(ids_docentes)
    result = [(id, 0.0) for id in preguntas]
```

A continuación se crea un diccionario a partir de lista compresa de tuplas conformadas por ids de pregunta con sus promedio:

```
promedios_preguntas = dict([(Pregunta.objects.get(id=id_pregunta), promedio) for id_pregunta in indicadores = {}]
```

Sumatorias por pregunta:

```
for pregunta, promedio in promedios_preguntas.items():
    suma = indicadores.get(pregunta.seccion, 0) + promedio
    ##indicadores[pregunta.seccion] = suma
    indicadores.update({pregunta.seccion: suma})
```

Promedio por sección (indicador):

```
for seccion, suma in indicadores.items():
    promedio = float(suma) / float(seccion.preguntas.count())
    indicadores.update({seccion : promedio})
```

Porcentaje por sección (indicador):

```
ESCALA_MAXIMA = 4
for seccion, promedio in indicadores.items():
    porcentaje = round((100 * promedio) / ESCALA_MAXIMA)
    indicadores.update({seccion : int(porcentaje)})
```

Genera diccionario de diccionarios. Los 'Objetos' Sección son diferentes pues pertenecen a Cuestionarios diferentes por tal motivo se usa el código como clave del diccionario para sumarizar los porcentajes de las secciones con el mismo código en todos los Cuestionarios:

```
for seccion, porcentaje in indicadores.items():
    indicador = resultados_indicadores.get(seccion.codigo, None)
    if not indicador:
        # Se guarda el atributo ponderacion solo la primera vez para luego calcular
        # Puesto que todas las secciones tienen los mismos datos

        indicador = {'informantes' : {}, 'ponderacion_seccion' : seccion.ponderacion}

    # Se inserta ademas el objeto Sección el indicador para disponer de información extra en el diccionario

    indicador.update({'objeto_seccion' : seccion})
    indicador['informantes'].update({ informante : porcentaje })
    resultados_indicadores.update({seccion.codigo : indicador})
```

Calculos totales en todos los indicadores de acuerdo al peso de los informantes:

```
aux_estudiante = []
aux_docente = []
aux_paracademico = []
aux_directivo = []
promedio_primaria = 0
```

Los Modelos

```
promedio_ponderada = 0
# Solo el codigo de la seccion
for seccion, resultado in resultados_indicadores.items():
    valores = resultado['informantes']
    informantes = valores.keys()
    informantes.sort()
    primaria = 0.0
```

Con un solo informante Comision Academica:

```
if informantes == ['directivo', 'paracademico']:
    # (pdir * vdir) + (ppa * vpa)) / (pdir + ppa)
    primaria = pesos['directivo'] * valores['directivo'] + pesos['paracademico'] * valores['paracademico']
    primaria = primaria / (pesos['directivo'] + pesos['paracademico'])
```

Con dos informantes:

```
elif informantes == ['directivo', 'docente', 'paracademico']:
    # ((mpne + pdir * vdir) + (mpne + ppa * vpa) + (??? + pd * cd)) / 2
    mitad = pesos['estudiante'] / 2
    primaria = (mitad / 2 + pesos['directivo']) * valores['directivo']
    primaria += (mitad / 2 + pesos['paracademico']) * valores['paracademico']
    primaria += (mitad + pesos['docente']) * valores['docente']
elif informantes == ['directivo', 'estudiante', 'paracademico']:
    # ((mpne + pdir * vdir) + (mpne + ppa * vpa) + (??? + pd * cd)) / 2
    mitad = pesos['docente'] / 2
    primaria = (mitad / 2 + pesos['directivo']) * valores['directivo']
    primaria += (mitad / 2 + pesos['paracademico']) * valores['paracademico']
    primaria += (mitad + pesos['estudiante']) * valores['estudiante']
elif informantes == ['docente', 'estudiante']:
    # (mpni + pe * ve) + (mpni + pd * vd)
    mitad = (pesos['directivo'] + pesos['paracademico']) / 2
    primaria = (mitad + pesos['estudiante']) * valores['estudiante']
    primaria += (mitad + pesos['docente']) * valores['docente']
```

Con tres informantes:

```
elif informantes == ['directivo', 'docente', 'estudiante', 'paracademico']:
    # (pe + ve) + ((pdir * vdir) + (ppa * vpa) + (pd * vd)) / 4
    primaria = pesos['estudiante'] * valores['estudiante']
    primaria += pesos['docente'] * valores['docente']
    primaria += pesos['paracademico'] * valores['paracademico']
    primaria += pesos['directivo'] * valores['directivo']

    primaria = round(primaria)
    resultado.update({'primaria': primaria})
    promedio_primaria += primaria
    ponderada = primaria * resultado['ponderacion_seccion'] / 100
    promedio_ponderada += ponderada
    resultado.update({'ponderada': ponderada})
    resultado.update({'cualitativa': self._cualificar_valor(primaria)})
    aux_estudiante.append(valores.get('estudiante', -1))
    aux_directivo.append(valores.get('directivo', -1))
    aux_docente.append(valores.get('docente', -1))
    aux_paracademico.append(valores.get('paracademico', -1))
```

En caso de que no se trata de un componente en particular:

```
if not componente:
    # Se envia a calcular los promedios por cada componente
    promedios_componentes = self._calcular_componentes(resultados_indicadores)
else:
    promedios_componentes = None
```

```

aux_estudiante = [e for e in aux_estudiante if e >= 0]
aux_docente = [e for e in aux_docente if e >= 0]
aux_paracademico = [e for e in aux_paracademico if e >= 0]
aux_directivo = [e for e in aux_directivo if e >= 0]

prom_estudiante = (sum(aux_estudiante) / float(len(aux_estudiante))) if aux_estudiante else 0
prom_docente = (sum(aux_docente) / float(len(aux_docente))) if aux_docente else 0
prom_paracademico = (sum(aux_paracademico) / float(len(aux_paracademico))) if aux_paracademico else 0
prom_directivo = (sum(aux_directivo) / float(len(aux_directivo))) if aux_directivo else 0

promedio_primaria = (promedio_primaria / len(resultados_indicadores)) if resultados_indicadores else 0

# Solo se suma la ponderacion hasta el final

promedios= {'estudiante' : prom_estudiante,
            'docente' : prom_docente,
            'paracademico' : prom_paracademico,
            'directivo' : prom_directivo,
            'primaria' : round(promedio_primaria,2),
            'ponderada' : promedio_ponderada,
            'cualitativa' : self._cualificar_valor(promedio_primaria)
        }
# Se ordena el diccionario por la clave (codigo del indicador)

resultados_indicadores = OrderedDict(sorted(resultados_indicadores.items(), key=lambda i: i[0]))
logg.info('Calculado docente: {} promedios: {} total: {}'.format(ids_docentes, promedios))
return dict(resultados_indicadores=resultados_indicadores, promedios_componentes=promedios_componentes,
            promedios=promedios, total=promedio_ponderada, seccion_componente=seccion_componente)

```

Método para cálculo de componentes CPF, CPG Y PV:

```

def _calcular_componentes(self, resultados_indicadores):
    promedios_componentes = {'CPF' : {'estudiante':[], 'docente':[], 'paracademico':[], 'directivo':[], 'primaria':[], 'ponderada':[], 'cualitativa':[]},
                             'CPG' : {'estudiante':[], 'docente':[], 'paracademico':[], 'directivo':[], 'primaria':[], 'ponderada':[], 'cualitativa':[]},
                             'PV' : {'estudiante':[], 'docente':[], 'paracademico':[], 'directivo':[], 'primaria':[], 'ponderada':[], 'cualitativa':[]}}
    for codigo_seccion, resultado in resultados_indicadores.items():
        componente = resultado['objeto_seccion'].superseccion.codigo
        promedios_componentes[componente]['estudiante'].append(resultado['informantes'].get('estudiante'))
        promedios_componentes[componente]['docente'].append(resultado['informantes'].get('docente'))
        promedios_componentes[componente]['paracademico'].append(resultado['informantes'].get('paracademico'))
        promedios_componentes[componente]['directivo'].append(resultado['informantes'].get('directivo'))
        promedios_componentes[componente]['primaria'].append(resultado.get('primaria', -1))
        promedios_componentes[componente]['ponderada'].append(resultado.get('ponderada', -1))
    for componente in ('CPF', 'CPG', 'PV'):
        for tipo_promedio in ('estudiante', 'docente', 'paracademico', 'directivo', 'primaria'):
            lista = [n for n in promedios_componentes[componente][tipo_promedio] if n >= 0]
            if tipo_promedio == 'ponderada':
                promedio = sum(lista)
            else:
                promedio = sum(lista)/len(lista)

            # En este momento se cambia el contenido tipo lista por un numero

            promedios_componentes[componente][tipo_promedio] = promedio
            promedios_componentes[componente]['cualitativa'] = self._cualificar_valor(

```

Los Modelos

```
    promedios_componentes[componente]['primaria'])
return promedios_componentes
```

Se cualifica con valores enteros:

```
def _cualificar_valor(self, valor):
    rangos = {'IS':range(0,41), 'PS':range(41,61), 'S':range(61,81), 'D':range(81,101)}
    for k,v in rangos.items():
        if v[0] <= round(valor) <= v[-1]:
            return k
    return ''
```

Método para la obtener todas las sugerencias:

```
def extraer_sugerencias(self, area, carrera, ids_docentes):
    sugerencias = {}
    if area == 'ACE':
        tipos = ('EstudianteIdiomas', 'DocenteIdiomas', 'ParAcademicoIdiomas', 'DirectivoIdiomas')
    else:
        tipos = ('Estudiante', 'Docente', 'ParAcademico', 'Directivo')
    for id_docente in ids_docentes:
        resultado = {}
```

Solo nombres del docente:

```
nombre_docente = DocentePeriodoAcademico.objects.get(id=id_docente).__unicode__()
for tipo in tipos:
    informante = tipo.lower().replace('idiomas', '')
```

Código es PV, CPG o CPF:

```
preguntas_ensayo = Pregunta.objects.filter(
    seccion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
    seccion_cuestionario_informante_tipo=tipo, tipo_tipo='Ensayo'
).values('id', 'seccion_codigo')
resultado[tipo] = []
```

Contendrá 3 ítems, uno por cada componente:

```
aux_contestaciones = {}
if informante == 'estudiante':
    for pregunta in preguntas_ensayo:
        contestaciones = Contestacion.objects.filter(
            evaluacion_cuestionario_informante_tipo_icontains='estudiante',
            evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente,
            pregunta=pregunta['id']).values_list('respuesta', flat=True)
        aux_contestaciones[pregunta['seccion_codigo']] = contestaciones
elif informante == 'docente':
    for pregunta in preguntas_ensayo:
        contestaciones = Contestacion.objects.filter(
            evaluacion_parAcademico_isnull=True, evaluacion_directorCarrera_isnull=True,
            evaluacion_docentePeriodoAcademico_id=id_docente, pregunta=pregunta['id']
        ).values_list('respuesta', flat=True)
        aux_contestaciones[pregunta['seccion_codigo']] = contestaciones
elif informante == 'paracademico':
    for pregunta in preguntas_ensayo:
        contestaciones = Contestacion.objects.filter(
            evaluacion_parAcademico_isnull=False, evaluacion_directorCarrera_isnull=True,
            evaluacion_docentePeriodoAcademico_id=id_docente, pregunta=pregunta['id']
        ).values_list('respuesta', flat=True)
        aux_contestaciones[pregunta['seccion_codigo']] = contestaciones
elif informante == 'directivo':
    for pregunta in preguntas_ensayo:
        contestaciones = Contestacion.objects.filter(
```

```

        evaluacion_directorCarrera__isnull=False, evaluacion_parAcademico__isnull=True
        evaluacion_docentePeriodoAcademico__id=id_docente, pregunta=pregunta['id']
    ).values_list('respuesta', flat=True)
aux_contestaciones[pregunta['seccion_codigo']] = contestaciones

```

Se supone que en los tres componentes hay la misma cantidad de respuestas CPF, CPG y PV:

```

num_contestaciones = len(aux_contestaciones.values()[0])
for i in range(num_contestaciones):
    aux_dict = {'CPF' : aux_contestaciones['CPF'][i],
                'CPG' : aux_contestaciones['CPG'][i],
                'PV' : aux_contestaciones['PV'][i]}
    resultado[tipo].append(aux_dict)
sugerencias[nombre_docente] = resultado
return dict(sugerencias=sugerencias)

```

Clase TabulacionAdicionales2012

Tabulación de parámetros adicionales:

```

class TabulacionAdicionales2012:
    tipo = u'EAAD2012'
    descripcion = u'Evaluación Actividades Adicionales a la Docencia 2012'

```

Constructor:

```

def __init__(self, periodoEvaluacion=None):
    self.periodoEvaluacion = periodoEvaluacion
    self.calculos = (
        # codigo, descripcion, metodo, titulo
        ('a', u'Resultados de la Evaluacion de Actividades Adicionales a la Docencia POR DOCENTE',
         self.por_docente, u'Evaluación Actividades Adicionales por Docente'),
        ('b', u'Resultados de la Evaluación de Actividades Adicionales a la Docencia POR CARRERA',
         self.por_carrera, u'Evaluación Actividades Adicionales por Carrera'),
    )

```

Método para obtener resultados de la Evaluación de Actividades Adicionales a la Docencia 2012 POR DOCENTE:

```

def por_docente(self, siglas_area, nombre_carrera, id_docente):
    docente = DocentePeriodoAcademico.objects.get(id=id_docente)

```

Procesamiento de la Autoevaluacion del Docente:

```

try:

    # Autoevaluacion de Actividades Adicionales del Docente

    autoevaluacion=docente.evaluaciones.get(cuestionario_periodoEvaluacion_id=2,
                                             cuestionario_informante_tipo='Docente')
except exceptions.MultipleObjectsReturned:

    # Existe una evaluacion duplicada

    logg.warning('Autoevaluacion duplicada docente {0} en periodo:{1}'.format(docente, 2))

    # Se toma la primera evaluacion

    autoevaluacion = docente.evaluaciones.filter(cuestionario_periodoEvaluacion_id=2,
                                                cuestionario_informante_tipo='Docente')[0]

except Evaluacion.DoesNotExist:
    logg.warning(u'No existe autoevaluación del docente {0} en periodo:{1}'.format(docente,
    return None

```

Solo de selección única:

```
contestaciones1 = [c for c in autoevaluacion.contestaciones.all() if Pregunta.objects.get(id=c.id).pregunta]
total1 = sum([int(c.respuesta) for c in contestaciones1])
peso = (total1 / float(len(contestaciones1))) if len(contestaciones1) > 0 else 0
porcentaje1 = (peso * 100 / float(4))
```

Se coloca en Contestacion un objeto Pregunta en vez del id_pregunta entero:

```
for c in contestaciones1:
    c.pregunta = Pregunta.objects.get(id=c.pregunta)
```

Procesamiento de la Evaluacion de la Comision Academica:

try:

Evaluacion de Actividades Adicionales del Docente por parte de los Directivo

evaluacion=docente.evaluaciones.get(cuestionario_periodoEvaluacion_id=2,
 cuestionario_informante_tipo='Directivo')

except exceptions.MultipleObjectsReturned:

Existe una evaluacion duplicada

logg.warning('Evaluacion duplicada docente {0} en periodo:{1}'.format(docente, 2))

Se toma la primera evaluacion

evaluacion = docente.evaluaciones.filter(cuestionario_periodoEvaluacion_id=2,
 cuestionario_informante_tipo='Directivo')

except Evaluacion.DoesNotExist:

logg.warning(u'No existe evaluacion de directivos para el docente {0} en periodo:{1}'.format(docente, 2))
return None

Solo preguntas de seleccion única:

```
contestaciones2 = [c for c in evaluacion.contestaciones.all() if Pregunta.objects.get(id=c.pregunta)]
total2 = sum([int(c.respuesta) for c in contestaciones2])
peso = (total2 / float(len(contestaciones2))) if len(contestaciones2) > 0 else 0
porcentaje2 = (peso * 100 / float(4))
```

Se coloca en Contestacion un objeto Pregunta en vez del id_pregunta entero:

```
for c in contestaciones2:
    c.pregunta = Pregunta.objects.get(id=c.pregunta)
```

Valor Total obtenido con ponderacion: Comision Academica 80% - Docente 20%:

```
total = (porcentaje1 * 20 / 100) + (porcentaje2 * 80 / 100)
contestaciones = {}
contestaciones['secciones'] = []
for s in autoevaluacion.cuestionario.secciones.all():
    seccion = {'titulo':s.titulo, 'resultados': []}
    for p in s.preguntas.all():
        11 = [c.respuesta for c in contestaciones1 if c.pregunta.codigo == p.codigo]
        valor_docente = 11[0] if 11 else ''
        11 = [c.observaciones for c in contestaciones1 if c.pregunta.codigo == p.codigo]
        observaciones_docente = 11[0] if 11 else ''
        12 = [c.respuesta for c in contestaciones2 if c.pregunta.codigo == p.codigo]
        valor_comision = 12[0] if 12 else ''
        12 = [c.observaciones for c in contestaciones2 if c.pregunta.codigo == p.codigo]
        observaciones_comision = 12[0] if 12 else ''
        porcentaje_docente = int(valor_docente)*25 if valor_docente else ''
        porcentaje_comision = int(valor_comision)*25 if valor_comision else ''
        if valor_docente or valor_comision:
            seccion['resultados'].append({'codigo':p.codigo, 'texto':p.texto,
                                         'valor_comision':valor_comision, 'valor_docente':valor_docente})
```

```

        'porcentaje_comision':porcentaje_comision, 'porcen-
        'observaciones_comision':observaciones_comision,
        'observaciones_docente':observaciones_docente})}

    contestaciones['secciones'].append(seccion)
num_actividades = sum([len(s['resultados'])] for s in contestaciones['secciones']])
contestaciones['num_actividades'] = num_actividades
return dict(contestaciones1=contestaciones1, porcentaje1=porcentaje1,
            contestaciones2=contestaciones2, porcentaje2=porcentaje2,
            contestaciones=contestaciones, total=total)

```

Resultados de la Evaluación de Actividades Adicionales a la Docencia 2012 POR CARRERA(incompleto):

```

def por_carrera(self, siglas_area, nombre_carrera):
    pass

```

Clase TabulacionSatisfaccion2012

Tabulación de datos para la encuesta de satisfacción estudiantil año 2012:

```

class TabulacionSatisfaccion2012:
    tipo = u'ESE2012'
    descripcion = u'Encuesta de Satisfacción Estudiantil 2012'

    def __init__(self, periodoEvaluacion=None):
        self.periodoEvaluacion = periodoEvaluacion
        self.calculos = (
            # codigo, descripcion, metodo, titulo
            ('a', u'La valoración global de la Satisfacción Estudiantil por DOCENTE',
             self.por_docente, u'Satisfacción Estudiantil por Docente'),
            ('b', u'La valoración global de la Satisfacción Estudiantil por CARRERA',
             self.por_carrera, u'Satisfacción Estudiantil en la Carrera'),
            ('c', u'La valoración de la Satisfacción Estudiantil en cada uno de los CAMPOS',
             self.por_campos, u'Satisfacción Estudiantil en Por Campos Específicos'),
            ('d', u'La valoración estudiantil en cada uno de los INDICADORES por carrera',
             self.por_indicador, u'Satisfacción Estudiantil por Indicador'),
            ('e', u'Los 10 indicadores de mayor SATISFACCIÓN en la Carrera',
             self.mayor_satisfaccion, u'Indicadores de Mayor Satisfacción'),
            ('f', u'Los 10 indicadores de mayor INSATISFACCIÓN en la Carrera',
             self.mayor_insatisfaccion, u'Indicadores de mayor Insatisfacción'),
        )

```

Método para obtener resultados de la encuesta, POR DOCENTE. Satisfacción Estudiantil de un docente en los módulos, cursos, unidades o talleres:

```

def por_docente(self, siglas_area, nombre_carrera, id_docente):
    if siglas_area == u'ACE':
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionario,
                                             cuestionario_informante_tipo=u'EstudianteIdioma')
    else:
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionario,
                                             cuestionario_informante_tipo=u'Estudiante')

```

Para asegurar que se tomen únicamente preguntas que representen indicadores además. Se seleccionan solo ids para poder comparar:

```

indicadores=Pregunta.objects.filter(seccion__in=secciones).filter(tipo__tipo=u'SeleccionUnica')
conteo_ms=Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.periodoEvaluacion,
                                       evaluacion__cuestionario__periodoEvaluacion__tabulacion_tipo='ESE2012').filter(
                                           evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area,
                                           evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera)

```

```

# Unica diferencia con respecto al metodo 'por_carrera'

evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente).filter(
    pregunta__in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
        respuesta='4').order_by('pregunta')
conteo_s=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignatura_area=siglas_area,
evaluacion_estudianteAsignaturaDocente_asignatura_carrera=nombre_carrera)

# Unica diferencia con respecto al metodo 'por_carrera'

evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente).filter(
    pregunta__in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
        respuesta='3').order_by('pregunta')
conteo_ps=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignatura_area=siglas_area,
evaluacion_estudianteAsignaturaDocente_asignatura_carrera=nombre_carrera)

# Unica diferencia con respecto al metodo 'por_carrera'

evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente).filter(
    pregunta__in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
        respuesta='2').order_by('pregunta')
conteo_ins=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignatura_area=siglas_area,
evaluacion_estudianteAsignaturaDocente_asignatura_carrera=nombre_carrera)

# Unica diferencia con respecto al metodo 'por_carrera'

evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente).filter(
    pregunta__in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
        respuesta='1').order_by('pregunta')
conteos = []
for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo, por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_s:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ps:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ins:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for grado in ('MS', 'S', 'PS', 'INS'):
        if grado not in conteo.keys():

```

```

        conteo[grado] = 0
conteos.append(conteo)
totales = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    totales[grado] = sum([c[grado] for c in conteos])
universo = Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area).filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera)
    # Unica diferencia con respecto al método 'por_carrera'
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_docente_id=id_docente).filter(
    pregunta_in=indicadores).count()
totales['total'] = universo
porcentajes = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    if universo != 0:
        numero = totales[grado] * 100 / float(universo)
    else:
        numero = 0
    porcentajes[grado] = numero
porcentajes['MSS'] = porcentajes['MS'] + porcentajes['S']

return dict(conteos=conteos, totales=totales, porcentajes=porcentajes)

```

Método para obtener Resultados de la encuesta, POR CARRERA. Todas las Secciones de todos los cuestionarios que pertenecen al periodo de evaluación establecido:

```

def por_carrera(self, siglas_area, nombre_carrera):
    if siglas_area == u'ACE':
        secciones = Seccion.objects.filter(cuestionario_in=self.periodoEvaluacion.cuestionario).filter(
            cuestionario_informante_tipo=u'EstudianteIdioma')
    else:
        secciones = Seccion.objects.filter(cuestionario_in=self.periodoEvaluacion.cuestionario).filter(
            cuestionario_informante_tipo=u'Estudiante')

```

Para asegurar que se tomen únicamente preguntas que representen indicadores además, se seleccionan solo ids para poder comparar:

```

indicadores=Pregunta.objects.filter(seccion_in=secciones).filter(tipo_tipo=u'SeleccionUnica').values('pregunta').annotate(MS=Count('respuesta')).filter(
    respuesta='4').order_by('pregunta')
conteo_ms=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area).filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
    pregunta_in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
    respuesta='4').order_by('pregunta')
conteo_s=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area).filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
    pregunta_in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
    respuesta='3').order_by('pregunta')
conteo_ps=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area).filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
    pregunta_in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
    respuesta='2').order_by('pregunta')
conteo_ins=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area).filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
    pregunta_in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
    respuesta='1').order_by('pregunta')

```

```

pregunta__in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
    respuesta='1').order_by('pregunta')
conteos = []
for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_s:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ps:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ins:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for grado in ('MS', 'S', 'PS', 'INS'):
        if grado not in conteo.keys():
            conteo[grado] = 0
    conteos.append(conteo)
totales = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    totales[grado] = sum([c[grado] for c in conteos])
universo = Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nomb_
pregunta__in=indicadores).count()
totales['total'] = universo
porcentajes = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    if universo is not 0:
        numero = totales[grado] * 100 / float(universo)
    else:
        numero = 0
    porcentajes[grado] = numero
    porcentajes['MSS'] = porcentajes['MS'] + porcentajes['S']

return dict(conteos=conteos, totales=totales, porcentajes=porcentajes)

```

Método para obtener información POR CAMPOS:

```

def por_campos(self, siglas_area, nombre_carrera, id_seccion):
    # @param id_seccion representa el campo del cuestionario.

    Seccion = Campo
    if id_seccion == None:
        return None

```

La sección específica del cuestionario (por Área) que corresponde, ya se determina en la vista anterior:

```

seccion = Seccion.objects.get(id=id_seccion)
logg.info("Campo Seccion: " + str(seccion.id) + str(seccion))

```

```
# Se tratan únicamente preguntas abiertas

if seccion.orden == 4:
    return self.por_otros_aspectos(siglas_area, nombre_carrera, seccion)
```

Para asegurar que se tomen únicamente preguntas que representen indicadores además de que pertenezcan únicamente al campo (sección) especificado. Se seleccionan solo ids para poder comparar luego:

```
indicadores=Pregunta.objects.filter(seccion=seccion).filter(tipo_tipo=u'SeleccionUnica').values('pregunta')
conteo_ms=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera,
    pregunta_in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
        respuesta='4').order_by('pregunta')
conteo_s=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera,
    pregunta_in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
        respuesta='3').order_by('pregunta')
conteo_ps=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera,
    pregunta_in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
        respuesta='2').order_by('pregunta')
conteo_ins=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera,
    pregunta_in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
        respuesta='1').order_by('pregunta')
conteos = []
for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_s:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ps:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ins:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for grado in ('MS', 'S', 'PS', 'INS'):
        if grado not in conteo.keys():
            conteo[grado] = 0
    conteos.append(conteo)
totales = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    totales[grado] = sum([c[grado] for c in conteos])
universo = Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
    evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera)
```

```

evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_
pregunta_in=indicadores).count()
totales['total'] = universo
porcentajes = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    if universo is not 0:
        numero = totales[grado] * 100 / float(universo)
    else:
        numero = 0
    porcentajes[grado] = numero
porcentajes['MSS'] = porcentajes['MS'] + porcentajes['S']

return dict(conteos=conteos, totales=totales, porcentajes=porcentajes)

```

Método para obtener información POR OTROS ASPECTOS:

```

def por_otros_aspectos(self, siglas_area, nombre_carrera, seccion):
    indicadores=Pregunta.objects.filter(seccion=seccion).filter(tipo__tipo=u'Ensayo').values()

    conteo=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_
pregunta_in=indicadores).values('pregunta', 'respuesta').annotate(
frecuencia=Count('respuesta')).order_by('pregunta')

```

Para acceder a los datos del objeto pregunta en el template:

```

for c in conteo:
    c['pregunta'] = Pregunta.objects.get(id=c['pregunta'])
return conteo

```

Método para obtener información POR INDICADOR:

```

def por_indicador(self,siglas_area, nombre_carrera, id_pregunta ):

```

El id_pregunta representa el indicador del campo del cuestionario. Pregunta = Indicador.

La pregunta específica del cuestionario (por Área) que corresponde ya se determina en la vista anterior, se selecciona únicamente el id para la comparación posterior:

```

indicadores = (id_pregunta, )
conteo_ms=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area)
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
pregunta_in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
respuesta='4').order_by('pregunta')
conteo_s=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area)
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
pregunta_in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
respuesta='3').order_by('pregunta')
conteo_ps=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area)
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
pregunta_in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
respuesta='2').order_by('pregunta')
conteo_ins=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area)
evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nombre_carrera).filter(
pregunta_in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
respuesta='1').order_by('pregunta')

```

```

evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area
evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera
pregunta__in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
    respuesta='1').order_by('pregunta')

conteos = []
for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_s:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ps:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ins:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for grado in ('MS', 'S', 'PS', 'INS'):
        if grado not in conteo.keys():
            conteo[grado] = 0
    conteos.append(conteo)

totales = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    totales[grado] = sum([c[grado] for c in conteos])

universo = Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.
    evaluacion__cuestionario__periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area,
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera,
    pregunta__in=indicadores).count()

totales['total'] = universo
porcentajes = {}
for grado in ('MS', 'S', 'PS', 'INS'):
    if universo is not 0:
        numero = totales[grado] * 100 / float(universo)
    else:
        numero = 0
    porcentajes[grado] = numero
    porcentajes['MSS'] = porcentajes['MS'] + porcentajes['S']

return dict(conteos=conteos, totales=totales, porcentajes=porcentajes)

```

Parámetros que generan mayor satisfacción. Contiene todas las Secciones de todos los cuestionarios que pertenecen al periodo de evaluación establecido:

```

def mayor_satisfaccion(self, siglas_area, nombre_carrera):
    if siglas_area == u'ACE':
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionarios,
                                             cuestionario_informante_tipo=u'EstudianteIdioma')
    else:
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionarios,
                                             cuestionario_informante_tipo=u'Estudiante')

```

La siguiente sección de código tiene la finalidad de asegurar que se tomen únicamente preguntas que representen indicadores, además se seleccionan solo ids para poder comparar:

```

indicadores=Pregunta.objects.filter(seccion__in=secciones).filter(tipo__tipo=u'SelecciónÚnica')
conteo_ms=Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion__cuestionario__periodoEvaluacion__tabulacion__tipo='ESE2012').filter(
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area)
evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera)
pregunta__in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
    respuesta='4').order_by('pregunta')
conteo_s=Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion__cuestionario__periodoEvaluacion__tabulacion__tipo='ESE2012').filter(
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area)
evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera)
pregunta__in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
    respuesta='3').order_by('pregunta')
conteo_ps=Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion__cuestionario__periodoEvaluacion__tabulacion__tipo='ESE2012').filter(
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area)
evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera)
pregunta__in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
    respuesta='2').order_by('pregunta')
conteo_ins=Contestacion.objects.filter(evaluacion__cuestionario__periodoEvaluacion=self.periodoEvaluacion).filter(
    evaluacion__cuestionario__periodoEvaluacion__tabulacion__tipo='ESE2012').filter(
    evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__area=siglas_area)
evaluacion__estudianteAsignaturaDocente__asignaturaDocente__asignatura__carrera=nombre_carrera)
pregunta__in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
    respuesta='1').order_by('pregunta')
conteos = []

```

Se realiza el conteo por cada pregunta:

```

for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_s:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ps:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for c in conteo_ins:
        if c['pregunta'] == i:
            conteo.update(c)
            conteo['pregunta'] = Pregunta.objects.get(id=i)
    for grado in ('MS','S','PS','INS'):
        if grado not in conteo.keys():
            conteo[grado] = 0
    conteos.append(conteo)

```

Se ordena de mayor a menor por 'MS' luego por 'S':

```

conteos.sort(lambda c1, c2: -cmp(c1['MS'],c2['MS']) or -cmp(c1['S'],c2['S']))
return dict(conteos=conteos[:10], totales=None, porcentajes=None)

```

Método que contiene parámetros que producen mayor insatisfacción. Contiene todas las Secciones de todos los cuestionarios que pertenecen al periodo de evaluación establecido:

```
def mayor_insatisfaccion(self, siglas_area, nombre_carrera):
    if siglas_area == u'ACE':
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionarios,
                                             cuestionario_informante_tipo=u'EstudianteIdioma')
    else:
        secciones = Seccion.objects.filter(cuestionario__in=self.periodoEvaluacion.cuestionarios,
                                             cuestionario_informante_tipo=u'Estudiante')
```

Sección de código para asegurar que se tomen únicamente preguntas que representen indicadores además, se seleccionan solo ids para poder comparar:

```
indicadores=Pregunta.objects.filter(seccion__in=secciones).filter(tipo__tipo=u'SeleccionUnica')
conteos = self._contabilizar(siglas_area, nombre_carrera, indicadores)
```

A continuación se ordena de mayor a menor por 'INS' luego por 'PS':

```
conteos.sort(lambda c1, c2: -cmp(c1['INS'], c2['INS']) or -cmp(c1['PS'], c2['PS']))
return dict(conteos=conteos[:10], totales=None, porcentajes=None)
```

El parámetro *indicadores*: lista de ids de pregunta que se involucran en el conteo. El parámetro *id_docente*: para el caso único en el que no se contabiliza en toda la carrera:

```
def _contabilizar(self, siglas_area, nombre_carrera, indicadores=[], id_docente=None):
    conteo_ms=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
                                           evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nomb_carrera,
        pregunta__in=indicadores).values('pregunta').annotate(MS=Count('respuesta')).filter(
            respuesta='4').order_by('pregunta')
    conteo_s=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
                                           evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nomb_carrera,
        pregunta__in=indicadores).values('pregunta').annotate(S=Count('respuesta')).filter(
            respuesta='3').order_by('pregunta')
    conteo_ps=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
                                           evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nomb_carrera,
        pregunta__in=indicadores).values('pregunta').annotate(PS=Count('respuesta')).filter(
            respuesta='2').order_by('pregunta')
    conteo_ins=Contestacion.objects.filter(evaluacion_cuestionario_periodoEvaluacion=self.periodoEvaluacion,
                                           evaluacion_cuestionario_periodoEvaluacion_tabulacion_tipo='ESE2012').filter(
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_area=siglas_area,
        evaluacion_estudianteAsignaturaDocente_asignaturaDocente_asignatura_carrera=nomb_carrera,
        pregunta__in=indicadores).values('pregunta').annotate(INS=Count('respuesta')).filter(
            respuesta='1').order_by('pregunta')
    if id_docente:
        conteo_ms = conteo_ms.filter(evaluacion_estudianteAsignaturaDocente_asignaturaDocente_id=id_docente)
        conteo_s = conteo_s.filter(evaluacion_estudianteAsignaturaDocente_asignaturaDocente_id=id_docente)
        conteo_ps = conteo_ps.filter(evaluacion_estudianteAsignaturaDocente_asignaturaDocente_id=id_docente)
        conteo_ins = conteo_ins.filter(evaluacion_estudianteAsignaturaDocente_asignaturaDocente_id=id_docente)
    conteos = []
```

Se realiza el conteo por cada pregunta:

```
for i in indicadores:
    conteo = {}
    for c in conteo_ms:
        if c['pregunta'] == i:
            conteo.update(c)
            # Se intercambia por el objeto completo por versatilidad
            conteo['pregunta'] = Pregunta.objects.get(id=i)
```

```

for c in conteo_s:
    if c['pregunta'] == i:
        conteo.update(c)
        conteo['pregunta'] = Pregunta.objects.get(id=i)
for c in conteo_ps:
    if c['pregunta'] == i:
        conteo.update(c)
        conteo['pregunta'] = Pregunta.objects.get(id=i)
for c in conteo_ins:
    if c['pregunta'] == i:
        conteo.update(c)
        conteo['pregunta'] = Pregunta.objects.get(id=i)
for grado in ('MS', 'S', 'PS', 'INS'):
    if grado not in conteo.keys():
        conteo[grado] = 0
conteos.append(conteo)
return conteos

```

Clase Usuario

Clase principal. Contiene los atributos y métodos generales para los usuarios del sistema y administradores. Se crea el perfil de Usuario que podrá pertenecer a los grupos Estudiante y/o Docente:

```

class Usuario(User):
    cedula = models.CharField(max_length='15', unique=True)
    titulo = models.CharField(max_length='100', blank=True, null=True)

    def get_nombres(self):
        return self.first_name

    def set_nombres(self, nombres):
        self.first_name = nombres

    def get_apellidos(self):
        return self.last_name

    def set_apellidos(self, apellidos):
        self.last_name = apellidos

```

Abreviaturas para los distintos tipos de Títulos y sus equivalencias:

```

def get_abreviatura(self):
    equivalencias = {u'magister':u'Mg.', u'magíster':u'Mg.', u'ing.':u'Ing.', u'ingeniero':u'Ing.'
                    u'Ing.':u'Ing.', u'doctor':u'Dr.', u'docentra':u'Dra.', u'master':u'Ms.'
                    u'mg.':u'Mg.', u'licenciado':u'Lic.', u'licenciada':u'Lic.', u'economis'
                    u'dra.':u'Dra.', u'dr.':u'Dr.', u'lic.':u'Lic.', u'licdo.':u'Lic.', u'ing.'
                    u'odontólogo':u'Odont.', u'odontóloga':u'Odont.', u'odontologo':u'Odont.'
                    u'esp.':u'Esp.', u'Especialista':u'Esp.'}
    palabras = self.titulo.split() if self.titulo else ""
    for p in palabras:
        if p.lower() in equivalencias.keys():
            return equivalencias[p.lower()]
    return ""

```

Demás atributos que distinguen al usuario:

```

nombres = property(get_nombres, set_nombres)
apellidos = property(get_apellidos, set_apellidos)
abreviatura = property(get_abreviatura)

```

Presentación mediante cédula y nombre completo:

```
def __unicode__(self):
    return u'{0} {1}'.format(self.cedula, self.get_full_name());
```

Las plantillas

Introducción

Una plantilla de Django es una cadena de texto pensada para separar la presentación de un documento de sus datos. Una plantilla define variables y varios trozos de lógica básica (tags) que regulan la manera en que debería mostrarse el documento. Normalmente, las plantillas se usan para producir HTML, pero las plantillas de Django son igualmente capaces de generar cualquier formato basado en texto.

Para el presente sistema se uso las plantillas que por defecto nos facilita Django, usadas en la parte de administración de cada proyecto. Por lo general, una vez intalado Python y Django, dichas plantillas se encuentran ubicadas en C:\Python27\Lib\site-packages\django-1.5.2-py2.7.egg\django\contrib\admin\templates\admin.

Note

Para mas información visitar: [curso de Django en español](#).

Plantilla Base

Contiene el diseño básico de las pantallas que se presentarán al usuario:

```
{% load admin_static %}<!DOCTYPE html>
<html lang="{{ LANGUAGE_CODE|default:"en-us" }}"% {%- if LANGUAGE_BIDI %}dir="rtl"{%- endif %}>
```

Cabecera:

```
<head>
<title>{% block title %}{% endblock %}</title>
<link rel="stylesheet" type="text/css" href="{% block stylesheet %}{% static "admin/css/base.css" %}" />
{% block extrastyle %}{% endblock %}
<!--[if lte IE 7]><link rel="stylesheet" type="text/css" href="{% block stylesheet_ie %}" />
{%- if LANGUAGE_BIDI %}<link rel="stylesheet" type="text/css" href="{% block stylesheet_rtl %}" />
<script type="text/javascript">window.__admin_media_prefix__ = "{% filter escapejs %}{% static "admin/js/" %}"</script>
{% block extrahead %}{% endblock %}
{% block blockbots %}<meta name="robots" content="NONE,NOARCHIVE" />{% endblock %}
</head>
```

Carga de *internationalization*:

```
{% load i18n %}
```

Inicio del body:

```
1  <body class="{% if is_popup %}popup {%- endif %}{% block bodyclass %}{% endblock %}">
2  <!-- Container -->
3  <div id="container">
4      {% if not is_popup %}
5          <!-- Header -->
6          <div id="header">
7              <div id="branding">
8                  {% block branding %}{% endblock %}
9              </div>
10             {% if user.is_active and user.is_staff %}
11                 <div id="user-tools">
12                     {% trans 'Welcome,' %}
13                     <strong>{% filter force_escape %}{% firstof user.get_short_name user.get_full_name %}</strong>
14                 </div>
15             {% endif %}
16         </div>
17     <!-- Content -->
18     <div id="content">
19         <div id="main">
20             {% block main %}{% endblock %}
21         </div>
22         <div id="sidebar">
23             {% block sidebar %}{% endblock %}
24         </div>
25     </div>
26     <div id="footer">
27         <div id="site-name">
28             {% block site_name %}{% endblock %}
29         </div>
30         <div id="site-urls">
31             {% block site_urls %}{% endblock %}
32         </div>
33     </div>
34 </div>
```

```

14         { % block userlinks %}
15             { % url 'django-admindocs-docroot' as docsroot %}
16             { % if docsroot %}
17                 <a href="#">{% docsroot %}">{ % trans 'Documentation' %}</a> /
18             { % endif %}
19             { % if user.has_usable_password %}
20                 <a href="#">{% url 'admin:password\_change' %}">{ % trans 'Change password' %}</a>
21             { % endif %}
22                 <a href="#">{% url 'admin:logout' %}">{ % trans 'Log out' %}</a>
23             { % endblock %}
24         </div>
25     { % endif %}
26     { % block nav-global %}{ % endblock %}
27 </div>
28 <!-- END Header -->
29 { % block breadcrumbs %}
30 <div class="breadcrumbs">
31     <a href="#">{% url 'admin:index' %}">{ % trans 'Home' %}</a>
32     { % if title %} &rsaquo; {{ title }}{ % endif %}
33 </div>
34 { % endblock %}
35 { % endif %}
36 { % block messages %}
37     { % if messages %}
38         <ul class="messagelist">{ % for message in messages %}
39             <li{ % if message.tags %} class="#">{% message.tags %}"{ % endif %}>{{ message }}
40             { % endfor %}
41             { % endif %}
42     { % endblock messages %}
43 <!-- Content -->
44 <div id="content" class="#">{% block coltype %}colM{ % endblock %}>
45     { % block pretitle %}{ % endblock %}
46     { % block content_title %}{ % if title %}<h1>{{ title }}</h1>{ % endif %}{ % endblock %}
47     { % block content %}
48         { % block object-tools %}{ % endblock %}
49         {{ content }}
50     { % endblock %}
51     { % block sidebar %}{ % endblock %}
52         <br class="clear" />
53     </div>
54 <!-- END Content -->
55 { % block footer %}<div id="footer"></div>{ % endblock %}
56 </div>
57 <!-- END Container -->
58 </body>
```

Fin del html:

```
</html>
```

Plantillas para Administración

Asignaturas y docentes

Ubicación: dentro de {{TEMPLATE_DIRS}}/admin/app/asignaturadocente/change_list.html:

```

{ % extends "admin/change_list.html" %}
{ % load i18n %}

{ % block extrahead %}
```

Las plantillas

```
{{ block.super }}  
<script type="text/javascript">  
    (function($){  
        $(document).ready(function(){  
            //$("#paralelo").hide();  
            $('#changelist-form .actions').bind('click', function(){  
                //var seleccion=$('#action option:selected').val();  
                alert(seleccion);  
            });  
        });  
    })(django.jQuery);  
</script>  
{%- endblock %}  
  
{% block result_list %}  
<!-- Para clonar una Asignatura Docente en otro Paralelo --&gt;<br/><span id="paralelo"> <b> Paralelo: </b> <input type="text" name="paralelo"/> </span>  
  
{{ block.super }}  
  
{%- endblock %}
```

Evaluaciones

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/evaluacion/change_form.html:

```
{% extends "admin/change_form.html" %}  
{% load i18n %}  
  
{% block extrahead %}  
{{ block.super }}  
<style type="text/css">  
  
</style>  
{%- endblock %}  
  
{% block content_title %}  
<!-- Se anula el título por defecto de la página --&gt;<br/>{%- endblock %}  
  
{% block field_sets %}  
{{ original.cuestionario.informante }}: <b>{{ original.evaluador }}</b>  
<b> Docente evaluado: <b>{{ original.evaluado }}</b></b>  
<p/>  
{{ block.super }}  
{{ formsets }}  
  
<table>  
    {% for contestacion in original.contestaciones.all %}  
        <tr>  
            <td rowspan="2" > {{ contestacion.get_pregunta.get_codigo }} </td>  
            <td> {{ contestacion.get_pregunta }} </td>  
        </tr>  
        <tr>  
            <td style="border-bottom: 1px dashed;"> Respuesta: <b> {{ contestacion.respuesta }}</b>  
        </tr>  
    {% endfor %}  
</table>  
  
{%- endblock %}
```

```
{% block submit_buttons_bottom %}
<!-- Se elimina los botones de grabacion (submit) -->
{% endblock %}
```

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/evaluacion/change_list.html:

```
{% extends "admin/change_list.html" %}
{% load i18n %}

{% block extrahead %}
{{ block.super }}
<script language="javascript">
function resumen(){
    var win = window.open("/admin/resumen/evaluaciones", "resumen",
'height=500,width=800,resizable=yes,scrollbars=yes');
    win.focus();
}
</script>
{% endblock %}

{% block object-tools %}
{{ block.super }}
<ul class="object-tools">
- Resumen de Evaluaciones

</ul>
{% endblock %}
```

Periodo académico

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/periodoacademico/change_form.html:

```
{% extends "admin/change_form.html" %}
{% load i18n %}

{% block extrahead %}
{{ block.super }}
<script type="text/javascript">
(function($){
    $(document).ready(function($) {
        var logjs = "";
        /*
         Funcionalidad muy 'pesada', se resolvió hacerla por scripts
        $("#link_load_sga").click(function() {
            var r=confirm("Se va a cargar la información Académica del SGA");
            if (r == true){
                $.ajax({
                    url: "/sga/cargar_info_sga/{{ object_id }}",
                    method: "GET",
                    //data: "periodo_academico_id={{ object_id }}",
                    success: function (data){
                        alert(data);
                    },
                    error: function(xhr,status,error){
                        alert("error: " + status + "-" + error);
                    }
                });
            }
        });
    });
})(jQuery);
</script>
```

```

        }
    });

    $("#link_load_ofertas").click(function() {
        var r=confirm("Va a recargar ofertas académicas del SGA");
        if (r == true){
            $.ajax({
                url: "/sga/cargar_ofertas_sga/{{ object_id }}",
                method: "GET",
                //data: "periodo_academico_id={{ object_id }}",
                success: function (data){
                    // Regresamos a la lista de Periodos Académicos
                    $(window.location).attr('href','{% url 'admin:app_periodoacademi
                },
                error: function(xhr,status,error){
                    alert("error: " + status + "-" + error);
                }
            });
        }
    });
})(django.jQuery);
</script>
{% endblock %}

```

```

{% block after_field_sets %}
<br/>
<br/>
<ul class="object-tools">
    <li> <a id="link_load_ofertas" class="xxhistorylink"> Recargar Ofertas </a> </li>
</ul>
{% endblock %}

```

```

{% block before_related_objects %}

```

```

{% endblock %}

```

Preguntas en general

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/periodoacademico/change_form.html:

```

{% extends "admin/change_form.html" %}
{% load i18n %}

{% block extrahead %}
{{ block.super }}

<script type="text/javascript">
(function($){
    $(document).ready(function(){
        $('#id_tipo').change(function(){
            var seleccion=$('#id_tipo option:selected').val();
            // Tipo de pregunta: SeleccionUnica
            if (seleccion == '2'){
                $('#id_tipo').parent().append(
                    '<span id="activador"><input type="checkbox" id="tipo_predefinido" ! +
                    !name="tipo_predefinido" value="Valor">' +
                    'Predeterminado </input></span>'
                );
            }
        });
    });
})

```

```

        $('#tipo_predefinido').change(function(){
            if ($('#tipo_predefinido').is(':checked')) {
                $('#items-group').hide();
                var contenido = '<span id="opciones_grupo">';
                contenido += '<b> Tamaño: </b> <select name="longitud">' +
                    '<option value="" selected> ----- </option> ';
                for (var i=2; i<10; i++){
                    var selected = '';
                    if ( i==4 ){
                        selected = 'selected';
                    }
                    contenido += '<option value="' + i + '" ' + selected + '>' + i
                }
                contenido += '</select> <b> Numeración:</b>' +
                    '<input type="radio" name="numeracion" value="a"> Letras <input type="radio" name="numeracion" value="1" checked> Núme
                contenido += "</span>";
                $('#id_tipo').parent().append(contenido);
            } else {
                $('#opciones_grupo').remove();
                $('#items-group').show();
            }
        });
    }else{
        // Ocultar activador de SeleccionUnica por predeterminada
        $('#activador').remove();
        // Tipo de pregunta: Ensayo, no tiene items
        if (seleccion == 1){
            $('#items-group').hide();
        } else{
            // Cualquier otro tipo de pregunta
            $('#items-group').show();
        }
    }
});
})(django.jQuery);
</script>

{%
    endblock%
}

```

Formularios eaad2012

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/formulario_eaad2012.html:

```

<style type="text/css">
input[type=radio] {
    margin-top: 7px;
}

fieldset{
    margin: 15px 0px 15px 0px;
    width: 75%;
    padding: 10px 20px 10px 20px;
}
#id_campos, #id_indicadores{
    width: 75%;
}

</style>

```

```

<fieldset>
    <legend> Opciones de Resultados </legend>
    <ol type="a">
        <li>
            <label for="id_opciones_0">
                <input type="radio" id="id_opciones_0" value="a" name="opciones" />
                Resultados Evaluación Actividades Adicionales por Docente
            </label>
        </li>
        <ul>
            <li> {{ form.docentes }} </li>
        </ul>

        <!--li>
            <label for="id_opciones_1">
                <input type="radio" id="id_opciones_1" value="b" name="opciones" />
                Resultados Evaluación Actividades Adicionales por Carrera
            </label>
        </li-->
    </ol>
</fieldset>

```

Formularios edd2013

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/formulario_edd2013.html:

```

<style type="text/css">
input[type=radio] {
    margin-top: 7px;
}

fieldset{
    margin: 15px 0px 15px 0px;
    width: 75%;
    padding: 10px 20px 10px 20px;
}
#id_campos, #id_indicadores{
    width: 75%;
}

</style>
<fieldset>
    <legend> Opciones de Resultados </legend>

    <table>
        <tr>
            <!-- Columna 1 -->
            <td style="border-right: 1px solid; ">
                <b>Seleccione el tipo de Reporte:</b>
                <ol type="a">
                    <li>
                        <label for="id_opciones_0">
                            <input type="radio" id="id_opciones_0" value="a" name="opciones" />
                            Resultados POR DOCENTE
                        </label>
                    </li>
                    <ul>
                        <li> {{ form.docentes }} </li>
                    </ul>

```

```

<li>
    <label for="id_opciones_1">
        <input type="radio" id="id_opciones_1" value="b" name="opciones" />
        Resultados POR CARRERA
    </label>
</li>

<li>
    <label for="id_opciones_2">
        <input type="radio" id="id_opciones_2" value="c" name="opciones" />
        Resultados POR AREA
    </label>
</li>

</ol>
</td>

<!-- Columna 2 -->
<td>
    <b> Seleccione el Campo Específico: </b>
    <ol type="a">
        <li>
            <label for="id_filtros_0">
                <input type="radio" id="id_filtros_0" value="a" name="filtros" checked="checked" />
                Todos los COMPONENTES
            </label>
        </li>

        <li>
            <label for="id_filtros_1">
                <input type="radio" id="id_filtros_1" value="b" name="filtros" />
                Componente CAPACIDAD PROFESIONAL (CPF)
            </label>
        </li>

        <li>
            <label for="id_filtros_2">
                <input type="radio" id="id_filtros_2" value="c" name="filtros" />
                Componente CAPACIDAD PEDAGÓGICA (CPG)
            </label>
        </li>

        <li>
            <label for="id_filtros_3">
                <input type="radio" id="id_filtros_3" value="d" name="filtros" />
                Componente PRÁCTICA DE VALORES (PV)
            </label>
        </li>

        <li>
            <label for="id_filtros_4">
                <input type="radio" id="id_filtros_4" value="e" name="filtros" />
                Reporte de SUGERENCIAS
            </label>
        </li>

    </ol>
</td>
</tr>

```

```
</table>
</fieldset>
```

Formularios ese2012

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/formulario_ese2012.html:

```
<style type="text/css">
input[type=radio] {
    margin-top: 7px;
}

fieldset{
    margin: 15px 0px 15px 0px;
    width: 75%;
    padding: 10px 20px 10px 20px;
}
#id_campos, #id_indicadores{
    width: 75%;
}

</style>
<fieldset>
<legend> Opciones de Resultados </legend>
<ol type="a">
<li><label for="id_opciones_0"><input type="radio" id="id_opciones_0" value="a" name="opcion0" /> La valoración global de la Satisfacción Estudiantil por DOCENTE</label></li>
<ul>
<li> {{ form.docentes }} </li>
</ul>
</li>
<li><label for="id_opciones_1"><input type="radio" id="id_opciones_1" value="b" name="opcion1" /> La valoracion global de la Satisfacción Estudiantil por CARRERA</label></li>
<li><label for="id_opciones_2"><input type="radio" id="id_opciones_2" value="c" name="opcion2" /> La valoracion de la Satisfacción Estudiantil en cada uno de los CAMPOS, por carrera</label>
<ul>
<li> {{ form.campos }} </li>
</ul>
</li>

<li><label for="id_opciones_3"><input type="radio" id="id_opciones_3" value="d" name="opcion3" /> La valoración estudiantil en cada uno de los INDICADORES por carrera</label>
<ul>
<li>{{ form.indicadores }} </li>
</ul>
</li>
<li><label for="id_opciones_4"><input type="radio" id="id_opciones_4" value="e" name="opcion4" /> 0 indicadores de mayor SATISFACCIÓN en la Carrera</label></li>
<li><label for="id_opciones_5"><input type="radio" id="id_opciones_5" value="f" name="opcion5" /> 0 indicadores de mayor INSATISFACCIÓN en la Carrera</label></li>
</ol>
</fieldset>
```

Menú resultados

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/menu_resultados.html:

```
{% extends "admin/base_site.html" %}
{% block extrahead %}
<script type="text/javascript" src="/static/js/jquery-1.6.2.min.js"></script>
<script type="text/javascript" src="/static/js/sedd.js"></script>
```

```

<script type="text/javascript">
$(document).ready(function(){
    $('.boton').hide();
    menu_academico_ajax();
    // Para actualización de los dos campos Docente
    $('#id_carrera').change(function(){
        var id_periodoe = $("#id_período_evaluación option:selected").val();
        /* Si se trata de la Encuesta de Satisfacción Estudiantil */
        if (id_periodoe == 1){
            // Se oculta el primer campo Docente del menú general
            $("label[for=id_docente]").hide();
            $("#id_docente").hide();
        }
        $("#ver-opciones").click();
    });
    $('#ver-opciones').click(function(){
        /* NOTA: JavaScript no soporta nombres de variables largos! */
        var id_periodoe = $("#id_período_evaluación option:selected").val();
        var area = $("#id_area option:selected").val();
        var carrera = $("#id_carrera option:selected").val();
        if (id_periodoe != '' && area != '' && carrera != ''){
            $.ajax({
                url: "/resultados/periodo/" + id_periodoe + "/",
                method: "GET",
                data: {area:area, carrera:carrera},
                // El servicio devuelve un formulario HTML formateado
                dataType: 'html',
                success: function (data){
                    $("#menu-opciones").html(data);
                    $('.boton').show();
                },
                error: function(xhr,status,error){
                    alert("error: " + status + "-" + error);
                }
            });
        }else{
            $("#menu-opciones").html("");
            $('.boton').hide();
        }
    });
});
</script>
<style type="text/css">
body{
    font-size: 12px;
}
.botón {
    align: right;
    width: 150px;
    height: 30px;
    font-size: 15px;
}
/*select nth: eq(1) {
    width: 600px;
    font-size: 15px;
    font-weight: bold;
    color: #666666;
} */
td {

```

Las plantillas

```
    vertical-align: center;
}
</style>
{%
  block content_title %
}

{%
  block content %
  {{ block.super }}
<form action="/resultados/mostrar/" method="POST" target="_blank">

  {%
    csrf_token %
  }

  <table style="margin-bottom: 20px;">
    {{ form.as_table }}
  </table>
  <div id="menu-opciones">
  </div>
  <div id="acciones">
    <input class="boton" id="borrar" type="reset" value="Borrar"/>
    <input class="boton" id="procesar" type="submit" value="Aceptar"/>
  </div>
</form>
{%
  endblock %
}
```

Resumen de Evaluaciones

Ubicación: dentro de {{ TEMPLATE_DIRS }} /admin/app/resumen_evaluaciones.html:

```
{%
  extends "admin/base_site.html" %

  {%
    load i18n %
  }

  {%
    block extrahead %
  }
```

```

{{ block.super }}

<style>
input {
    font-size: 14px;
    font-weight: bold;
    text-align: right;
    margin-right: 20px;
}
img {
    vertical-align: middle;
}
</style>

<script type="text/javascript" src="/static/js/jquery-1.6.2.min.js"></script>
<script type="text/javascript" src="/static/js/sedd.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    menu_academico_ajax();
    /* Calculos de los datos de Resumen */
    $('#calcular').click(function(event){
        var pa = $('#id_periodo_academico option:selected').val();
        var c = $('#id_carrera option:selected').val();
        if ( pa != '' && c != '' ) {
            $('#cargando').show();
            $.ajax({
                method: 'GET',
                url: '/admin/resumen/calcular',
                data: {'id_periodo_evaluacion': $('#id_periodo_evaluacion option:selected'),
                        'area': $('#id_area option:selected').val(),
                        'carrera': $('#id_carrera option:selected').val(),
                        'semestre': $('#id_semestre option:selected').val(),
                        'paralelo': $('#id_paralelo option:selected').val()},
                dataType: 'JSON',
                success: function(response){
                    $('#estudiantes').val(response['estudiantes']);
                    $('#completados').val(response['completados']);
                    $('#faltantes').val(response['faltantes']);
                    $('#cargando').hide();
                },
                error: function(xhr,status,error){
                    alert("error: " + status + "-" + error);
                }
            });
            event.preventDefault();
        } // end if !=
    });
});
</script>
{%
    %>
{%
    %>
<h2> Total de Evaluadores del Periodo {{ periodoEvaluacion }} </h2>
<table width="100%" style="margin-bottom: 40px;">
    <tr>
        <td> <h3> Estudiantes: {{ evaluadores.estudiantes }} </h3></td>
        <td> <h3> Docentes: {{ evaluadores.docentes }} </h3> </td>
        <td> <h3> Pares Academico: {{ evaluadores.pares }} </h3></td>
        <td> <h3> Coordinadores: {{ evaluadores.directores }} </h3></td>
    </tr>
</table>

```

```

        </h3>
    </tr>
</table>
<h2> Detalle de Evaluaciones de Estudiantes </h2>
<table>
{{ form.as_table }}
</table>
<hr/>
<div style="padding-top: 10px; " >
    <button id="calcular" style="margin-right: 40px;"> Calcular </button>
    Total: <input type='text' id='estudiantes' size='10' readonly='readonly' />
    Han Evaluado: <input type='text' id='completados' size='10' readonly='readonly' />
    NO Evaluan: <input type='text' id='faltantes' size='10' readonly='readonly' />
    <br/>
    
</div>

{%
    % endblock %
}

```

Plantillas para usuarios y tareas

Portada

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/portada.html:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml11/
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<!-- Title -->
<title>SEED-UNL</title>

<!-- Link to the stylesheet -->
<link href="/static/style.css" type="text/css" rel="stylesheet" />
<!-- Scripts in js -->
<!--script type="text/javascript" src="/static/js/jquery-1.7.2.min.js"></script-->
<script type="text/javascript" src="/static/js/jquery-1.6.2.min.js"></script>
<script type="text/javascript" src="/static/js/efectos.js"></script>

</head>
<body>

<!-- Main Block -->
<div id="wrapper">

    <!-- Header -->
    <div id="header">
        <div id="logo">
            <a href="#"></a>
        </div>
        <div id="slogan">
            
        </div>
    </div>
    <!-- End header -->

    <!-- Content -->
    <div id="content">

```

```

<!-- Menu -->
<div id="menu">
    <ul>
        <li><a href="#">Inicio</a></li>
        <li><a href="/admin/">Acceso al sistema</a></li>
        <li><a href="http://www.unl.edu.ec">Acerca de...</a></li>
    </ul>
</div>
<!-- End menu -->

<!-- Left column -->
<div id="left">
    <h1 style="color: #8EAD45;"> Bienvenidos al sistema de evaluación de...
    <p>
        {%
            if periodoEvaluacionActual %
        <a href="#"><% url 'login' %}">
            >>> <u> <b style="color: black; font-weight: 250%; font-size: 14px;">
                {{ periodoEvaluacionActual.titulo|upper }} </b> </u>
        </a>
        {%
            else %
        >>> <b> No existen Procesos de Evaluacion actualmente </b>
        {%
            endif %
        }
    </p>
    <p>
        {{ periodoEvaluacionActual.descripcion }} <br/>
        <b>
            (Finaliza el {{ periodoEvaluacionActual.fin|date:"l"|"title" })
            {{ periodoEvaluacionActual.fin }}
        </b>
    </p>
</div>
<!-- End left column -->

<!-- Right column -->
<div id="right">
    
    <div id="caption">
        <p>&quot;En los tesoros de la sabiduría está la glori...
        </p>
    </div>
</div>
<!-- End right column -->
<div class="clear"></div>
</div>
<!-- End content -->

<!-- Footer -->
<div id="footer">
    <p>2012 &copy; Design by Iceman.</p>
</div>
<!-- End footer -->

</div>
<!-- End Main Block -->

</body>
</html>

```

Página de Inicio

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/index.html:

```
% extends "admin/base_site.html" %

{% block content_title %}

{% endblock %}

{% block content %}

<p>

<!-- Estudiante --&gt;
{% if request.session.estudiante %}
{% if request.session.carreras_estudiante|length &gt; 0 %}
&lt;h1&gt; Carreras del Estudiante &lt;/h1&gt;
&lt;ul style="margin-top: 20px;"&gt;
    {% for carrera in request.session.carreras_estudiante %}
        &lt;li&gt;
            &lt;a href="#"{{ url 'estudiante_asignaturas_docentes' carrera.num_carrera }}&gt;
                {{ carrera.nombre|upper }}
            &lt;/a&gt;
        &lt;/li&gt;
    &lt;h1&gt;
        {% endfor %}
&lt;/ul&gt;
&lt;hr/&gt;
{% endif %}
{% endif %}

<!-- End Estudiante --&gt;

<!-- Docente --&gt;

{% if request.session.docente %}
&lt;h1&gt; Encuestas para Autoevaluación del Docente &lt;/h1&gt;
&lt;!--ul style="margin-top: 20px;"--&gt;
&lt;!--h3--&gt;
    {## for cuestionario in request.session.cuestionarios_docente ##}
    &lt;!-- Se recorren los cuestionarios para autoevaluacion con su estado --&gt;
    {% for da in request.session.docente_autoevaluaciones %}
        &lt;ul class="#"{{ if da.evaliado }} messagelist {{ endif }}&gt;
            {% if forloop.first %} style="margin-top: 20px;" {{ endif }}&gt;
                &lt;h3&gt;
                    &lt;li&gt;
                        {% if da.evaliado %}
                            {{ da.cuestionario.titulo|upper }}
                        {% else %}
                            &lt;a href="#"{{ url 'encuestas' id_docente=request.session.docente.id id_asignatura=0 }}&gt;
                                {{ da.cuestionario.titulo|upper }}
                            &lt;/a&gt;
                        {% endif %}
                    &lt;/li&gt;
                &lt;/h3&gt;
            &lt;/ul&gt;
            {% empty %}
            &lt;ul&gt;</pre>

```

Las plantillas

```
<h3> No existen encuestas para el Docente el presente Periodo.</h3>
</ul>
{%
  endfor %
}
<!--/h3-->
<!--/ul-->
<hr/>
{%
  endif %
}

<!-- Fin Docente -->

<!-- Par Academico -->

{%
  if request.session.docente.parAcademico %
}

<h1 style="margin-top: 20px;"> Encuestas para Pares Académicos de la Carrera </h1>
<ul style="margin-top: 20px;">
{%
  if not request.session.cuestionarios_pares_academicos %
    <h3> No existen encuestas para Pares Académicos en el presente Periodo.</h3>
  {%
    else %
  {%
    for carrera in request.session.carreras_pares_academicos %
  <h3>
    <li>
      <a href="#"% url 'pares_academicos_docentes' carrera.num_carrera %>
        SELECCIONAR DOCENTES DE LA CARRERA {{ carrera.nombre|upper }} [{{ carrera.area|upper }}]
      </a>
    </li>
  </h3>
{%
  endfor %
{%
  endif %
</ul>
<hr/>
{%
  endif %
}
<!-- Fin Par Academico -->

<!-- Evaluaciones Director de Carrera -->

{%
  if request.session.docente.direcciones.count > 0 %
}

<h1 style="margin-top: 20px;"> Encuestas para Coordinadores Carrera </h1>
<ul style="margin-top: 20px;">
{%
  if not request.session.cuestionarios_directivos %
    <h3> No existen encuestas para los Coordinadores en el presente Periodo.</h3>
  {%
    else %
  {%
    for carrera in request.session.carreras_director %
  <h3>
    <li>
      <a href="#"% url 'director_docentes' carrera.num_carrera %>
        SELECCIONAR DOCENTES DE LA CARRERA {{ carrera.nombre|upper }} [{{ carrera.area|upper }}]
      </a>
    </li>
  </h3>
{%
  endfor %
{%
  endif %
</ul>
<hr/>

<!-- Fin Evaluaciones Director de Carrera -->
```

Las plantillas

```
<!-- Resultados -->

<h1 style="margin-top: 20px;"> Resultados de Evaluaciones de la Carrera </h1>
<ul style="margin-top: 20px;">
    {%
        for carrera in request.session.carreras_director %
    <h3>
        <li>
            <a href="/resultados/carrera/{{ carrera.num_carrera }}/">
                RESULTADOS DE EVALUACIONES CARRERA {{ carrera.nombre|upper }} [{{ carrera.area|upper }}]
            </a>
        </li>
    </h3>
    {%
        endfor %
    </ul>
    <hr/>

    {%
        endif %
    }

<!-- Fin Resultados -->

{%
    endblock %
}
```

Login

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/login.html:

```
{% extends "admin/login.html" %}

{% block title %} Evaluación Docente  {% endblock %}

{% csrf_token %}
{% if next %}
    <input type="hidden" name="next" value="{{ next }}" />
{% endif %}

{% block branding %}
    <h1 id="site-name"> Sistema de Evaluación de Docentes UNL </h1>
{% endblock %}

{% block content %}
    {{ form.non_field_errors }}
    {{ block.super }}
{% endblock %}
```

Carreras

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/carreras.html:

```
{% extends "admin/base_site.html" %}
{# extends "base.html" #}

<!--link rel="stylesheet" type="text/css" href="/static/admin/css/base.css" /-->

{% block content_title %}
    <h3> Carreras </h3>
{% endblock %}

{% block content %}
```

```
<p/>
<ul>
{%
  for carrera in request.session.carreras %
}

<h1> <li> <a href="/carrera/asignaturas/{{carrera.num_carrera}}"> {{ carrera.nombre }} </a>

{%
  endfor %
}</ul>

{%
  endblock %
}
```

Asignaturas y Docentes

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/asignaturas_docentes.html:

```
{% extends "admin/base_site.html" %}
{#% extends "base.html" %#}

{% block content_title %}
<h1>Seleccione el Docente a Evaluar </h1>
{% endblock %}

{% block content %}


| <b> {{ad.asignatura.nombre}} </b> | <b> {% for d in ad.docentes %} <ul class="list-group" style="list-style-type: none"> <li> {% if d.evaluado %} <a href="{% url 'encuestas' id_docente=d.docente.id id_asignatura=ad.asignatura.id %}"> {{d.docente}}</a> <br/> {% else %} <a href="#"> {{d.docente}}</a> <br/> </li> </ul> </b> |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

```

        </td>
    </tr>
</tbody>
</table>
```

```
{% endblock %}
```

Directores y Docentes

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/director_docentes.html:

```

{% extends "admin/base_site.html" %}
{#% extends "base.html" %#}

{% block content_title %}
<h1>Seleccione el Docente a Evaluar </h1>
{% endblock %}

{% block content %}


|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {% for de in docentes_evaluaciones %}         <b>             <ul class="{% if de.evaluado %} messagelist {% endif %}">                 <li>                     {% if de.evaluado %}                     {{ de.docente }}                     {% else %}                     <a href="{% url 'encuestas' id_docente=de.docente.id id_asignatura=0 id_tinfor                         {{ de.docente }}"                     </a>                     {% endif %}                 </li>             </ul>         </b>         <!-- Paginador de docentes -->         {% if forloop.counter divisibleby:"10" %}         </td>         <td>             {% endif %}             {% endfor %}         </td>     </tr>     <tr>         <td>         </td>         <td align="right">             <a href="{% url 'proyecto.app.controllers.index' %}"> <u><<< Carreras >>> </u></a>         </td>     </tr> </tbody> </table>  {% endblock %} |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

Pares Académicos y Docentes

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/pares_academicos_docentes.html:

```
% extends "admin/base_site.html" %

{% block content_title %}
<h1>Seleccione el Docente a Evaluar </h1>
{% endblock %}

{% block content %}


|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| {% for de in docentes_evaluaciones %}         <b>           <ul class="list-group" style="list-style-type: none; padding-left: 0;">             {% if de.evaluado %}               <li>                 {% if de.evaluado %}                   {{ de.docente }}                 {% else %}                   <a href="{% url 'encuestas' id_docente=de.docente.id id_asignatura=0 id_tinfor=%}                     {{ de.docente }}                   </a>                 {% endif %}               </li>             {% endif %}           </ul>         </b>         <!-- Paganador de docentes -->         {% if forloop.counter divisibleby:"10" %}         </td>         <td>           {% endif %}           {% endfor %}         </td> |                                                                                                                           |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <a href="{% url 'proyecto.app.controllers.index' %}">&lt;u&gt;&lt;&lt;&lt; Carreras &gt;&gt;&gt; &lt;/u&gt;&lt;/a&gt;</a> |


{% endblock %}
```

en construccion

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/en_construccion.html:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title> En construcción </title>
    <style type="text/css">

      body {
        font-family: Helvetica, Arial, Sans;
```

```

    }

    </style>
</head>
<body>
<center>
    <center><h2> En Construcción ... </h2> </center>
    <div id="pie" style="font-size: 9px; color: #666666">
        <hr style="margin-bottom: 6px;" />
        Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de So
        Correo Sistema de Evaluacion:
        <a href="mailto:evaluacion.docente@unl.edu.ec" > <u>evaluacion.docente@unl.ed
        Loja - Ecuador
    </div>

    </center>
</body>
</html>

```

Encuestas

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/encuestas.html:

```

{% extends "admin/base_site.html" %}

{% block content_title %}
<h1> Encuestas Disponibles </h1>
{% endblock %}

{% block content %}

{% if periodo_no_iniciado %}

<ul class="messagelist" style="margin-top: 20px;">
    <b><li class="warning"> El Periodo de Evaluacion aun no inicia </li></b>
</ul>

{% else %}
{% if periodo_finalizado %}

<ul class="messagelist" style="margin-top: 20px;">
    <b><li class="error"> Lo sentimos, el Periodo de Evaluacion ha Finalizado </li></b>
</ul>

{% else %}

<ul>
    {% for cuestionario in cuestionarios %}
        <h3>
            <li>
                <a href="/encuesta/responder/{{ cuestionario.id }}"> {{ cuestionario.titulo }} </a>
            </li>
        </h3>
        {% empty %}
        <h3> No existen Encuestas Disponibles </h3>
        {% endfor %}
    </ul>

    {% endif %}
    {% endif %}

```

```
{% endblock %}
```

Encuesta para Responder

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/encuesta_responder.html:

```
{% extends "admin/base_site.html" %}

{% load i18n %}

{% block extrahead %}

{{ block.super }}

<style type="text/css">
#encabezado {
    margin-top: 20px;
    margin-bottom: 20px;
    align: center;
}

.seccion{
    /*background-color: #EDF3FE;*/
    background-color: #B4CBE5;
}

.td-item-seleccion{
    border: 2px white solid;
    background-color: #B4CBE5;
    text-align: center;
    display: table-cell;
    /* Para cortar las palabras al ancho de la celda */
    word-break: break-all;

}

.td-tabla-contenedor {
    border-bottom: 1px dotted #B4CBE5;
    border-left: 1px dashed #B4CBE5;
}

#tabla-items{
    width: 100%;
}
</style>

<script type="text/javascript" src="/static/js/jquery-1.6.2.min.js"> </script>

<script type="text/javascript">

function validarSeleccionUnica(){
    var preguntas = new Array();
    // Se validan como obligatorias todas las preguntas de Selección Unica
    $.each($('#formulario input[name^="pregunta"]:radio'), function(){
        var nombre = $(this).attr("name");
        preguntas[nombre] = nombre;
    });
    for (var pregunta in preguntas){
```

Las plantillas

```
if (! $('#formulario input[name=' + pregunta +']:radio').is(':checked') ){
    alert('ERROR: Faltan preguntas por contestar');
    return false;
}

}
return true;
};

function validarEnsayos(){
var preguntas = new Array();
// Se validan como obligatorias todas las preguntas de Ensayo
$.each($('#formulario textarea[name^="pregunta"]'), function(){
    var nombre = $(this).attr("name");
    preguntas[nombre] = nombre;
});
for (var pregunta in preguntas){
    if ( $('#formulario textarea[name=' + pregunta +']').val().length < 4 ){
        alert('ERROR: Faltan preguntas por completar');
        return false;
    }
}
return true;
};

function validarObservaciones(){
/* Validacion en Evaluacion de Actividades Adicionales a la Docencia 2011-2012 */
var flag = true;
$.each($('#formulario input[name^="pregunta"]:radio:checked'), function(){
    var p = $(this).attr("name");
    var r = $(this).attr("value");
    var np = p.split('-')[1]; //Numero de la pregunta
    var o = 'observaciones-pregunta-' + np;
    var observaciones = $('textarea[name='+o+']').val();
    if (r != '1' && observaciones == ''){
        alert('Faltan ingresar fuentes de verificación')
        flag=false;
        return flag;
    }
});
return flag;
}

function grabarEncuesta(){
/* Se trata de la Evaluacion de Actividades Adicionales a la Docencia 2011-2012 */
{%
  ifequal cuestionario.periodoEvaluacion.id 2 %}
if (validarObservaciones()){
    $('#formulario').submit();
} else {
    return false;
}
{%
  endifequal %}
/* Para el resto de los casos de Periodos de Evaluacion */
$('#formulario').submit();

/*$.ajax({
    url: "/encuesta/grabar/",
    method: "POST",
    //data: "periodo_academico_id={{ object_id }}",
    */
}
```

```

        success: function (data){
            alert(data);
        },
        error: function(xhr,status,error){
            alert("error: " + status + "-" + error);
        }
    });
}

$(document).ready(function(){
    $('#grabar').click(function(){
        {% if cuestionario.preguntas_obligatorias %}
        if (validarSeleccionUnica() == true && validarEnsayos() == true){
            grabarEncuesta();
        }
        {% else %}
            grabarEncuesta();
        {% endif %}
    });
});

</script>

{% endblock %}

{% block content_title %}

<h3> {{ cuestionario.titulo }} </h3>

{% endblock %}

{% block content %}

<form action="/encuesta/grabar/" method="post" id="formulario" name="formulario" >

    {% csrf_token %}

    <table id="tabla-contenedor">

        <td colspan="2">
            <div id="encabezado-cuestionario" width="25%">
                {{ cuestionario.encabezado|safe }}
            </div>

        <!-- Encuesta dirigida a los ESTUDIANTES -->
        {## if cuestionario.informante.tipo.startswith('Estudiante') ##}
        {% if asignaturaDocente %}
            <div id="datos_academicos" style="line-height: 18px;">
                Area: <b> {{ asignaturaDocente.asignatura.area }} </b> <br/>
                Carrera: <b> {{ asignaturaDocente.asignatura.carrera }} </b> <br/>
                Modulo ( {% ifequal asignaturaDocente.asignatura.tipo "modulo" %} <b>X</b> {% en
                Unidad ( {% ifequal asignaturaDocente.asignatura.tipo "unidad" %} <b>X</b> {% en
                Curso ( {% ifequal asignaturaDocente.asignatura.tipo "curso" %} <b>X</b> {% en
                Taller ( {% ifequal asignaturaDocente.asignatura.tipo "taller" %} <b>X</b> {% en
                Otro ( {% ifequal asignaturaDocente.asignatura.tipo "otro" %} <b>X</b> {% endif
                <br/>
                Nombre de el/la Docente Evaluado(a): <b> {{ asignaturaDocente.docente }} </b> <br/>
                Fecha de Evaluacion: <b> {{ fecha }} </b> <br/>
            </div>
        {% endif %}
    </td>
</table>

```

```

</div>
{%
  endif %}

<!-- Encuesta dirigida a los DIRECTORES DE CARRERA -->
{%
  if 'Directivo' in cuestionario.informante.tipo %}
<div id="datos_academicos" style="line-height: 18px;">
  AREA: <b> {{ request.session.area }} </b> <br/>
  CARRERA: <b> {{ request.session.carrera }} </b> <br/>
  MÓDULO: (<b>X</b>) <br/>
  NOMBRE DEL PROFESOR (A) EVALUADO (A): <b> {{ request.session.docente_evaluar }} </b>
</div>
{%
  endif %

<!-- Encuesta dirigida a los PARES ACADEMICOS -->
{%
  if 'ParAcademico' in cuestionario.informante.tipo %}
<div id="datos_academicos" style="line-height: 18px;">
  AREA: <b> {{ request.session.area }} </b> <br/>
  CARRERA: <b> {{ request.session.carrera }} </b> <br/>
  MÓDULO: (<b>X</b>) <br/>
  NOMBRE DEL PROFESOR (A) EVALUADO (A): <b> {{ request.session.docente_evaluar }} </b>
</div>
{%
  endif %

<!-- Encuesta de autoevaluacion para los DOCENTES -->
{%
  if cuestionario.informante.tipo == 'Docente' %}
<div id="datos_academicos" style="line-height: 18px;">
  AREA: <b> {{ request.session.areas_docente }} </b> <br/>
  CARRERA: <b> {{ request.session.carreras_docente }} </b> <br/>
  MÓDULO: (<b>X</b>) <br/>
  NOMBRE DEL PROFESOR (A) EVALUADO (A): <b> {{ request.session.docente }} </b>
</div>
{%
  endif %

</td>
<!-- end datos academicos -->

<tbody>

{%
  for seccion in cuestionario.secciones.all %} <!-- secciones de cuestionario -->
<tr>
  <td class="seccion" colspan="3">
    <b> {{ seccion.orden }}. {{ seccion.titulo|upper }} </b>
    {{ seccion.descripcion }}
  </td>
</tr>

<!-- ===== Tratamiento de preguntas de SUBSECCIONES ===== -->

{%
  for subseccion in seccion.subsecciones.all %}
<tr>
  <!-- columna subseccion -->
  <td class="td-tabla-contenedor" width="10%" rowspan="{{ subseccion.preguntas }}%
    {%
      if subseccion.codigo %} {{ subseccion.codigo }} {%
      else %} {{ subseccion.codigo }} {%
    }
  </td>
  <!--/columna subseccion -->

  <!-- preguntas de subsecciones -->
  {%
    for pregunta in subseccion.preguntas.all %}
    <!-- columna pregunta -->
    <td class="td-tabla-contenedor" width="60%" colspan="{{ ifequal pregunta.tipo 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 19 20 20 21 21 22 22 23 23 24 24 25 25 26 26 27 27 28 28 29 29 30 30 31 31 32 32 33 33 34 34 35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43 44 44 45 45 46 46 47 47 48 48 49 49 50 50 51 51 52 52 53 53 54 54 55 55 56 56 57 57 58 58 59 59 60 60 61 61 62 62 63 63 64 64 65 65 66 66 67 67 68 68 69 69 70 70 71 71 72 72 73 73 74 74 75 75 76 76 77 77 78 78 79 79 80 80 81 81 82 82 83 83 84 84 85 85 86 86 87 87 88 88 89 89 90 90 91 91 92 92 93 93 94 94 95 95 96 96 97 97 98 98 99 99 100 100 101 101 102 102 103 103 104 104 105 105 106 106 107 107 108 108 109 109 110 110 111 111 112 112 113 113 114 114 115 115 116 116 117 117 118 118 119 119 120 120 121 121 122 122 123 123 124 124 125 125 126 126 127 127 128 128 129 129 130 130 131 131 132 132 133 133 134 134 135 135 136 136 137 137 138 138 139 139 140 140 141 141 142 142 143 143 144 144 145 145 146 146 147 147 148 148 149 149 150 150 151 151 152 152 153 153 154 154 155 155 156 156 157 157 158 158 159 159 160 160 161 161 162 162 163 163 164 164 165 165 166 166 167 167 168 168 169 169 170 170 171 171 172 172 173 173 174 174 175 175 176 176 177 177 178 178 179 179 180 180 181 181 182 182 183 183 184 184 185 185 186 186 187 187 188 188 189 189 190 190 191 191 192 192 193 193 194 194 195 195 196 196 197 197 198 198 199 199 200 200 201 201 202 202 203 203 204 204 205 205 206 206 207 207 208 208 209 209 210 210 211 211 212 212 213 213 214 214 215 215 216 216 217 217 218 218 219 219 220 220 221 221 222 222 223 223 224 224 225 225 226 226 227 227 228 228 229 229 230 230 231 231 232 232 233 233 234 234 235 235 236 236 237 237 238 238 239 239 240 240 241 241 242 242 243 243 244 244 245 245 246 246 247 247 248 248 249 249 250 250 251 251 252 252 253 253 254 254 255 255 256 256 257 257 258 258 259 259 260 260 261 261 262 262 263 263 264 264 265 265 266 266 267 267 268 268 269 269 270 270 271 271 272 272 273 273 274 274 275 275 276 276 277 277 278 278 279 279 280 280 281 281 282 282 283 283 284 284 285 285 286 286 287 287 288 288 289 289 290 290 291 291 292 292 293 293 294 294 295 295 296 296 297 297 298 298 299 299 300 300 301 301 302 302 303 303 304 304 305 305 306 306 307 307 308 308 309 309 310 310 311 311 312 312 313 313 314 314 315 315 316 316 317 317 318 318 319 319 320 320 321 321 322 322 323 323 324 324 325 325 326 326 327 327 328 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 
```

```

<b>
    { % if pregunta.codigo %} {{ pregunta.codigo }}. { % else %} {{ pregunta.codigo }} {{ pregunta.descripcion|safe }}
</b>
<br/>
{{ pregunta.descripcion|safe }} <!-- Puede contener HTML -->
{ % ifequal pregunta.tipo.tipo 'Ensayo' %}
<br/>
<textarea colspan="2" name="pregunta-{{ pregunta.id }}" cols="100" rows="10" style="width: 100%; height: 100%;">
{ % endifequal %}
</td>
<!-- preguntas de subsecciones -->

<!-- items de pregunta si es de seleccion -->
{ % ifequal pregunta.tipo.tipo 'SeleccionUnica' %}
<td class="td-tabla-contenedor" width="30%">
    <table id="tabla-items">
        <tbody>
            <tr>
                { % for item in pregunta.items.all %}
                <!--Dividir para el numero exacto de items de la pregunta Ejemplo: si hay 3 items se dividen en 3 filas de 1 columna cada una -->
                <td class="td-item-seleccion" { % if not item.descripcion %}>
                    { % if not pregunta.observaciones %} style="width:25%;" { % else %} style="width:75%;" { % endif %}
                    <input type="radio" name="pregunta-{{ pregunta.id }}" value={{ item.id }} checked="checked" { % endif %}
                    <b> {{ item.texto }} </b>
                    <br/>
                    { % if item.descripcion %} {{ item.descripcion }} { % endif %}
                </td>
                { % endfor %}

                { % if pregunta.observaciones %}
                <td id="observaciones">
                    {{ pregunta.observaciones|safe }}:
                    <textarea name="observaciones-pregunta-{{ pregunta.id }}" style="width: 100%; height: 100%; border: none; border-radius: 0; padding: 0; margin: 0; font-size: inherit; color: inherit; background-color: transparent; outline: none; resize: none;">
        { % if cuestionario.secciones.all|length > 1 %}
        <b>
            { % if pregunta.codigo %} {{ pregunta.codigo }}. { % else %} {{ pregunta.codigo }} {{ pregunta.descripcion|safe }} { % endif %}
        </b>
        { % else %}
    
```

```

<b>
    { % if pregunta.codigo %} {{ pregunta.codigo }}. { % else %} {{ pregunta.codigo }}
</b>
{{ pregunta.texto|safe }}
{ % endif %}
<br/>
{ % if pregunta.descripcion %}
{{ pregunta.descripcion|safe }} <!-- Puede contener HTML -->
{ % endif %}
{ % ifequal pregunta.tipo.tipo 'Ensayo' %}
<br/>
<textarea name="pregunta-{{ pregunta.id }}" cols="100" rows="10" style="width:100%;height:100px;">
{ % endifequal %}
</td>

<!-- items pregunta de seccion si es de seleccion -->
{ % ifequal pregunta.tipo.tipo 'SeleccionUnica' %}
<td class="td-tabla-contenedor" width="40%">
    <table id="tabla-items">
        <tbody>
            <tr>
                { % for item in pregunta.items.all %}
                <!--TODO: Dividir para el numero exacto de items de la pregunta -->
                <td class="td-item-seleccion" { % if not item.descripcion %} style="width:25%;" { % else %} style="width:75%;" { % endif %}
                    { % if not pregunta.observaciones %} style="width:25%;" { % else %} style="width:75%;" { % endif %}
                    <input type="radio" name="pregunta-{{ pregunta.id }}" value="{{ item.id }}">
                    <b> {{ item.texto }} </b>
                    <br/>
                    { % if item.descripcion %} {{ item.descripcion }} { % endif %}
                </td>
                { % endfor%}

                { % if pregunta.observaciones %}
                <td id="observaciones">
                    {{ pregunta.observaciones|safe }}:
                    <textarea name="observaciones-pregunta-{{ pregunta.id }}" cols="100" rows="10" style="width:100%;height:100px;">
                </td>
                { % endif %}
            </tr>
        </tbody>
    </table>
</td>
{ % endifequal %}
<!--/ items pregunta de seccion si es de seleccion -->
</tr>
{ % endfor %} <!--/ preguntas de seccion -->

{ % endfor %} <!--/ secciones de cuestionario -->

</tbody>

<tfoot>
    <div id="comandos">
        <tr>
            <td colspan="2">
                <input type="button" id="grabar" name="grabar" value="GRABAR"/>
                <input type="reset" value="BORRAR"/>
                <input type="button" id="cancelar" name="cancelar" value="CANCELAR"/>
            </td>
        </tr>
    </div>
</tfoot>

```

Las plantillas

```
</div>
</tfoot>

</table>

</form>

{ % endblock % }
```

Encuesta Finalizada

Ubicación: dentro de {{TEMPLATE_DIRS}}/app/encuesta_finalizada.html:

```
{% extends "admin/base_site.html"%}

{% block title %} Encuesta Grabada  {% endblock %}

{% block content %}

<h1> Información grabada satisfactoriamente. </h1>

<p> Gracias por participar de la Encuesta. </p>

{% if request.session.estudianteAsignaturaDocente %}

<p>
  <a href="#"{{ url 'proyecto.app.controllers.estudiante_asignaturas_docentes' num_carrera }}>
    <<< Continuar >>>
  </a>
</p>
{% else %}
  {% if request.session.director_carrera and num_carrera %}

<p>
  <a href="#"{{ url 'proyecto.app.controllers.director_docentes' num_carrera }}>
    <!--a href="#"{{ url encuestas id_docente=request.session.docente_evaluar.id id_asignatura }}>
    <<< Continuar >>>
  </a>
</p>
  {% else %}
    {% if request.session.docente %}

<p>
  <a href="#"{{ url 'proyecto.app.controllers.index' }}>
    <<< Continuar >>>
  </a>
</p>
  {% endif %}
  {% endif %}
  {% endif %}
  {% endif %}

  {% endblock %}
```

Resultados eaad 2012

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/imprimir_resultados_eaad2012.html:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title> Resultados Evaluación de Actividades Adicionales a la Docencia 2011 - 2012 </title>
    <style type="text/css">
      body {
        font-family: Helvetica, Arial, Sans;
```

```

}
table {
    margin-top: 10px;
    margin-bottom: 10px;
    width: 75%;
    border-spacing: 0;
}
th {
    text-align: center;
    border: 1px solid gray;
    font-size: 13px;
}
td {
    text-align: left;
    border: 1px solid gray;
    font-size: 13px;
}
textarea {
    font-size: 10px;
    overflow: hidden;
}

</style>
</head>
<body>
<center>
    
    <h4>
        {{ "Universidad Nacional de Loja" | upper }} <br/>
        Comisión de Evaluación Interna <br/>
        EVALUACIÓN DEL DESEMPEÑO ACADÉMICO <p/>
        {{ "Resultados de la Evaluación de Actividades Adicionales a la Docencia 2011-2012" | upper }}
    </h4>
    <!-- Tabla de encabezado -->

    <table id="tabla-docente" width="100%" border="0">
        <tr>
            <td width="10%"><b>AREA</b></td> <td width="90%">{{ area }}</td>
        </tr>
        <tr>
            <td width="10%"><b>CARRERA</b></td> <td width="90%">{{ carrera }}</td>
        </tr>
        <tr>
            <td width="10%"><b>DOCENTE</b></td> <td width="90%">{{ docente }}</td>
        </tr>
    </table>

    <!-- Tabla Central -->

    <table id="tabla-resultados" width="100%" cellspacing="0" >
        <th rowspan="3" colspan="2" style="width: 80%;">ACIVIDADES </th>
        <th colspan="4">INFORMANTES </th>
        <th rowspan="3">Fuente de Verificación </th>
        <tr>
            <th colspan="2">COMISION </th>
            <th colspan="2">DOCENTE </th>
        </tr>
        <tr>
            <th width="4%"> Esc. </th>
            <th width="4%"> % </th>
        </tr>
    </table>

```

```

<th width="4%"> Esc. </th>
<th width="4%"> % </th>
</tr>
{%
  for seccion in contestaciones.secciones %
}
<tr>
  <td colspan="7"> <b> {{ seccion.titulo|upper }} </b> </td>
</tr>
{%
  for r in seccion.resultados %
}
<tr>
  <td width="4%"> {{ r.codigo }} </td>
  <td width="70%" style=""> {{ r.texto }} </td>
  <td style="text-align: center;"> {{ r.valor_comision }} </td>
  <td style="text-align: center;"> {{%if r.porcentaje_comision%} {{ r.porcentaje_comision }} %}
  <td style="text-align: center;"> {{ r.valor_docente }} </td>
  <td style="text-align: center;"> {{%if r.porcentaje_docente%} {{ r.porcentaje_docente }} %}
  <td>
    <textarea readonly="readonly" rows="4">COMISION: {{ r.observaciones_comision }} </td>
  </td>
</tr>
{%
  endfor %
}
{%
  endfor %
}
</table>

<!-- Tabla de Resultados -->

<table width="100%">
  <tr>
    <td width="70%"> <h3> Número de Actividades: </h3></td>
    <td colspan="2" style="text-align: center;"> <h3> {{ contestaciones.num_actividades }} </h3> </td>
  </tr>
  <tr>
    <td rowspan="2" style="vertical-align: middle; text-align: center;"> <h3> Valoración Global Actividades Adicionales: </h3></td>
    <td style="text-align: center;"> <h3> Comision: {{ porcentaje1|floatformat:"2" }} % </h3> </td>
    <td style="text-align: center;"> <h3> Docente: {{ porcentaje2|floatformat:"2" }} % </h3> </td>
  </tr>
  <tr>
    <td colspan="2" style="text-align: center; vertical-align: middle;"> <h1> {{ total }} </h1> </td>
  </tr>
</table>

<div id="pie" style="font-size: 9px; color: #666666">
  <hr style="margin-bottom: 6px;" />
  Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de Software <br>
  Correo Sistema de Evaluacion:
  <a href="mailto:evaluacion.docente@unl.edu.ec"> <u>evaluacion.docente@unl.edu.ec</u> </a>
  Loja - Ecuador
</div>

</center>
</body>
</html>

```

Resultados Satisfacción Estudiantil 2012

Plantilla para imprimir los resultados en la encuesta de Satisfacción Estudiantil 2012. Ubicación: {{ TEMPLATE_DIRS }} /app/imprimir_resultados_ese2012.html:

```

<html>
  <head>

```

Las plantillas

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title> Resultados Encuesta de Satisfacción Estudiantil 2012 </title>
```

Estilos:

```
<style type="text/css">
body {
    font-family: Helvetica, Arial, Sans;
}
table {
    margin-top: 10px;
    margin-bottom: 10px;
    width: 75%;
    border-spacing: 0;
}
td, th {
    text-align: center;
    border: 1px solid gray;
    font-size: 13px;
    width: 20%;
}
</style>
</head>
```

Inicio del body:

```
<body>
<center>

<h4> {{ "Universidad Nacional de Loja" |upper }} </h4>
<h3 style="margin-bottom: 3px;"> Encuesta de Satisfacción Estudiantil 2012 </h3>
{{ titulo |safe }}
<table width="100%" cellspacing="0">
<th rowspan="2"> Indicadores </th>
<th colspan="4"> Niveles de Satisfacción </th>
<tr>
<th> Muy Satisfecho MS (4) </th>
<th> Satisfecho S (3) </th>
<th> Poco Satisfecho PS (2) </th>
<th> Insatisfecho INS (1) </th>
</tr>
{%
    for c in conteos %
}
<tr>
<td title="{{ c.pregunta.texto }}" width="20%">
<b>{{ c.pregunta.seccion.orden }}.{{c.pregunta.orden}} </b>
</td>
<td> {{ c.MS }} </td>
<td> {{ c.S }} </td>
<td> {{ c.PS }} </td>
<td> {{ c.INS }} </td>
</tr>
{%
    endfor %
}
</table>

{%
    if totales and porcentajes %
}
<h3> Resultados </h3>
<table cellspacing="0">
<tr>
<td> Suman ( {{ totales.total }} ) </td>
<td> {{ totales.MS }} </td>
<td> {{ totales.S }} </td>
```

Las plantillas

```
<td> {{ totales.PS }} </td>
<td> {{ totales.INS }} </td>
</tr>
<tr>
    <td> % de Estudiantes que declaran su Satisfacción </td>
    <td> {{ porcentajes.MS|floatformat:"-2" }}% </td>
    <td> {{ porcentajes.S|floatformat:"-2" }}% </td>
    <td> {{ porcentajes.PS|floatformat:"-2" }}% </td>
    <td> {{ porcentajes.INS|floatformat:"-2" }}% </td>
</tr>
<tr>
    <td> % de Estudiantes que se declaran Satisfechos </td>
    <td colspan="2"> {{ porcentajes.MSS|floatformat:"-2" }} % </td>
    <!--td colspan="2"> {{ porcentajes.MS|add:porcentajes.S }}% </td-->
    <td colspan="2"> </td>
</tr>
</table>
{%endif%}
```

Sección final:

```
<div id="pie" style="font-size: 9px; ; color: #666666">
    <hr style="margin-bottom: 6px;" />
    Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de Software <br>
    Correo Sistema de Evaluacion:
    <a href="mailto:evaluacion.docente@unl.edu.ec"> <u>evaluacion.docente@unl.edu.ec</u>
    Loja - Ecuador
</div>

</center>
</body>
</html>
```

Otros resultados Satisfacción Estudiantil 2012

Plantilla para imprimir otros resultados en la encuesta de Satisfacción Estudiantil 2012. Ubicación: {{TEMPLATE_DIRS}}/app/imprimir_otros_ese2012.html:

```
<html>
    <head>
```

Encabezados y estilos:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title> Resultados Encuesta de Satisfacción Estudiantil 2012 </title>
<style type="text/css">
body {
    font-family: Helvetica, Arial, Sans;
}
table {
    margin-top: 10px;
    margin-bottom: 10px;
    width: 75%;
    border-spacing: 0;
}
td, th {
    border: 1px solid gray;
    font-size: 13px;
}
</style>
</head>
```

Las plantillas

Inicio del body:

```
<body>
<center>

<h4> {{ "Universidad Nacional de Loja" | upper }} </h4>
<h3 style="margin-bottom: 3px;"> Encuesta de Satisfacción Estudiantil 2012 </h3>
{{ titulo | safe }}





```

Pie de página:

```
<div id="pie" style="font-size: 9px; ; color: #666666">
<hr style="margin-bottom: 6px;" />
    Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de Software <br/>
    Correo Sistema de Evaluacion:
    <a href="mailto:evaluacion.docente@unl.edu.ec"> <u>evaluacion.docente@unl.edu.ec</u>
        Loja - Ecuador
    </div>
</center>
</body>
</html>
```

Resultados Desempeño Docente 2013

Ubicación: dentro de {{ TEMPLATE_DIRS }} /app/imprimir_resultados_edd2013.html:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title> Resultados Evaluación del Desempeño Académico 2012 - 2013 </title>
```

Sección de estilos:

```
<style type="text/css">
body {
    font-family: Helvetica, Arial, Sans;
}
table {
    margin-top: 10px;
    margin-bottom: 10px;
    width: 85%;
    border-spacing: 0;
}
th {
    text-align: center;
    border: 1px solid gray;
    font-size: 13px;
}
td {
    text-align: center;
    border: 1px solid gray;
```

Las plantillas

```
    font-size: 13px;
}
textarea {
    font-size: 10px;
    overflow: hidden;
}
.promedio_componente td {
    text-decoration: overline;
    /*font-size: 0%; */
}
</style>
</head>
```

Inicio del body:

```
<body>
<center>

<h4>
{{ "Universidad Nacional de Loja" | upper }} <br/>
Comisión de Evaluación Interna <br/>
EVALUACIÓN DEL DESEMPEÑO ACADÉMICO <p/>
{{ "Resultados de la Evaluación del Desempeño Docente 2012" | upper }}
```

Tabla de encabezado:

```
<table id="tabla-docente" border="0">
{%
  if area %
  <tr>
    <td width="10%"><b>AREA</b></td> <td style="text-align: left;" width="90%">{{ area
  </tr>
{%
  endif %
{%
  if carrera %
  <tr>
    <td width="10%"><b>CARRERA</b></td> <td style="text-align: left;" width="90%">{{ carrera
  </tr>
{%
  endif %
{%
  if docente %
  <tr>
    <td width="10%"><b>DOCENTE</b></td> <td style="text-align: left;" width="90%">{{ docente
  </tr>
{%
  endif %
</table>
```

Si se filtra por Componente:

```
{%
  if seccion_componente %
  <table>
    <tr>
      <td style="text-align: left;" width="40%">
        <h2> {{ seccion_componente.titulo }} :</h2>
      </td>
      <td style="text-align: left;">
        <h3> {{ seccion_componente.descripcion }} </h3>
      </td>
    </tr>
  </table>
{%
  endif %}
```

Tabla Central:

```
<table id="tabla-resultados" width="100%" cellspacing="0" >
  <tr>
```

Las plantillas

```
<th rowspan="3" colspan="2"> DESCRIPCIÓN DE LOS INDICADORES DE CALIDAD </th>
<th colspan="4"> INFORMANTES </th>
<th colspan="3"> EVALUACIÓN </th>
</tr>
<tr>
    <th rowspan="2"> ESTUDIANTES </th>
    <th rowspan="2"> DOCENTES </th>
    <th colspan="2"> COMISIÓN ACADÉMICA </th>
    <th rowspan="2"> PRIMARIA </th>
    <th rowspan="2"> PONDERAD. </th>
    <th rowspan="2"> CUALITAT. </th>
</tr>
<tr>
    <th> PAR ACAD. </th>
    <th> DIRECTIVO </th>
</tr>
```

Indicador, resultado:

```
{% for i, r in resultados_indicadores.items %}
```

Titulos de los componentes:

```
{# Titulos de los diferentes componentes o secciones #}
{% ifchanged r.objeto_seccion.superseccion.codigo %}
{# Para acortar el nombre del objeto #}
{% with superseccion=r.objeto_seccion.superseccion %}
<tr>
    <td colspan="9" style="text-align: left;">
        <b>
            COMPONENTE {{ superseccion.orden }}: {{ superseccion.titulo|upper }} ({{ supersecc...</b>
    </td>
</tr>
{% endwith %}
{% endifchanged %}
```

Cada uno de los indicadores:

```
<tr>
    <td style="text-align: left; font-size: 9px;">
        {{ r.objeto_seccion.descripcion }}
    </td>
    <td>
        {{ i }}
    </td>
    <td width="10%">
        {% if r.informantes.estudiante >= 0 %} {{ r.informantes.estudiante }} % {% else %} - {% endif %}
    </td>
    <td width="10%">
        {% if r.informantes.docente >= 0 %} {{ r.informantes.docente }} % {% else %} - {% endif %}
    </td>
    <td width="10%">
        {% if r.informantes.paracademico >= 0 %} {{ r.informantes.paracademico }} % {% else %} - {% endif %}
    </td>
    <td width="10%">
        {% if r.informantes.directivo >= 0 %} {{ r.informantes.directivo }} % {% else %} - {% endif %}
    </td>
    <td width="10%">
        {% if r.primaria >= 0 %} {{ r.primaria|floatformat:"0" }} % {% else %} - {% endif %}
    </td>
    <td width="10%"> {{ r.ponderada|floatformat:"2" }} <b>/</b> {{ r.ponderacion_seccion }} </td>
```

```
<td width="10%"> {{ r.cualitativa }} </td>
</tr>
```

Totales por Componente:

```
{% if promedios_componentes %} <!-- if promedios_componentes -->
{# Total CPF #}
{% ifequal forloop.counter promedios_componentes.CPF.fila %}
<tr class="promedio_componente">
    <td colspan="2" style="text-align: left;">
        TOTAL {{ r.objecto_seccion.superseccion.titulo }}
    </td>
    <td>
        {{ promedios_componentes.CPF.estudiante }} %
    </td>
    <td>
        {{ promedios_componentes.CPF.docente }} %
    </td>
    <td>
        {{ promedios_componentes.CPF.paracademico }} %
    </td>
    <td>
        {{ promedios_componentes.CPF.directivo }} %
    </td>
    <td>
        {{ promedios_componentes.CPF.primaria|floatformat:"0" }} %
    </td>
    <td>
        {{ promedios_componentes.CPF.ponderada|floatformat:"2" }} / {{ r.objecto_seccio
    <td>
        {{ promedios_componentes.CPF.cualitativa }}
    </td>
</tr>
{% endifequal %}

{# Total CPG #}
{% ifequal forloop.counter promedios_componentes.CPG.fila %}
<tr class="promedio_componente">
    <td colspan="2" style="text-align: left;">
        TOTAL {{ r.objecto_seccion.superseccion.titulo }}
    </td>
    <td>
        {{ promedios_componentes.CPG.estudiante }} %
    </td>
    <td>
        {{ promedios_componentes.CPG.docente }} %
    </td>
    <td>
        {{ promedios_componentes.CPG.paracademico }} %
    </td>
    <td>
        {{ promedios_componentes.CPG.directivo }} %
    </td>
    <td>
        {{ promedios_componentes.CPG.primaria|floatformat:"0" }} %
    </td>
    <td>
        {{ promedios_componentes.CPG.ponderada|floatformat:"2" }} / {{ r.objecto_seccio
    <td>
```

Las plantillas

```
        {{ promedios_componentes.CPG.cualitativa }}
```

```
</td>
</tr>
{%- endifequal %}

{# Total PV #}
{%- ifequal forloop.counter promedios_componentes.PV.fila %}
|  |  |
| --- | --- |
| TOTAL {{ r.objeto_seccion.superseccion.titulo }} | |
| {{ promedios_componentes.PV.estudiante }} % |
| {{ promedios_componentes.PV.docente }} % |
| {{ promedios_componentes.PV.paracademico }} % |
| {{ promedios_componentes.PV.directivo }} % |
| {{ promedios_componentes.PV.primaria|floatformat:"0" }} % |
| {{ promedios_componentes.PV.ponderada|floatformat:"2" }} / {{ r.objeto_seccion }} |
| {{ promedios_componentes.PV.cualitativa }} |


{%- endifequal %}
{%- endif %} <!-- end if promedios_componentes -->

{%- endfor %}

```

Tabla de Resultados:

```
<table>
|  |  |
| --- | --- |
| VALOR TOTAL OBTENIDO: | |
| {{ promedios.estudiante|floatformat:"2" }} % |
| {{ promedios.docente|floatformat:"2" }} % |
| {{ promedios.paracademico|floatformat:"2" }} % |
| {{ promedios.directivo|floatformat:"2" }} % |
| {{ promedios.primaria }} % |
| {{ promedios.ponderada|floatformat:"2" }} / 100 |
| {{ promedios.cualitativa }} % |

```

Tabla para escalas:

```
<table id="tabla-escala">
|  |
| --- |
| D = Destacado | |

```

Las plantillas

```
<b> S </b> = Satisfactorio |  
<b> PS </b> = Poco Satisfactorio |  
<b> IS </b> = Insatisfactorio  
</td>  
</tr>  
</table>
```

Sección final:

```
<a href='javascript:window.print();'>  
      
</a>  
<div id="pie" style="font-size: 9px; color: #666666">  
    <hr style="margin-bottom: 6px;" />  
    Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de Sistemas y Servicios  
    Correo Sistema de Evaluacion:  
    <a href="mailto:evaluacion.docente@unl.edu.ec" style="color: #666666; text-decoration: none; font-weight: bold;"><u>evaluacion.docente@unl.edu.ec</u></a>  
    Loja - Ecuador  
</div>  
</center>  
</body>  
</html>
```

Sugerencias Evaluación Desempeño Docente 2013

Plantilla para impresión de sugerencias en la Evaluación Desempeño Docente 2013. Ubicación: {{TEMPLATE_DIRS}}/app/imprimir_sugerencias_edd2013.html:

```
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
        <title> Resultados Evaluación del Desempeño Académico 2012 - 2013 </title>  
    </head>
```

Sección de estilos:

```
<style type="text/css">  
body {  
    font-family: Helvetica, Arial, Sans;  
}  
table {  
    margin-top: 10px;  
    margin-bottom: 10px;  
    width: 75%;  
    border-spacing: 0;  
}  
th {  
    text-align: center;  
    border: 1px solid gray;  
    font-size: 13px;  
}  
td {  
    text-align: left;  
    border: 1px solid gray;  
    font-size: 13px;  
}  
textare {  
    font-size: 10px;  
    overflow: hidden;  
}  
</style>  
</head>
```

Las plantillas

Inicio del body:

```
<body>
<center>

<h4>
{{ "Universidad Nacional de Loja" | upper }} <br/>
Comisión de Evaluación Interna <br/>
EVALUACIÓN DEL DESEMPEÑO ACADÉMICO <p/>
{{ "Resultados de la Evaluación del Desempeño Académico 2012-2013" | upper }}
```

Tabla de encabezado:

```
<table id="tabla-docente" width="100%" border="0">
{%
  if area %
}
<tr>
  <td width="10%"><b>AREA</b></td> <td style="text-align: left;" width="90%">{{ area | upper }}
</tr>
{%
  endif %
}
{%
  if carrera %
}
<tr>
  <td width="10%"><b>CARRERA</b></td> <td style="text-align: left;" width="90%">{{ carrera | upper }}
</tr>
{%
  endif %
}
</table>
```

Tabla Central:

```
<table id="tabla-resultados" width="100%" cellspacing="0" >
{%
  for docente, informantes in sugerencias.items %
}
<tr>
  <td colspan="3" > <h2> {{ docente }} </h2></td>
</tr>
{%
  for informante, resultados in informantes.items %
}
<tr>
  <td colspan="3" > <h3> <center> INFORMANTE {{ informante | upper }} </center> <h3>
    <td width="33%"> <b> CAPACIDAD PROFESIONAL </b> </td>
    <td width="33%"> <b> CAPACIDAD PEDAGÓGICA </b> </td>
    <td width="33%"> <b> PRÁCTICA DE VALORES </b> </td>
  </h3>
</tr>
{%
  for r in resultados %
}
<tr>
  <td width="33%"> {{ r.CPF }} </td>
  <td width="33%"> {{ r.CPG }} </td>
  <td width="33%"> {{ r.PV }} </td>
</tr>
{%
  endfor %
}
{%
  endfor %
}
{%
  endfor %
}
</table>
```

Sección final:

```
<a href='javascript:window.print();'>
  
  <div id="pie" style="font-size: 9px; color: #666666">
    <hr style="margin-bottom: 6px;"/>
    Unidad de Telecomunicaciones e Infomacion - Departamento de Desarrollo de Sistemas y Servicios
  </div>
</a>
```

```
        Correo Sistema de Evaluacion:  
        <a href="mailto:evaluacion.docente@unl.edu.ec"> <u>evaluacion.docente@unl.edu.ec</u>  
        Loja - Ecuador  
    </div>  
    </center>  
    </body>  
</html>
```

Menú Resultados Carreras

Permite realizar la presentación de el munú con opciones para presentar y administrar los resultados de las evaluaciones. Ubicación: {{TEMPLATE_DIRS}}/app/menu_resultados_carrera.html:

```
{% extends "admin/base_site.html" %}  
  
{% block extrahead %}
```

Estilos básicos:

```
<style>  
    #id_periodos {  
        width: 100%;  
        font-size: 15px;  
        font-weight: bold;  
        color: #666666;  
    }  
</style>  
  
<script type="text/javascript" src="/static/js/jquery-1.6.2.min.js"></script>
```

Método AJAX para la obtención de datos:

```
<script type="text/javascript">  
    $(document).ready(function(){  
        $("#procesar").hide();  
        $("#id_periodo_evaluacion").change(function(){  
            var id_periodo = $("#id_periodo_evaluacion option:selected").val();  
        })  
    })
```

Si existe un periodo académico entonces se genera el menú con las distintas opciones disponibles:

```
if (id_periodo != ""){  
    $.ajax({  
        url: "/resultados/periodo/" + id_periodo + " / ",  
        method: "GET",  
        //data: "periodo_academico_id={{ object_id }}",  
        success: function (data){  
            $("#menu-resultados").html(data);  
            $("#procesar").show();  
            $("#borrar").show();  
        },  
        error: function(xhr,status,error){  
            alert("error: " + status + " - " + error);  
        }  
    });  
}
```

Caso contrario no presentar nada:

```
} else{  
    $("#menu-resultados").html("");  
    $("#procesar").hide();  
    $("#borrar").hide();  
}
```

Fín del script:

Las plantillas

```
    });
}
</script>
```

Bloques heredados de la plantilla Base:

```
{% block.super %}

//Cierre del bloque 'extrahead'

{% endblock %}

{% block content_title %}

<!--h1> Resultados de la Carrera </h1-->

{% endblock %}
```

Despliegue del formulario con una estructura ordenada por tablas:

```
{% block content %}

<form action="/resultados/mostrar/" method="POST" target="_blank">

    {% csrf_token %}

    <table>

        {{ form.as_table }}

    </table>

    <!--legend> Opciones </legend-->

    <div id="menu-resultados"> </div>
    <p>
        <input id="procesar" type="submit" value="Aceptar"/>
        <input id="borrar" type="reset" value="Borrar"/>
    </form>

    {% endblock %}
```

Error 404

Esta plantilla permite realizar la presentación del error 404 (no encontrado). Ubicación: {{TEMPLATE_DIRS}}/404.html:

```
{% extends "admin/base_site.html" %}

{% load i18n %}
```

Mensaje de página no encontrada:

```
{% block title %} {% trans "Page not found" %} {% endblock %}

{% block content %}

    <h1> {% trans "Page not found" %}. </h1>
    <p> Lo sentimos, pero la página solicitada no puede ser econtrada. </p>

{% endblock %}
```

Error 500

Esta plantilla permite realizar la presentación del error 500 (Error interno del servidor). Ubicación: {{TEMPLATE_DIRS}}/500.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML5.01//EN" "http://w3.org/TR/html4/strict.dtd">
<html lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title> Página no disponible </title>
  </head>
  <body>
```

Mensajes de error personalizados:

```
<h1> Página no disponible </h1>
<p> Lo sentimos, pero la página solicitada no está disponible debido a un error de...
<p> Nuestros ingenieros serán notificados, y lo solucionarán luego.!</p>
</body>
</html>
```

Las vistas y controladores

Introducción

Un función de vista o una vista, como es conocida generalmente, es una función en Python que hace una solicitud Web y devuelve una respuesta Web, esta respuesta puede ser el contenido de una página, un error 404, una imagen, un documento XML, entre muchas cosas más.

Comunmente se hace uso del archivo views.py para la redacción del código necesario. Pero para el desarrollo del presente sistema se ha modificado el nombre del archivo a controllers.py ubicado en sedd\ proyecto\app.

Note

Para mas información revisar CURSO DJANGO: LAS VISTAS.

Importaciones de los modulos de Django

Primero antes de todo se agrega el sistema de codificación para que acepte caracteres especiales:

```
#encoding:utf-8
```

Luego agregamos todos los paquetes necesarios para el correcto funcionamiento de las vistas:

```
from django.http import HttpResponseRedirect
from django.core.context_processors import csrf
from django.shortcuts import render_to_response
from django.shortcuts import redirect
from django.template import RequestContext
from django.template.loader import render_to_string
```

Para la autenticación:

```
from django.contrib import auth
```

Anotaciones para confirmación del login:

```
from django.contrib.auth.decorators import login_required
```

Formularios y restricciones:

```
from django import forms
from django.forms.forms import NON_FIELD_ERRORS
```

Consultas avanzadas , mensajes de error y algunas otras adiciones:

```
from django.contrib import messages
from django.db.models import Q
from django.utils import simplejson
```

Importaciones de las demás clases del Sistema

Haciendo uso de los *Los Modelos* para usar sus metodo y funciones:

```
from proyecto.app.models import Configuracion
from proyecto.app.models import Cuestionario
from proyecto.app.models import Seccion
from proyecto.app.models import Pregunta
from proyecto.app.models import Evaluacion
from proyecto.app.models import Contestacion
from proyecto.app.models import AsignaturaDocente
from proyecto.app.models import EstudianteAsignaturaDocente
from proyecto.app.models import EstudiantePeriodoAcademico
from proyecto.app.models import DocentePeriodoAcademico
from proyecto.app.models import DireccionCarrera
from proyecto.app.models import Asignatura
from proyecto.app.models import PeriodoAcademico
from proyecto.app.models import PeriodoEvaluacion
from proyecto.app.models import Tabulacion
from proyecto.app.models import TabulacionSatisfaccion2012
from proyecto.app.models import TabulacionAdicionales2012
from proyecto.app.models import TabulacionEvaluacion2013
from proyecto.app.models import OfertaAcademicaSGA
from proyecto.app.models import AreaSGA
```

Importación de todos los archivos ubicados en sedd\proyecto\app\forms.py:

```
from proyecto.app.forms import *
```

Módulos para autenticación haciendo uso de la red institucional para acceso al SGA:

```
from proyecto.tools.sgaws.cliente import SGA
from proyecto.settings import SGAWS_USER, SGAWS_PASS
```

Importaciones para trabajar con fechas y logs:

```
from datetime import datetime
import logging
log = logging.getLogger('logapp')
```

Portada

Mediante el uso de la Clase configuración obtenemos el periodo de evaluación actual y a su vez lo envía y renderiza la plantilla *Portada* . La información es enviada haciendo uso de la variable datos:

```
def portada(request):
    pea = Configuracion.getPeriodoEvaluacionActual()
    datos = dict(periodoEvaluacionActual=pea)
    return render_to_response("app/portada.html",datos)
```

Ingreso al Sistema

Para el acceso se hace uso de un registro común mediante un usuario y contraseña. Dichos datos son obtenidos de un formulario y comparados con la información de la base de datos para realizar su verificación:

```
def login(request):
    form = forms.Form()
```

Se implementó mensajes de error personalizados en caso de que se presente algún error en los datos:

```
form.fields['username'] = forms.CharField(label="Cedula", max_length=30,
                                            widget=forms.TextInput(attrs={'title': 'Cedula',}),
                                            error_messages={'required': 'Ingrese nombre de usuario'})

form.fields['password'] = forms.CharField(label="Clave SGA",
                                            widget=forms.PasswordInput(attrs={'title': 'Clave de acceso',}),
                                            error_messages={'required': 'Ingresar el password'})

data = dict(form=form)
if request.POST:
    username = request.POST.get('username')
    password = request.POST.get('password')
    user = auth.authenticate(username=username, password=password)
```

Controles necesarios en caso de que el usuario este o no activo:

```
if user is not None and user.is_active:
    auth.login(request, user)
    if not Configuracion.getPeriodoAcademicoActual() or not Configuracion.getPeriodoEvaluacionActual():
        return redirect('/logout')
    else:
        return redirect('/index')
else:
    form.full_clean()
    form._errors[NON_FIELD_ERRORS] = form.error_class(['Error de usuario o contraseña'])
data.update(csrf(request))
```

Renderiza la plantilla que se presentará [Login](#):

```
return render_to_response("app/login.html", data)
```

El siguiente método se ejecutará para el cierre de sesión del usuario. Su redirección es hacia la raíz ([Portada](#)) del proyecto:

```
def logout(request):
    auth.logout(request)
    return redirect('/')
```

Página Inicial

Una vez realizado el registro, el siguiente paso es el acceso a la página inicial. La anotación `@login_required()` permite dar acceso al método `index()` solo en caso de que exista un acceso previo al Sistema caso contrario redirecciona a la página de [Ingreso al Sistema](#):

```
@login_required(login_url='/login/')
def index(request):
    periodoAcademico = Configuracion.getPeriodoAcademicoActual()
    usuario = request.user
    noEstudiante = False
    noDocente = False
    periodoEvaluacion = Configuracion.getPeriodoEvaluacionActual()
    datos = dict()
```

Se obtiene solo las siglas de las áreas del periodo de evaluación actual:

```
areas_periodo = periodoEvaluacion.areasSGA.values_list('siglas', flat=True)
```

En caso de que se trate de un estudiante:

Las vistas y controladores

```
try:  
    estudiante = EstudiantePeriodoAcademico.objects.get(periodoAcademico=periodoAcademico, u
```

Aquí se toma en cuenta solo las carreras que están dentro de las áreas asignadas al presente Periodo de Evaluación:

```
carreras_estudiante = estudiante.asignaturasDocentesEstudiante.filter(  
    asignaturaDocente_asignatura_area_in=areas_periodo).values(  
        'asignaturaDocente_asignatura_carrera', 'asignaturaDocente_asignatura_area').distinct
```

Se agrega a sesión un diccionario con las carreras del estudiante registrado:

```
carreras_estudiante = [ dict(num_carrera=i, nombre=c['asignaturaDocente_asignatura_carrera'],  
                            area=c['asignaturaDocente_asignatura_area'])  
                        for i,c in enumerate(carreras_estudiante) ]  
request.session['estudiante'] = estudiante  
request.session['carreras_estudiante'] = carreras_estudiante  
except EstudiantePeriodoAcademico.DoesNotExist:  
    noEstudiante = True
```

En caso de tratarse de un docente:

```
try:  
    docente = DocentePeriodoAcademico.objects.get(periodoAcademico=periodoAcademico, usuario=  
    request.session['docente'] = docente  
    if 'ACE' in docente.get_areas():  
        cuestionarios_docente = periodoEvaluacion.cuestionarios.filter(informante_tipo='Docente')  
    else:  
        cuestionarios_docente = periodoEvaluacion.cuestionarios.filter(informante_tipo='Docente')  
    request.session['cuestionarios_docente'] = cuestionarios_docente  
    docente_autoevaluaciones = list()  
    cuestionarios_evaluados = [e.cuestionario for e in docente.evaluaciones.all()]  
    for c in cuestionarios_docente:  
        if c in cuestionarios_evaluados:  
            docente_autoevaluaciones.append(dict(cuestionario=c, evaluado=True))  
        else:  
            docente_autoevaluaciones.append(dict(cuestionario=c, evaluado=False))  
    request.session['docente_autoevaluaciones'] = docente_autoevaluaciones
```

En caso de tratarse de un docente DIRECTOR/COORDINADOR DE CARRERA:

```
if docente.direcciones.count() > 0:  
    request.session['director_carrera'] = docente  
    if 'ACE' in docente.get_areas():  
        cuestionarios_directivos = periodoEvaluacion.cuestionarios.filter(informante_tipo='Director')  
    else:  
        cuestionarios_directivos = periodoEvaluacion.cuestionarios.filter(informante_tipo='Coordinador')  
    request.session['cuestionarios_directivos'] = cuestionarios_directivos
```

Se agregan las carreras en las que el docente es coordinador o director, incluyendo también el nombre del área a la que pertenece:

```
carreras_director = [ dict(num_carrera=i,  
                           nombre=dc.carrera.split(' | ')[0],  
                           area=dc.carrera.split(' | ')[1])  
                     for i,dc in enumerate(docente.direcciones.all()) ]  
request.session['carreras_director'] = carreras_director
```

Si el docente es también PAR ACADEMICO:

```
elif docente.parAcademico:  
    if 'ACE' in docente.get_areas():  
        cuestionarios_pares_academicos = periodoEvaluacion.cuestionarios.filter(  
            informante_tipo='ParAcademicoIdiomas')  
    else:
```

```

cuestionarios_pares_academicos = periodoEvaluacion.cuestionarios.filter(
    informante_tipo='ParAcademico')
request.session['cuestionarios_pares_academicos'] = cuestionarios_pares_academicos
areas_periodo = periodoEvaluacion.areasSGA.values_list('siglas', flat=True)
carreras_pares_academicos = docente.asignaturasDocente.filter(asignatura_area_in=areas
    'asignatura_carrera', 'asignatura_area').distinct()

```

De igual manera se almacena un diccionario de carreras del para academicos en la session:

```

carreras_pares_academicos = [ dict(num_carrera=i, nombre=c['asignatura_carrera'],
    area=c['asignatura_area'])
        for i,c in enumerate(carreras_pares_academicos) ]
carreras_aux = [ c['nombre'] for c in carreras_pares_academicos ]
if docente.carrera and docente.carrera not in carreras_aux:

```

En docente solo existe el nombre de la carrera:

```

carreras_pares_academicos.append(dict(num_carrera=len(carreras_pares_academicos),
    nombre=docente.carrera,
    area=''))
aux_carreras = [c['nombre'] for c in carreras_pares_academicos]
aux_areas = [c['area'] for c in carreras_pares_academicos]

```

A continuación también se toma en cuenta los Pares Academicos de Frances y Russo:

```

if 'ACE' in aux_areas and 'Curso de Ingles' in aux_carreras:
    carreras_pares_academicos.append(dict(num_carrera=len(carreras_pares_academicos),
        nombre=u'Curso de Francés',
        area='ACE'))
    carreras_pares_academicos.append(dict(num_carrera=len(carreras_pares_academicos),
        nombre=u'Curso de Russo',
        area='ACE'))
request.session['carreras_pares_academicos'] = carreras_pares_academicos

```

except DocentePeriodoAcademico.DoesNotExist:
 noDocente = True

Si por algún motivo intenta acceder un usuario que no es Estudiante ni Docente en el Periodo Academicos Actual:

```

if noEstudiante and noDocente:
    return redirect('/login/')
return render_to_response('app/index.html', datos, context_instance=RequestContext(request))

```

Evaluaciones de los Estudiantes

Contiene la lógica necesaria para llevar a cabo la evaluación de los Estudiantes:

```

@login_required(login_url='/login/')
def estudiante_asignaturas_docentes(request, num_carrera):

```

Se recupera el estudiante y sus datos almacenados en sesión:

```

estudiante = request.session['estudiante']
carrera = [c['nombre'] for c in request.session['carreras_estudiante']
    if c['num_carrera'] == int(num_carrera)][0]
area = [c['area'] for c in request.session['carreras_estudiante']
    if c['num_carrera'] == int(num_carrera)][0]
request.session['carrera'] = carrera
request.session['area'] = area
request.session['num_carrera'] = num_carrera

```

Obtenemos los id de las AsignaturasDocente:

```

ids = estudiante.asignaturasDocentesEstudiante.filter(
    asignaturaDocente_asignatura_carrera=carrera, asignaturaDocente_asignatura_area=area)

```

Las vistas y controladores

```
'asignaturaDocente__id', flat=True).distinct()
asignaturasDocentes = AsignaturaDocente.objects.filter(id__in=ids).order_by('asignatura__nom...'
```

El código siguiente es necesario para obtener solo asignaturas distintas:

```
asignaturas = set([ad.asignatura for ad in asignaturasDocentes])
asignaturas_docentes = []
```

Los Objetos presentes a continuación serán renderizados en la vista:

```
periodoEvaluacion = Configuracion.getPeriodoEvaluacionActual()
for a in asignaturas:
    diccionario = {}
    diccionario['asignatura'] = a
    diccionario['docentes'] = []
```

Una asignatura puede tener mas de un docente:

```
docentes = [ad.docente for ad in asignaturasDocentes if ad.asignatura == a]
for d in docentes:
    if a.area == 'MED':
        cuestionarios = [c for c in periodoEvaluacion.cuestionarios.all()
                         if c.informante.tipo == 'EstudianteMED']
    elif a.area == 'ACE':
        cuestionarios = [c for c in periodoEvaluacion.cuestionarios.all()
                         if c.informante.tipo == 'EstudianteIdiomas']
```

Aplicable a Modalidad Presencial:

```
elif a.semestre == u"1":
    cuestionarios = [c for c in periodoEvaluacion.cuestionarios.all()
                     if c.informante.tipo == 'EstudianteNovel']
```

Si tienen los mismos cuestionarios que el resto de semestres:

```
else:
    cuestionarios = [c for c in periodoEvaluacion.cuestionarios.all()
                     if c.informante.tipo == 'Estudiante']

estudianteAsignaturaDocente = estudiante.asignaturasDocentesEstudiante.get(
    asignaturaDocente__asignatura=a, asignaturaDocente__docente=d
)
```

Si ya ha contestado todos los cuestionario disponibles para el docente 'd':

```
num_evaluaciones = estudianteAsignaturaDocente.evaluaciones.count()
if num_evaluaciones > 0 and num_evaluaciones == len(cuestionarios):
    diccionario['docentes'].append(dict(docente=d, evaluado=True))
else:
```

Si no ha contestado aun todos los cuestionario disponibles para el docente 'd':

```
    diccionario['docentes'].append(dict(docente=d, evaluado=False))
asignaturas_docentes.append(diccionario)
```

Para regresar posteriormente a esta vista:

```
request.session['num_carrera'] = str(num_carrera)
title = u"{}>>{}".format(area,carrera)
datos = dict(asignaturas_docentes=asignaturas_docentes, title=title)
```

Renderización de la plantilla *Asignaturas y Docentes*:

```
return render_to_response("app/asignaturas_docentes.html", datos, context_instance=RequestCo...
```

Evaluacion de Pares Académicos a Docentes

Las vistas y controladores

El siguiente método contiene la lógica necesaria para realizar la Evaluación del Par Académico al Docente:

```
def pares_academicos_docentes(request, num_carrera):
```

El Par Académico de la Carrera elige el docente a evaluar:

```
par_academico = request.session['docente']
carrera = [c['nombre'] for c in request.session['carreras_pares_academicos']]
    if c['num_carrera'] == int(num_carrera)][0]
area = [c['area'] for c in request.session['carreras_pares_academicos']]
    if c['num_carrera'] == int(num_carrera)][0]
request.session['carrera'] = carrera
request.session['area'] = area
request.session['num_carrera'] = num_carrera
```

Obtenemos los id de los Docentes que dictan Asignaturas en la carrera seleccionada:

```
ids_docentes = AsignaturaDocente.objects.filter(
    docente_periodoAcademico=Configuracion.getPeriodoAcademicoActual(),
    asignatura_carrera=carrera,
    asignatura_area=area).values_list(
        'docente_id', flat=True).distinct()
```

Se agregan también los docentes que no tengan Asignaturas pero que pertenezcan a la Carrera:

```
docentes = DocentePeriodoAcademico.objects.filter(
    periodoAcademico=Configuracion.getPeriodoAcademicoActual()) and
    (Q(id__in=ids_docentes) or Q(carrera=carrera))).order_by(
        'usuario_last_name', 'usuario_first_name')
docentes_evaluaciones = list()
cuestionarios_pares_a = request.session['cuestionarios_pares_academicos']
```

A continuación una comprobación para verificar si los ciertos cuestionarios ya han sido evaluados:

```
for d in docentes:
    cuestionarios_evaluados = sum([e.cuestionario in cuestionarios_pares_a for e in d.evalua
```

Comparación entre cuestionarios evaluados y cuestionarios establecidos:

```
if cuestionarios_evaluados >= len(cuestionarios_pares_a):
    docentes_evaluaciones.append(dict(docente=d, evaluado=True))
else:
    docentes_evaluaciones.append(dict(docente=d, evaluado=False))
docentes_evaluaciones.sort(
    lambda x,y: cmp(x['docente'].usuario.first_name, y['docente'].usuario.first_name) or
    cmp(x['docente'].usuario.last_name, y['docente'].usuario.last_name))
)
title = u"{}>>{}".format(area, carrera)
datos = dict(docentes_evaluaciones=docentes_evaluaciones, title=title)
```

Renderización al template :ref:`template-user-pares_academicos_docentes.html` con envío de datos:

```
return render_to_response('app/pares_academicos_docentes.html', datos, context_instance=Requ
```

Evaluaciones de Directores a Docentes

A continuación se presenta la lógica necesaria para la evaluación de Directores a Docentes Académicos. El Director de Carrera elige los docentes a evaluar:

```
def director_docentes(request, num_carrera):
    director_carrera = request.session['director_carrera']
    carrera = [c['nombre'] for c in request.session['carreras_director']]
        if c['num_carrera'] == int(num_carrera)][0]
    area = [c['area'] for c in request.session['carreras_director']]
        if c['num_carrera'] == int(num_carrera)][0]
    request.session['carrera'] = carrera
```

Las vistas y controladores

```
request.session['area'] = area
request.session['num_carrera'] = num_carrera
```

Obtenemos los id de los Docentes que dictan Asignaturas en la carrera seleccionada:

```
ids_docentes = AsignaturaDocente.objects.filter(
    docente_periodoAcademico=Configuracion.getPeriodoAcademicoActual(),
    asignatura_carrera=carrera,
    asignatura_area=area).values_list(
        'docente_id', flat=True).distinct()
```

Se agregan también los docentes que no tengan Asignaturas pero que pertenezcan a la Carrera:

```
docentes = DocentePeriodoAcademico.objects.filter(
    periodoAcademico=Configuracion.getPeriodoAcademicoActual()) and
    (Q(id__in=ids_docentes) or Q(carrera=carrera)).order_by(
        'usuario_last_name', 'usuario_first_name')
direccion = DireccionCarrera.objects.get(carrera=u'{0}|{1}'.format(carrera, area))
docentes_evaluaciones = list()
cuestionarios = request.session['cuestionarios_directivos']
for d in docentes:
```

Cuestionarios disponibles ya han sido evaluados? Forma pythonica de comparar, contar y sumar:

```
cuestionarios_evaluados = sum([e.cuestionario in cuestionarios for e in d.evaluciones.all()])
```

Compara cuestionarios evaluados con cuestionarios establecidos:

```
if cuestionarios_evaluados >= len(cuestionarios):
    docentes_evaluaciones.append(dict(docente=d, evaluado=True))
else:
    docentes_evaluaciones.append(dict(docente=d, evaluado=False))
docentes_evaluaciones.sort(
    lambda x,y: cmp(x['docente'].usuario.first_name, y['docente'].usuario.first_name) or
    cmp(x['docente'].usuario.last_name, y['docente'].usuario.last_name))
)
title = u'{0}>>{1}'.format(area, carrera)
datos = dict(docentes_evaluaciones=docentes_evaluaciones, direccion=direccion, title=title)
```

Renderización al template :ref:`template-user-director_docentes.html` con envio de datos:

```
return render_to_response('app/director_docentes.html', datos, context_instance=RequestContext)
```

Listado de Encuestas

El siguiente código tiene la finalidad el listar las encuestas para los informantes en el presente periodo de evaluación:

```
def encuestas(request, id_docente, id_asignatura=0, id_tinformante=0, id_cuestionario=0):
```

El parámetro `id_docente` representa el Docente a ser evaluado El parámetro `id_asignatura` contiene la asignatura que dicta el docente a evaluar El parámetro `id_informante` contiene el Id Tipo Informante, por ejemplo: 1 Estudiante, 5 Docente, 6 Directivo El parámetro `id_cuestionario` Solo se usa en el caso de cuestionarios para docentes.

```
datos = dict()
title=''
periodoEvaluacionActual = Configuracion.getPeriodoEvaluacionActual()
cuestionarios = []
periodo_finalizado = False
periodo_no_iniciado = False
```

En caso de Periodo no iniciado aun:

```
if periodoEvaluacionActual.noIniciado():
    periodo_no_iniciado = True
```

Las vistas y controladores

En caso de Periodo Vigente:

```
elif periodoEvaluacionActual.vigente():
```

Si se ha ingresado como ESTUDIANTE:

```
if id_informante == '1':  
    estudiante = request.session['estudiante']
```

Se maneja las siglas del Area en la session:

```
area = request.session['area']
carrera = request.session['carrera']
asignaturaDocente = AsignaturaDocente.objects.get(docente__id=id_docente, asignatura__id=id_asignatura)
estudianteAsignaturaDocente = EstudianteAsignaturaDocente.objects.get(
    estudiante=estudiante, asignaturaDocente=asignaturaDocente)
request.session['estudianteAsignaturaDocente'] = estudianteAsignaturaDocente
title = u"{}>>{}>>{}>>{}".format(area, carrera, asignaturaDocente.asignatura.nombre,
                                             asignaturaDocente.docente)
if asignaturaDocente.asignatura.area == 'MED':
    cuestionarios = [c for c in periodoEvaluacionActual.cuestionarios.all()
                    if c.informante.tipo == 'EstudianteMED']
```

Estudiante (Asignatura) del Instituto de Idiomas:

```
elif asignaturaDocente.asignatura.area == "ACE":  
    cuestionarios = [c for c in periodoEvaluacionActual.cuestionarios.all()  
                    if c.informante.tipo == 'EstudianteIdiomas']
```

Estudiante del Primer Semestre:

```
elif asignaturaDocente.asignatura.semestre == u"1":  
    cuestionarios = [c for c in periodoEvaluacionActual.cuestionarios.all()  
                    if c.informante.tipo == 'EstudianteNovel']
```

Si los cuestionarios son los mismos para todos los semestres.

Estudiante del segundo semestre en adelante:

Si ya ha contestado todos los cuestionario disponibles. Para esto se realiza una comprobación para verificar si existen cuestionarios disponibles:

```
evaluados = sum([e.cuestionario in cuestionarios for e in estudianteAsignaturaDocente.evalua
```

Comparación entre cuestionarios evaluados y cuestionarios establecidos:

```
if evaluados >= len(cuestionarios):
    return redirect('/estudiante/asignaturas/docentes/' + request.session['num_carrera'])
```

La siguiente sección de código es ejecutado en el caso de que el usuario ha ingresado como docente y es DIRECTOR DE CARRERA:

```
elif id_tinformante == '6': #Tambien DirectorIdiomas

    # Se maneja las siglas del Area en la sesion

    director_carrera = request.session['director_carrera']
    area = request.session.get('area', None)
    carrera = request.session.get('carrera', None)
    docente_evaluar = DocentePeriodoAcademico.objects.get(id=id_docente)
    request.session['docente_evaluar'] = docente_evaluar
    title = u"{}>>{}>>Coordinador: {}".format(area, carrera, director_carrera)
```

```
cuestionarios = request.session['cuestionarios_directivos']
datos.update(dict(docente_evaluar=docente_evaluar))

# Cuestionarios disponibles ya han sido evaluados? Forma pythonica de comparar, contar y
evaluados = sum([e.cuestionario in cuestionarios for e in docente_evaluar.evaluaciones.all()])
```

Comparación entre cuestionarios evaluados y cuestionarios establecidos:

```
if evaluados >= len(cuestionarios):
    return redirect('/director/docentes/' + request.session['num_carrera'])
```

La siguiente sección de código es ejecutado en el caso de que el usuario ha ingresado como docente y es PAR ACADEMICO:

```
elif id_tinformante == '10': #Tambien ParAcademicoIdiomas

    # Se maneja las siglas del Area en la sesion

    par_academico = request.session['docente']
    area = request.session.get('area', None)
    carrera = request.session.get('carrera', None)
    docente_evaluar = DocentePeriodoAcademico.objects.get(id=id_docente)
    request.session['docente_evaluar'] = docente_evaluar
    title = u"{}>>{}>>Par Académico: {}".format(area, carrera, par_academico)
    cuestionarios = request.session['cuestionarios_pares_academicos']
    datos.update(dict(docente_evaluar=docente_evaluar))

    # Cuestionarios disponibles ya han sido evaluados? Forma pythonica de comparar, contar y
    evaluados = sum([e.cuestionario in cuestionarios for e in docente_evaluar.evaluaciones.all()])
```

Comparación cuestionarios evaluados y cuestionarios establecidos:

```
if evaluados >= len(cuestionarios):
    return redirect('/pares_academicos/docentes/' + request.session['num_carrera'])
```

La siguiente sección de código es ejecutado en el caso de que el usuario es DOCENTE:

```
elif id_tinformante == '5' and id_cuestionario: # Tambien DocenteIdiomas
```

Se filtra la petición de encuestas de autoevaluacion de docentes desde el index por este controlador para reutilizar la validación de vigencia de PeriodoEvaluacion:

```
return redirect('/encuesta/responder/' + id_cuestionario)
```

Si ha expirado el periodo de Evaluacion:

```
elif periodoEvaluacionActual.finalizado():
    periodo_finalizado = True
    datos.update(dict(cuestionarios=cuestionarios, title=title,
                      periodo_no_iniciado=periodo_no_iniciado, periodo_finalizado=periodo_finalizado))
```

Renderización al template :ref:`template-user-encuestas.html` con envio de datos:

```
return render_to_response("app/encuestas.html", datos, context_instance=RequestContext(request))
```

Generación de la Encuesta

La finalidad de este código es generar y presentar la encuesta que será completada por los estudiantes, Directores de carrera, pares académicos y Docentes en general para autoevaluacion:

```
def encuesta_responder(request, id_cuestionario):
    datos = dict()
    cuestionario = Cuestionario.objects.get(id=id_cuestionario)
    evaluacion = Evaluacion()
```

```
if cuestionario.informante.tipo.startswith('Estudiante'):
    area = request.session['area']
    carrera = request.session['carrera']
    estudianteAsignaturaDocente = request.session['estudianteAsignaturaDocente']
    evaluacion.estudianteAsignaturaDocente = estudianteAsignaturaDocente
    title = u'{0}>>{1}>>{2}>>{3}'.format(area, carrera,
                                                estudianteAsignaturaDocente.asignaturaDocente,
                                                estudianteAsignaturaDocente.asignaturaDocente,
                                                )
    datos.update(dict(asignaturaDocente=estudianteAsignaturaDocente.asignaturaDocente))
```

Encuesta para DIRECTORES DE CARRERA:

```
elif cuestionario.informante.tipo.startswith('Directivo'):
    area = request.session['area']
    carrera = request.session['carrera']
    director_carrera = request.session['director_carrera']
    docente_evaluar = request.session['docente_evaluar']
    title = u'{0}>>{1}>>{2}'.format(area, carrera, docente_evaluar)
    evaluacion.directorCarrera = director_carrera
    evaluacion.carreraDirector = u'{0}|{1}'.format(carrera, area)
    evaluacion.docentePeriodoAcademico = docente_evaluar
```

Encuesta para PARES ACADEMICOS:

```
elif cuestionario.informante.tipo.startswith('ParAcademico'):
    area = request.session['area']
    carrera = request.session['carrera']
    par_academico = request.session['docente']
    docente_evaluar = request.session['docente_evaluar']
    title = u'{0}>>{1}>>{2}'.format(area, carrera, docente_evaluar)
    evaluacion.parAcademico = par_academico
    evaluacion.docentePeriodoAcademico = docente_evaluar
```

Encuesta para Autoevaluacion de DOCENTES:

```
elif cuestionario.informante.tipo.startswith('Docente'):
```

En caso de que el docente sea a la vez director de carrera:

```
docente = None
if request.session.get('director_carrera', None):
    docente = request.session['docente']
elif request.session.get('docente', None):
    docente = request.session['docente']
evaluacion.docentePeriodoAcademico = docente
```

Areas en las que dicta clases el docente:

```
areas_docente = AsignaturaDocente.objects.filter(docente=docente).values_list('asignatura_a')
areas_docente = ',' .join(areas_docente)
```

Carreras en las que dicta clases el docente:

```
carreras_docente = AsignaturaDocente.objects.filter(docente=docente).values_list(
    'asignatura_carrera', flat=True).distinct()
carreras_docente = ',' .join(carreras_docente)
request.session['areas_docente'] = areas_docente
request.session['carreras_docente'] = carreras_docente
title = u'{0}>>{1}>>{2}'.format(areas_docente, carreras_docente, docente)

evaluacion.fechaInicio = datetime.now().date()
evaluacion.horaInicio = datetime.now().time()
evaluacion.cuestionario = cuestionario
```

```
request.session['evaluacion'] = evaluacion
fecha = datetime.now()
datos.update(dict(cuestionario=cuestionario, title=title, fecha=fecha))
```

Parametro adicional por cuestion del formulario:

```
datos.update(csrf(request))
```

Renderización al template :ref:`template-user-encuesta_responder.html` con envio de datos:

```
return render_to_response("app/encuesta_responder.html", datos, context_instance=RequestContext)
```

Grabación de la Encuesta

El presente código tiene la finalidad de grabar la encuesta una vez que ha sido desarrollada por los estudiantes o docentes adecuados:

```
def encuesta_grabar(request):
    datos = dict(num_carrera=request.session.get('num_carrera', None))
    evaluacion = request.session.get('evaluacion', None)
```

En caso de intentar dar nuevamente la encuesta terminada se redirecciona a *Encuesta Finalizada*

```
if not evaluacion:
    return render_to_response('app/encuesta_finalizada.html', datos, context_instance=RequestContext)
```

Finalización de la encuesta dirigida a ESTUDIANTES:

```
if evaluacion.cuestionario.informante.tipo.startswith('Estudiante'):
```

:: Si se regresa a grabar otra vez la misma encuesta:

```
if Evaluacion.objects.filter(estudianteAsignaturaDocente = estudianteAsignaturaDocente
                             ).filter(cuestionario = evaluacion.cuestionario).count() > 0:
    request.session['evaluacion'] = None
    request.session['estudianteAsignaturaDocente'] = None
    return render_to_response('app/encuesta_finalizada.html',
                             datos, context_instance=RequestContext(request))
```

Finalización de la Encuesta dirigida a DIRECTORES DE CARRERA:

```
elif evaluacion.cuestionario.informante.tipo.startswith('Directivo'):
    docente_evaluar = request.session['docente_evaluar']
    director_carrera = request.session['director_carrera']
```

Si ya ha contestado este cuestionario y se intenta grabar otra vez:

```
if docente_evaluar.evaluaciones.filter(cuestionario=evaluacion.cuestionario,
                                         directorCarrera=director_carrera).count() > 0:
    request.session['evaluacion'] = None
    return render_to_response('app/encuesta_finalizada.html', datos,
                             context_instance=RequestContext(request))
```

Finalización de la encuesta dirigida a PARES ACADEMICOS:

```
elif evaluacion.cuestionario.informante.tipo.startswith('ParAcademico'):
    docente_evaluar = request.session['docente_evaluar']
    par_academico = request.session['docente']
```

Si ya ha contestado este cuestionario y se intenta grabar otra vez:

```
if docente_evaluar.evaluaciones.filter(cuestionario=evaluacion.cuestionario,
                                         parAcademico=par_academico).count() > 0:
    request.session['evaluacion'] = None
    return render_to_response('app/encuesta_finalizada.html', datos,
                             context_instance=RequestContext(request))
```

Las vistas y controladores

Finalización de la Encuesta dirigida a DOCENTES:

```
elif evaluacion.cuestionario.informante.tipo.startswith('Docente'):
    docente = request.session.get('docente', None)
```

Si ya ha contestado este cuestionario y se intenta grabar otra vez:

```
if docente.evaluaciones.filter(cuestionario=evaluacion.cuestionario).count() > 0:
    request.session['evaluacion'] = None
    return render_to_response('app/encuesta_finalizada.html', datos,
                             context_instance=RequestContext(request))
```

Grabacion luego de las validaciones:

```
evaluacion.fechaFin = datetime.now().date()
evaluacion.horaFin = datetime.now().time()
evaluacion.save()
for k,v in request.POST.items():
    if k.startswith('csrf'):
        continue
    if k.startswith('pregunta'):
        id_pregunta = int(k.split('-')[1])
        observaciones = None
```

Solo se graban las observaciones de las preguntas respondidas:

```
if Pregunta.objects.get(id=id_pregunta).observaciones:
    observaciones = request.POST['observaciones-pregunta-' + str(id_pregunta)]
    contestacion = Contestacion(pregunta=id_pregunta, respuesta=v, observaciones=observaciones)
    contestacion.evaluacion = evaluacion
    contestacion.save()
log.info("Nueva Evaluacion realizada: {}".format(evaluacion))
```

Una vez terminada la encuesta se redirecciona a *Encuesta Finalizada*

```
return render_to_response('app/encuesta_finalizada.html', datos, context_instance=RequestContext(request))
```

Ofertas SGA

El método descrito a continuación se invoca a través de AJAX con la finalidad de recargar todas las ofertas presentes en SGA:

```
def cargar_ofertas_sga(request, periodoAcademicoId):
```

En caso de no existir un periodo académico:

```
if not periodoAcademicoId:
    return HttpResponse("Falta Periodo Academico")
```

Excepción:

```
try:
    proxy = SGA(SGAWS_USER, SGAWS_PASS)
    periodoAcademico = PeriodoAcademico.objects.get(id=periodoAcademicoId)
    ofertas_dict = proxy.ofertas_academicas(periodoAcademico.inicio, periodoAcademico.fin)
    ofertas = [OfertaAcademicaSGA(idSGA=oa['id'], descripcion=oa['descripcion']) for oa in ofertas_dict]
    for oa in ofertas:
        try:
            OfertaAcademicaSGA.objects.get(idSGA=oa.idSGA)
        except OfertaAcademicaSGA.DoesNotExist:
            oa.save()
    return HttpResponse("OK")
```

En caso de algún error se presenta un mensaje:

```
except Exception, e:  
    log.error("Error recargando ofertas SGA: " + str(e))  
    return HttpResponse("error: "+str(e))
```

Resumen de Evaluaciones en modo Administrador

Los métodos descritos a continuación tienen la lógica necesaria para permitirle, al Administrador del Sistema, la obtención de los resúmenes detallados de las distintas evaluaciones.

Resumen de Evaluaciones

Método invocado a través de AJAX cuando hay un cambio en el menú del Resumen

```
def resumen_evaluaciones(request):
```

Se detecta si es petición mediante AJAX:

```
if request.is_ajax():  
    respuesta = menu_academico_ajax(request)  
    return HttpResponse(respuesta, mimetype='application/json')
```

Creación del formulario con los campos necesarios:

```
else:
```

```
    form = forms.Form()  
    form.fields['periodo_academico'] = forms.ModelChoiceField(queryset=PeriodoAcademico.objects.all(), required=True)  
    form.fields['periodo_academico'].label = 'Periodos Academicos'  
    form.fields['periodo_evaluacion'] = forms.ModelChoiceField(PeriodoEvaluacion.objects.all(), required=True)  
    form.fields['periodo_evaluacion'].label = 'Periodos de Evaluacion'  
    form.fields['area'] = forms.ModelChoiceField(AreaSGA.objects.none())  
    form.fields['carrera'] = forms.ModelChoiceField(Asignatura.objects.none())  
    form.fields['semestre'] = forms.ModelChoiceField(Asignatura.objects.none())  
    form.fields['paralelo'] = forms.ModelChoiceField(Asignatura.objects.none())  
    periodoEvaluacion = Configuracion.getPeriodoEvaluacionActual()  
    datos = dict(form=form, evaluadores=periodoEvaluacion.contabilizar_evaluadores(), periodo=periodoEvaluacion)
```

Renderización del template *Resumen de Evaluaciones* ubicado en la carpeta admin/app que a su vez contiene los templates usados en la parte de administración:

```
return render_to_response("admin/app/resumen_evaluaciones.html", datos)
```

Calculos de Resumen

Se incluye dentro de una Excepción por si se presenta algún error durante el llamado:

```
def calcular_resumen(request):  
    try:  
        if request.is_ajax():  
            id_periodo_evaluacion = int(request.GET['id_periodo_evaluacion'])  
            area = request.GET['area']  
            carrera = request.GET['carrera']  
            semestre = request.GET['semestre']  
            paralelo = request.GET['paralelo']  
            periodoEvaluacion = PeriodoEvaluacion.objects.get(id=id_periodo_evaluacion)  
            resumen = periodoEvaluacion.contabilizar_evaluaciones_estudiantes(area, carrera, semestre, paralelo)
```

Contiene: estudiantes, completados, faltantes:

```
        return HttpResponse(simplejson.dumps(resumen), mimetype='application/json')  
    except Exception, ex:  
        logg.error("Error calculando resumen de evaluaciones {0}".format(ex))
```

Menú Académico

Las vistas y controladores

Método usado para la creación del menú que contiene las funcionalidades para los resúmenes de las evaluaciones. Se caracteriza por su funcionalidad reutilizada cuando se necesita información académica jerárquica estructurada en (PeriodoAcademico, PeriodoEvaluacion, AreaSGA, carrera, semestre, paralelo). Utilizado generalmente en menús de reportes:

```
def menu_academico_ajax(request):
    try:
        id_campo = request.GET['id']
        valor_campo = request.GET['valor']
        campo_siguiente = request.GET['siguiente']
        if valor_campo == '':
            return HttpResponseRedirect({'id': "", "valores": []}, mimetype="JSON")
        id = ""
        valores = [
```

Según el tipo de valor que contenga `id_campor`, se realizará una operación distinta:

```
if id_campo == 'id_periodo_academico':
    request.session['periodoAcademico'] = PeriodoAcademico.objects.get(id=int(valor_campo))
    id = 'id_periodo_evaluacion'
    objetos = PeriodoEvaluacion.objects.filter(periodoAcademico__id=int(valor_campo)).all()
    valores = [dict(id=o.id, valor=o.nombre) for o in objetos]
elif id_campo == 'id_periodo_evaluacion':
    request.session['periodoEvaluacion'] = PeriodoEvaluacion.objects.get(id=int(valor_campo))
    id = 'id_area'
    objetos = PeriodoEvaluacion.objects.get(id=int(valor_campo)).areasSGA.all()
    valores = [dict(id=o.siglas, valor=o.nombre) for o in objetos]
elif id_campo == 'id_area':
    # request.session['area'] = AreaSGA.objects.get(siglas=valor_campo)
    request.session['area'] = valor_campo
    id = 'id_carrera'
    objetos = EstudianteAsignaturaDocente.objects.filter(
        estudiante_periodoAcademico=request.session['periodoAcademico']).filter(
        asignaturaDocente_asignatura_area=valor_campo).values_list(
        'asignaturaDocente_asignatura_carrera', flat=True).distinct()
    for o in objetos:
        carrera = o.encode('utf-8') if isinstance(o, unicode) else o
        valores.append(dict(id=carrera, valor=carrera))
```

Se presentan dos bifurcaciones dependiendo del menú:

```
elif id_campo == 'id_carrera':
    request.session['carrera'] = valor_campo
    id = campo_siguiente
    if campo_siguiente == 'id_semestre':
        objetos = EstudianteAsignaturaDocente.objects.filter(
            estudiante_periodoAcademico=request.session['periodoAcademico']).filter(
            asignaturaDocente_asignatura_area=request.session['area']).filter(
            asignaturaDocente_asignatura_carrera=valor_campo).values_list(
            'asignaturaDocente_asignatura_semestre', flat=True).distinct()
        for o in objetos:
            semestre = o.encode('utf-8') if isinstance(o, unicode) else o
            valores.append(dict(id=semestre, valor=semestre))
    elif campo_siguiente == 'id_docente':
        objetos = AsignaturaDocente.objects.filter(
            docente_periodoAcademico=request.session['periodoAcademico']).filter(
            asignatura_area=request.session['area']).filter(
            asignatura_carrera=valor_campo)
        docentes = set([o.docente for o in objetos])
        valores = [dict(id=d.id, valor=d.__unicode__()) for d in docentes]
    elif id_campo == 'id_semestre':
        request.session['semestre'] = valor_campo
```

```

id = id_paralelo
objetos = EstudianteAsignaturaDocente.objects.filter(
    estudiante_periodoAcademico=request.session['periodoAcademico']).filter(
        asignaturaDocente_asignatura_area=request.session['area']).filter(
            asignaturaDocente_asignatura_carrera=request.session['carrera']).filter(
                asignaturaDocente_asignatura_semestre=valor_campo).values_list(
                    'asignaturaDocente_asignatura_paralelo', flat=True).distinct()
for o in objetos:
    paralelo = o.encode('utf-8') if isinstance(o, unicode) else o
    valores.append(dict(id=paralelo, valor=paralelo))

resultado = {id:id, 'valores':valores}
return simplejson.dumps(resultado)

```

Se genera un log de error en caso de presentarse una excepción:

```

except Exception, ex:
    logg.error("Error en menu_academico_ajax: " + str(ex))
    return ""

```

Cálculo y Presentación de resultados

Resultados por Carrera

El presente método se encarga de obtener los resultados de las evaluaciones tomando según la carrera. para su funcionamiento uno de sus parámetros en el número de la carrera(num_carrera) que lo identifica. Se trabaja con las carreras cuya dirección esta a cargo del docente almacenadas en la vista previa index:

```

def resultados_carrera(request, num_carrera):
    datos = dict()

    try:
        carreras_director = request.session['carreras_director']
        for cd in carreras_director:
            if cd['num_carrera'] == int(num_carrera):
                carrera = cd['nombre']
                area_siglas = cd['area']
                break
        else:
            logg.error('No hay carreras para este docente director')
            request.session['carrera'] = carrera
            request.session['area'] = area_siglas
    except:
        pass

```

Se hace uso del objeto AreaSGA:

```

area = AreaSGA.objects.get(siglas=area_siglas)
periodoAcademico = Configuracion.getPeriodoAcademicoActual()
# Periodos de Evaluacion del Periodo Academico Actual
periodosEvaluacion = area.periodosEvaluacion.filter(periodoAcademico=periodoAcademico)
form = forms.Form()

```

Se selecciona solo los períodos de evaluación en los que se encuentra el área del docente director:

```

form.fields['periodo_evaluacion'] = forms.ModelChoiceField(
    queryset=area.periodosEvaluacion.filter(periodoAcademico=periodoAcademico)
)
form.fields['periodo_evaluacion'].label = 'Periodo de Evaluación'
datos = dict(form=form, title='>> Coordinador Carrera ' + carrera )

```

Excepción en caso de error:

```

except Exception, ex:
    logg.error("Error : " + str(ex))

```

Se renderiza el template [Menú Resultados Carreras](#):

```
return render_to_response("app/menu_resultados_carrera.html", datos, context_instance=RequestContext)
```

Menú de resultados por Carrera

Su funcionamiento consiste en generar el menu de opciones para reportes de acuerdo al periodo de evaluación y su tipo de tabulacion específicamente. Llamado con AJAX:

```
def menu_resultados_carrera(request, id_periodo_evaluacion):
    try:
        periodoEvaluacion = PeriodoEvaluacion.objects.get(id=id_periodo_evaluacion)
        tabulacion = Tabulacion.objects.get(periodoEvaluacion=periodoEvaluacion)
```

Para los Docentes Coordinadores de Carrera:

```
if request.session.has_key('carreras_director'):
    carrera = request.session['carrera']
    area = request.session['area']
```

Para la Comision de Evaluacion (Administracion):

```
else:
    area = request.GET['area']
    carrera = request.GET['carrera']
```

Especifico para el Tipo de Tabulacion - Periodo de Evaluacion. Según el tipo de evaluación se formatean los formularios de administración: [Formularios ese2012](#), [Formularios eaad2012](#) y [Formularios edd2013](#):

```
if tabulacion.tipo == 'ESE2012':
    tabulacion = TabulacionSatisfaccion2012(periodoEvaluacion)
    form = ResultadosESE2012Form(tabulacion, area, carrera)
    formulario_formateado = render_to_string("admin/app/formulario_ese2012.html", dict(form=form))
elif tabulacion.tipo == 'EAAD2012':
    tabulacion = TabulacionAdicionales2012(periodoEvaluacion)
    form = ResultadosEAAD2012Form(tabulacion, area, carrera)
    formulario_formateado = render_to_string("admin/app/formulario_eaad2012.html", dict(form=form))
elif tabulacion.tipo == 'EDD2013':
    tabulacion = TabulacionEvaluacion2013(periodoEvaluacion)
    form = ResultadosEDD2013Form(tabulacion, area, carrera)
    formulario_formateado = render_to_string("admin/app/formulario_edd2013.html", dict(form=form))
return HttpResponse(formulario_formateado)
```

Excepciones:

```
except PeriodoEvaluacion.DoesNotExist:
    logg.error(u"No Existe el Periodo de Evaluación: {0}".format(id_periodo_evaluacion))
except Exception, ex:
    logg.error("Error en vista menu_resultados_carrera: " + str(ex))
```

Presentación de Resultados

Método orientado a realizar la presentación de los resultados obtenidos:

```
def mostrar_resultados(request):
```

Se hace uso de unas cuantas etiquetas HTML para mejor presentación:

```
if not (request.POST.has_key('periodo_evaluacion') and request.POST.has_key('opciones')):
    return HttpResponse("<h2> Tiene que elegir las Opciones de Resultados </h2>")
id_periodo = request.POST['periodo_evaluacion']
if id_periodo == '':
    return HttpResponse("<h2> Tiene que elegir el Periodo de Evaluación </h2>")
```

Se obtiene la opcion generica para cualquier tipo de evaluacion:

Las vistas y controladores

```
opcion = request.POST['opciones']
periodoEvaluacion=PeriodoEvaluacion.objects.get(id=int(id_periodo))
tabulacion = periodoEvaluacion.tabulacion
```

Encuesta de Satisfaccion Estudiantil 2012:

```
if tabulacion.tipo == 'ESE2012':
    tabulacion = TabulacionSatisfaccion2012(periodoEvaluacion)
    metodo = [c[2] for c in tabulacion.calculos if c[0] == opcion][0]
    titulo = request.session['area'] + '<br/> <b>' + request.session['carrera'] + '</b><br/>'
    titulo += [c[3] for c in tabulacion.calculos if c[0] == opcion][0]
    resultados = {}
```

Por docente:

```
if opcion == 'a':
    id_docente = request.POST['docentes']
    if id_docente != '':
        titulo += u': <b>{0}</b>'.format(DocentePeriodoAcademico.objects.get(id=int(id_docente)))
        resultados = metodo(request.session['area'], request.session['carrera'], int(id_docente))
```

Por campos:

```
elif opcion == 'c':
    id_seccion = request.POST['campos']
    if id_seccion != '':
        seccion = Seccion.objects.get(id=int(id_seccion))
        titulo += u': <b>{0}</b>'.format(seccion.titulo)
        resultados = metodo(request.session['area'], request.session['carrera'], int(id_seccion))
    if seccion.orden == 4:
        datos = dict(resultados=resultados, titulo=titulo)
        return render_to_response('app/imprimir_otros_ese2012.html', datos,
                                 context_instance=RequestContext(request));
```

Por indicadores:

```
elif opcion == 'd':
    id_pregunta = request.POST['indicadores']
    if id_pregunta != '':
        titulo += u': <b>{0}</b>'.format(Pregunta.objects.get(id=int(id_pregunta)).texto)
        resultados = metodo(request.session['area'], request.session['carrera'], int(id_pregunta))
```

Para el resto de casos:

```
else:
    resultados = metodo(request.session['area'], request.session['carrera'])
if resultados:
    resultados['titulo'] = titulo
plantilla = 'app/imprimir_resultados_ese2012.html'
```

Evaluacion de Actividades Adiconales a la Docencia 2011 - 2012:

```
if tabulacion.tipo == 'EAAD2012':
```

Nombre completo del Area para su presentacion en el reporte:

```
area = AreaSGA.objects.get(siglas=request.session['area']).nombre
carrera = request.session['carrera']
tabulacion = TabulacionAdicionales2012(periodoEvaluacion)
metodo = [c[2] for c in tabulacion.calculos if c[0] == opcion][0]
```

Por docente:

```
resultados = {}
if opcion == 'a':
    id_docente = request.POST['docentes']
```

Las vistas y controladores

```
if id_docente != '':
    docente = DocentePeriodoAcademico.objects.get(id=int(id_docente))
    # Referencia a lo que devuelve el metodo especifico invocado sobre la instancia de Tabulacion
    resultados = metodo(request.session['area'], request.session['carrera'], int(id_docente))
elif opcion == 'z':
    pass
```

Para el resto de casos:

```
else:
    resultados = metodo(request.session['area'], request.session['carrera'])
if resultados:
    resultados['docente'] = docente
    resultados['carrera'] = carrera
    resultados['area'] = area
plantilla = 'app/imprimir_resultados_eaad2012.html'
```

Evaluacion del Desempenio Docente 2012 - 2013:

```
if tabulacion.tipo == 'EDD2013':
    if opcion == 'c' and not request.user.is_staff:
        return HttpResponse("<h2> Ud no tiene permisos para revisar este reporte </h2>")
    codigos_filtro = {'a': '', 'b': 'CPF', 'c': 'CPG', 'd': 'PV', 'e': 'sugerencias'}
    objeto_area = AreaSGA.objects.get(siglas=request.session['area'])
```

Nombre completo del Area para su presentacion en el reporte:

```
area = objeto_area.nombre
area_siglas = request.session['area']
carrera = request.session['carrera']
tabulacion = TabulacionEvaluacion2013(periodoEvaluacion)
metodo = [c[2] for c in tabulacion.calculos if c[0] == opcion][0]
filtro = request.POST['filtros']
filtro = codigos_filtro[filtro]
```

Por docente:

```
resultados = {}
if opcion == 'a':
    id_docente = request.POST['docentes']
    if id_docente != '':
        docente = DocentePeriodoAcademico.objects.get(id=int(id_docente))
```

Referencia a lo que devuelve el metodo especifico invocado sobre la instancia de Tabulacion:

```
resultados = metodo(request.session['area'], request.session['carrera'], int(id_docente), filtro)
resultados['docente'] = docente
resultados['carrera'] = carrera
resultados['area'] = area
```

Por Carrera:

```
elif opcion == 'b':
    resultados = metodo(request.session['area'], request.session['carrera'], filtro)
    resultados['carrera'] = carrera
    resultados['area'] = area
```

Por Area:

```
elif opcion == 'c':
    resultados = metodo(request.session['area'], filtro)
    resultados['area'] = area
```

Para el resto de casos:

Mapeo de URLs

```
else:  
    resultados = metodo(request.session['area'], request.session['carrera'], filtro)
```

Posicion para ubicar el promedio por componente en la plantilla.

Si se trata del Instituto de Idiomas:

```
if resultados.get('promedios_componentes', None) and objeto_area.id == 6:  
  
    resultados['promedios_componentes']['CPF'].update({'fila' : 8})  
    resultados['promedios_componentes']['CPG'].update({'fila' : 23})  
    resultados['promedios_componentes']['PV'].update({'fila' : 29})
```

Para el resto de Areas:

```
elif resultados.get('promedios_componentes', None):  
  
    resultados['promedios_componentes']['CPF'].update({'fila' : 10})  
    resultados['promedios_componentes']['CPG'].update({'fila' : 27})  
    resultados['promedios_componentes']['PV'].update({'fila' : 33})
```

Se trata de reporte de sugerencias se escoge entre las plantillas *Sugerencias Evaluación Desempeño Docente 2013, Resultados Desempeño Docente 2013*:

```
if filtro == 'sugerencias':  
    plantilla = 'app/imprimir_sugerencias_edd2013.html'  
else:  
    plantilla = 'app/imprimir_resultados_edd2013.html'
```

Se formatea la vista:

```
return render_to_response(plantilla, resultados, context_instance=RequestContext(request));
```

Resultados para los Administradores

Manejo de resultados para los administradores:

```
def resultados(request):  
    datos = dict(form=ResultadosForm())
```

Se formatea la vista de administración *Menú resultados*:

```
return render_to_response('admin/app/menu_resultados.html', datos, context_instance=RequestContext(request));
```

Mapeo de URLs

Introducción

Django provee un sistema avanzado para la manipulación de URLs que convina la sencillez con la potencia y versatilidad que da como resultado un sistema de enrutamiento elegante.

Note

Para más información revisar [Mapear URLs y Vistas](#) o en la documentación oficial [URL dispatcher](#).

Importaciones Necesarias

El archivo que contiene el enruteamiento principal se encuentra ubicado en: `sedd/proyecto/urls.py`.

Se importa el módulo principal para la administración de las urls:

```
from django.conf.urls.defaults import patterns, include, url
```

Y para la administración:

```
from django.contrib import admin
```

Enrutamiento Principal

En esta sección se redacta el enrutamiento necesario para un correcto funcionamiento del Sistema:

```
admin.autodiscover()
```

```
urlpatterns = patterns('',
```

A continuación una breve descripción de cada ruta:

Método de origen: Portada.

URL: /.

Nombre: portada.

Código:

```
url(r'^$', 'proyecto.app.controllers.portada', name='portada'),
```

Método de origen: Ingreso al Sistema.

URL: /login.

Nombre: login.

Código:

```
url(r'^login/$', 'proyecto.app.controllers.login', name='login'),
```

Método de origen: Página Inicial.

URL: index/.

Nombre: index.

Código:

```
url(r'^index/$', 'proyecto.app.controllers.index', name='index'),
```

Método de origen: Ingreso al Sistema.

URL: logout/.

Nombre: logout.

Código:

```
url(r'^logout/$', 'proyecto.app.controllers.logout', name='logout'),
```

Método de origen: Ingreso al Sistema.

URL: estudiante/asignaturas/docentes/(\d{1})/.

Nombre: estudiante_asignaturas_docentes.

En (\d{1}) se recibe como parametro el número de carrera.

Código:

```
url(r'^estudiante/asignaturas/docentes/(\d{1})/$',
     'proyecto.app.controllers.estudiante_asignaturas_docentes', name='estudiante_asignaturas_docentes'),
```

Note

Para entender de mejor manera el uso de expresiones regulares en Python revisar: GUÍA PYTHON: EXPRESIONES REGULARES.

Método de origen: Evaluaciones de Directores a Docentes.

URL: director/docentes/(\d{1})/.

Nombre: director_docentes.

En (\d{1}) se recibe un número de carrera de la sesión.

Código:

```
url(r'^director/docentes/(\d{1})/$',  
    'proyecto.app.controllers.director_docentes',  
    name='director_docentes'),
```

Método de origen: Evaluacion de Pares Académicos a Docentes.

URL: pares_academicos/docentes/(\d{1})/.

Nombre: pares_academicos_docentes.

En (\d{1}) se recibe un número de carrera de la sesión.

Código:

```
url(r'^pares_academicos/docentes/(\d{1})/$', 'proyecto.app.controllers.pares_academicos_docentes',  
    name='pares_academicos_docentes'),
```

Método de origen: Listado de Encuestas.

URL:

encuestas/(\?P<id_docente>\d{1,5})/(\?P<id_asignatura>\d{1,5})/(\?P<id_tinformante>\d{1,2})/(\?P<

Nombre: encuestas.

En (\?P<id_docente>\d{1,5}) se recibe el id del docente. Mínimo 1 y máximo 5 dígitos.

En (\?P<id_asignatura>\d{1,5}) se recibe el id de la asignatura. Mínimo 1 y máximo 5 dígitos.

En (\?P<id_tinformante>\d{1,2}) se recibe el id del tipo de informante. Mínimo 1 y máximo 2 dígitos.

En (\?P<id_cuestionario>\d{1,2}) se recibe el id del cuestionario. Mínimo 1 y máximo 2 dígitos.

Código:

```
url(r'^encuestas/(\?P<id_docente>\d{1,5})/(\?P<id_asignatura>\d{1,5})/(\?P<id_tinformante>\d{1,2})/(\?P<id_cuestionario>\d{1,2})/$',  
    'proyecto.app.controllers.encuestas', name='encuestas'),
```

Método de origen: Generación de la Encuesta.

URL: encuesta/responder/(\d{1,5})/.

Nombre: encuestas.

En (\d{1,5}) se recibe el id de la encuesta. Mínimo 1 y máximo 5 dígitos.

Código:

```
url(r'^encuesta/responder/(\d{1,5})/$', 'proyecto.app.controllers.encuesta_responder',  
    name='encuesta_responder'),
```

Método de origen: Grabación de la Encuesta.

Mapeo de URLs

URL: encuesta/grabar/.

Nombre: encuesta_grabar.

Código:

```
url(r'^encuesta/grabar/$', ' proyecto.app.controllers.encuesta_grabar' ,name='encuesta_grabar')
```

Método de origen: Resultados por Carrera.

URL: resultados/carrera/(\d{1})/.

Nombre: resultados_carrera.

En (\d{1}) se recibe número de carrera.

Código:

```
url(r'^resultados/carrera/(\d{1})/$', ' proyecto.app.controllers.resultados_carrera' ,name='re')
```

Método de origen: Menú de resultados por Carrera.

URL: resultados/periodo/(\d{1,2})/.

Nombre: menu_resultados_carrera.

En (\d{1,2}) se recibe el periodo académico. Mínimo 1 y máximo 2 dígitos.

Código:

```
url(r'^resultados/periodo/(\d{1,2})/$', ' proyecto.app.controllers.menu_resultados_carrera' ,name='menu')
```

Método de origen: Presentación de Resultados.

URL: resultados/mostrar/.

Nombre: mostrar_resultados.

Código:

```
url(r'^resultados/mostrar/$', ' proyecto.app.controllers.mostrar_resultados' ,name='mostrar_re')
```

Método de origen: Ofertas SGA.

URL: sga/cargar_ofertas_sga/(\d{1,3})/.

Nombre: cargar_ofertas_sga.

En (\d{1,3}) las ofertas disponibles en el SGA. Mínimo 1 y máximo 3 dígitos.

Código:

```
url(r'^sga/cargar_ofertas_sga/(\d{1,3})/$', ' proyecto.app.controllers.cargar_ofertas_sga' ,name='cargar')
```

Método de origen: Resumen de Evaluaciones.

URL: admin/resumen/evaluaciones/.

Nombre: resumen_evaluaciones.

Código:

```
url(r'^admin/resumen/evaluaciones/$', ' proyecto.app.controllers.resumen_evaluaciones' ,name='resumen')
```

Método de origen: Calculos de Resumen.

URL: admin/resumen/calcular/.

Nombre: calcular_resumen.

Código:

```
url(r'^admin/resumen/calcular/$', 'proyecto.app.controllers.calcular_resumen', name='calcular')
```

Método de origen: Resultados para los Administradores.

URL: admin/app/resultados/.

Nombre: resultados.

Código:

```
url(r'^admin/app/resultados/$', 'proyecto.app.controllers.resultados', name='resultados'),
```

Enrutamiento Handler

Los Formularios

Introducción

Los formularios permiten el ingreso de datos para su procesamiento, ya sea para crear nuevos contenidos, para modificar el contenido que ya está registrado previamente y hasta para realizar búsquedas.

Note

Para ayuda revisar [Curso Django: Los formularios](#) o en la documentación oficial [Working with forms](#).

Importaciones Necesarias

Importación del módulo principal para la creación de formularios:

```
from django import forms
```

Para realizar consultas avanzadas a la BD:

```
from django.db.models import Q
```

Importación de todos los modelos que nos sirven de base para la creación de los formularios:

```
from proyecto.app.models import Configuracion
from proyecto.app.models import PeriodoAcademico
from proyecto.app.models import PeriodoEvaluacion
from proyecto.app.models import DocentePeriodoAcademico
from proyecto.app.models import DireccionCarrera
from proyecto.app.models import AreaSGA
from proyecto.app.models import Asignatura
from proyecto.app.models import EstudianteAsignaturaDocente
from proyecto.app.models import AsignaturaDocente
from proyecto.app.models import Seccion
from proyecto.app.models import Pregunta
```

Resultados Evaluación Desempeño Docente 2012 - 2013

El siguiente formulario es único para los resultados de la Evaluación del Desempeño Docente 2012 - 2013:

```
class ResultadosEDD2013Form(forms.Form):
```

Método inicial:

```
def __init__(self, tabulacion, area, carrera):
    forms.Form.__init__(self)
```

Los Formularios

```
opciones = [(o[0], o[1]) for o in tabulacion.calculos]
self.fields['opciones'] = forms.ChoiceField(widget=forms.RadioSelect(), choices=opciones)
```

En caso de presentarse un error:

```
try:
```

ids_docentes hace referencia a los docentes de la carrera que seleccionó el coordinador en el periodo Academico respectivo:

```
ids_docentes = set([ad.docente.id for ad in AsignaturaDocente.objects.filter(
    docente__periodoAcademico=tabulacion.periodoEvaluacion.periodoAcademico,
    asignatura__carrera=carrera, asignatura__area=area)])
self.fields['docentes'] = forms.ModelChoiceField(
    queryset=DocentePeriodoAcademico.objects.filter(
        Q(periodoAcademico=tabulacion.periodoEvaluacion.periodoAcademico) &
        (Q(id__in=ids_docentes) | Q(carrera=carrera))
    ).order_by('usuario__last_name', 'usuario__first_name')
)
except Exception, ex:
    logg.error("Error en Formulario Resultados EDD2013: {}" .format(ex))
```

Resultados Actividades Adicionales a la Docencia 2012

El siguiente formulario es único para los resultados de la Evaluación de Actividades Adicionales a la Docencia 2012:

```
class ResultadosEAAD2012Form(forms.Form):
```

Método inicial:

```
def __init__(self, tabulacion, area, carrera):
    forms.Form.__init__(self)
    opciones = [(o[0], o[1]) for o in tabulacion.calculos]
    self.fields['opciones'] = forms.ChoiceField(widget=forms.RadioSelect(), choices=opciones)
```

Docentes de la carrera que seleccionó el coordinador en el periodo Academico respectivo:

```
ids_docentes = set([ad.docente.id for ad in AsignaturaDocente.objects.filter(
    docente__periodoAcademico=tabulacion.periodoEvaluacion.periodoAcademico,
    asignatura__carrera=carrera, asignatura__area=area)])
self.fields['docentes'] = forms.ModelChoiceField(
    queryset=DocentePeriodoAcademico.objects.filter(
        Q(periodoAcademico=tabulacion.periodoEvaluacion.periodoAcademico) &
        (Q(id__in=ids_docentes) | Q(carrera=carrera))
    ).order_by('usuario__last_name', 'usuario__first_name')
)
```

Resultados Encuesta de Satisfacción Estudiantil 2012

Al igual que los anteriores, en un formulario Único para los resultados de la Encuesta de Satisfacción Estudiantil 2012 para cualquier tipo de Informante:

```
class ResultadosESE2012Form(forms.Form):
```

Método inicial:

```
def __init__(self, tabulacion, area, carrera):
    forms.Form.__init__(self)
    opciones = [(o[0], o[1]) for o in tabulacion.calculos]
    self.fields['opciones'] = forms.ChoiceField(widget=forms.RadioSelect(), choices=opciones)
```

Docentes de la carrera que seleccionó el coordinador:

```
ids_docentes = set([ad.docente.id for ad in AsignaturaDocente.objects.filter(
    asignatura__carrera=carrera, asignatura__area=area
```

Los Formularios

```
        )))
self.fields['docentes'] = forms.ModelChoiceField(queryset=DocentePeriodoAcademico.objects.filter())
periodoEvaluacion = tabulacion.periodoEvaluacion
if area == u'ACE':
    cuestionario = periodoEvaluacion.cuestionarios.get(informante__tipo=u'InstitutoIdiomas')
elif area == u'MED':
    cuestionario = periodoEvaluacion.cuestionarios.get(informante__tipo=u'EstudianteMED')
```

Y para todas las demás áreas:

```
else:
    cuestionario = periodoEvaluacion.cuestionarios.get(informante__tipo=u'Estudiante')
secciones = Seccion.objects.filter(cuestionario=cuestionario)
campos = [ (s.id, u'{0}. {1}'.format(s.orden, s.titulo)) for s in secciones ]
preguntas = []
for s in secciones:
    preguntas.extend(s.preguntas.all())
```

Por estética en la presentación del SELECT del form de HTML:

```
indicadores = [ (p.id, u'{0}.{1}. {2}'.format(p.seccion.orden, p.orden, p.texto[:150])) for p in PeriodoAcademico.objects.all() ]
self.fields['campos'] = forms.ChoiceField(choices=campos)
self.fields['indicadores'] = forms.ChoiceField(choices=indicadores)
```

Para usar javascript adicional en nuestro formulario:

```
class Media:
    js = ('js/ese2012.js',)
```

Formulario para Selección de Parámetros

La finalidad de este formulario es simplemente la selección de los parámetros iniciales para la presentación de resultados:

```
class ResultadosForm(forms.Form):
    periodo_academico = forms.ModelChoiceField(queryset=PeriodoAcademico.objects.all())
    periodo_academico.label = u'Periodo Académico'
    periodo_evaluacion = forms.ModelChoiceField(PeriodoEvaluacion.objects.none())
    periodo_evaluacion.label = u'Periodo Evaluación'
    area = forms.ModelChoiceField(AreaSGA.objects.none())
    carrera = forms.ModelChoiceField(Asignatura.objects.none())
```

Formulario EstudianteAsignaturaDocente

El siguiente formulario toma como base la *Clase EstudianteAsignaturaDocente* para la obtención de parámetros iniciales:

```
class EstudianteAsignaturaDocenteAdminForm(forms.ModelForm):
    carrera = forms.CharField(widget=forms.TextInput(attrs={'size':'80', 'readonly':'readonly'}))
    semestre = forms.CharField(widget=forms.TextInput(attrs={'readonly':'readonly'}))
    paralelo = forms.CharField(widget=forms.TextInput(attrs={'readonly':'readonly'}))
```

Referencia a la clase:

```
class Meta:
    model = EstudianteAsignaturaDocente
```

Método inicial:

```
def __init__(self, *args, **kwargs):
    super(EstudianteAsignaturaDocenteAdminForm, self).__init__(*args, **kwargs)
    if kwargs.has_key('instance'):
        instance = kwargs['instance']
```

```
    self.initial['carrera'] = instance.carrera
    self.initial['semestre'] = instance.semestre
    self.initial['paralelo'] = instance.paralelo
```

Formulario AsignaturaDocente

El siguiente formulario toma como base la [Clase AsignaturaDocente](#) para la obtención de parámetros iniciales:

```
class AsignaturaDocenteAdminForm(forms.ModelForm):

    carrera = forms.CharField(widget=forms.TextInput(attrs={'size': '80', 'readonly': 'readonly'}))
    semestre = forms.CharField(widget=forms.TextInput(attrs={'readonly': 'readonly'})))
    paralelo = forms.CharField(widget=forms.TextInput(attrs={'readonly': 'readonly'})))
```

Referencia a la clase:

```
class Meta:
    model = AsignaturaDocente
```

Método inicial:

```
def __init__(self, *args, **kwargs):
    super(AsignaturaDocenteAdminForm, self).__init__(*args, **kwargs)
    if kwargs.has_key('instance'):
        instance = kwargs['instance']
        self.initial['carrera'] = instance.carrera
        self.initial['semestre'] = instance.semestre
        self.initial['paralelo'] = instance.paralelo
```

Manejo de Administración

Introducción

El Sitio de administración de Django constituye una Web, limitada a administradores, que permite agregar, editar y eliminar el contenido del sitio. También nos ayuda con la gestión de usuarios si usamos los modelos propios de django

Note

Para ayuda revisar [Curso Django: web de administración](#) o en la documentación oficial: [The Django admin site](#).

Importaciones Necesarias

Importación del módulo principal para la administración:

```
from django.contrib import admin
```

Para el uso de mensajes personalizados:

```
from django.contrib import messages
```

Uso de formularios:

```
from django import forms
```

Importación del paquete completo de *Los Modelos*:

```
from proyecto.app import models
```

Importación del *Formulario EstudianteAsignaturaDocente*:

```
from proyecto.app.forms import EstudianteAsignaturaDocenteAdminForm
```

Importación del *Formulario AsignaturaDocente*:

```
from proyecto.app.forms import AsignaturaDocenteAdminForm
```

Para uso de logs:

```
import logging
log = logging.getLogger('logapp')
```

Clase PreguntaAdmin

Derivada de la *Clase Pregunta* con el fin de agregar nuevas funcionalidades específicas de administración:

```
class PreguntaAdmin(admin.ModelAdmin):
    inlines = (ItemPreguntaEnLinea,)
    fields = ('seccion', 'orden', 'codigo', 'texto', 'descripcion', 'tipo')
    search_fields = ('texto', 'descripcion')
    list_filter = ('seccion_cuestionario_periodoEvaluacion_periodoAcademico',
                   'seccion_cuestionario_periodoEvaluacion', 'seccion_cuestionario')
    list_per_page = 30
```

Método sobreescrito. Se crean y se agregan los items por defecto solo cuando se trata de la creación de una nueva pregunta:

```
def save_model(self, request, obj, form, change):
    if change:
        obj.save()
    return
```

request es un objeto de tipo WSGIRequest:

```
tipo = request.POST['tipo']
```

Solo sí : **Tipo de pregunta: SeleccionUnica**:

```
if tipo=='2':
    longitud = request.POST['longitud']
    numeracion = request.POST['numeracion']
    if longitud == '' or numeracion == '':
        return
    # Si se trata de una pregunta de Seleccion por predeterminada
    obj.save()
    longitud = int(longitud)
    if numeracion == '1':
        for n in range(1, longitud+1):
            item = models.ItemPregunta(texto=str(n), pregunta=obj, orden=n)
            item.save()
    elif numeracion.lower() == 'a':
        for i,n in enumerate(letters):
            item = models.ItemPregunta(texto=n, pregunta=obj, orden=i)
            item.save()
    # Cualquier otro tipo de pregunta
else:
    obj.save()

class Media:
    js = ['js/tiny_mce/tiny_mce.js', 'js/tmce_config.js', ]
```

Clase SeccionAdmin

```
class SeccionAdmin(admin.ModelAdmin):
    inlines = (SubSeccionEnLinea,)
    fields = ('cuestionario', 'codigo', 'nombre', 'titulo', 'descripcion', 'orden', 'pondera')
```

```
search_fields = ('nombre',)
list_filter = ('cuestionario_periodoEvaluacion_periodoAcademico', 'cuestionario_perio
    'cuestionario', 'superseccion')
```

Clase CuestionarioAdmin

```
class CuestionarioAdmin(admin.ModelAdmin):
    actions = ['clonar_cuestionario']
    #####inlines = ('SeccionEnLinea',)
    save_as = True
    list_filter = ('periodoEvaluacion_periodoAcademico', 'periodoEvaluacion')

    # Accion para copiar un cuestionario en el Admin
    def clonar_cuestionario(self, request, queryset):
        cantidad = queryset.count()
        if cantidad == 1:
            objeto = queryset.all()[0]
            objeto.clonar()
            mensaje = u'Se ha clonado satisfactoriamente el Cuestionario'
            self.message_user(request, mensaje)
        else:
            mensaje = u'Debe seleccionar uno solo Cuestionario! Ha seleccionado %s' % (str(cantidad))
            messages.error(request, mensaje)

    clonar_cuestionario.short_description = "Crear Copia de Cuestionario"

    class Media:
        js = ['js/tiny_mce/tiny_mce.js', 'js/tmce_config.js',]
```

Clase PeriodoEvaluacionAdmin

```
class PeriodoEvaluacionAdmin(admin.ModelAdmin):
    filter_horizontal = ('areasSGA',)
    list_filter = ('periodoAcademico',)
```

Clase PeriodoAcademicoAdmin

```
class PeriodoAcademicoAdmin(admin.ModelAdmin):
    filter_horizontal = ('ofertasAcademicasSGA',)
```

Clase EstudiantePeriodoAcademicoAdmin

```
class EstudiantePeriodoAcademicoAdmin(admin.ModelAdmin):
    list_display = ('cedula', '__unicode__', 'periodoAcademico')
    list_per_page = 20
    search_fields = ('usuario_username', 'usuario_cedula', 'usuario_last_name', 'usuario_first_name')
    inlines = (EstudianteAsignaturaDocenteEnLinea,)
    raw_id_fields = ('usuario',)
    list_filter = ('periodoAcademico',)
```

Clase EstudianteAsignaturaDocenteAdmin

```
class EstudianteAsignaturaDocenteAdmin(admin.ModelAdmin):
    raw_id_fields = ('asignaturaDocente', 'estudiante')
    search_fields = ('estudiante_usuario_cedula', 'estudiante_usuario_first_name',
                    'estudiante_usuario_last_name', 'asignaturaDocente_asignatura_carre
                    'asignaturaDocente_asignatura_nombre')
```

```

        'asignaturaDocente_asignatura_semestre' )
list_per_page = 30
list_display = ('estudiante', 'get_asignatura', 'get_area', 'get_carrera', 'get_semestre')
list_filter = ('estudiante_periodoAcademico',)

# Permitir filtros
def lookup_allowed(self, key, value):
    if key in ('asignaturaDocente_asignatura_semestre',):
        return True
    return super(EstudianteAsignaturaDocenteAdmin, self).lookup_allowed(key, value)

```

Clase AsignaturaDocenteAdmin

```

class AsignaturaDocenteAdmin(admin.ModelAdmin):
    actions = ['clonar_asignaturadocente']
    raw_id_fields = ('asignatura', 'docente')
    search_fields = ('docente_usuario_cedula', 'docente_usuario_first_name',
                     'docente_usuario_last_name', 'asignatura_carrera', 'asignatura_nombre',
                     'asignatura_idSGA')
    list_per_page = 30
    list_display = ('get_nombre_corto', 'get_carrera', 'get_semestre', 'get_paralelo')
    fields = ('docente', 'asignatura')
    list_filter = ('docente_periodoAcademico',)

    def validar_clonar(self, request, queryset):
        paralelo = request.POST['paralelo']
        mensaje = ''
        if paralelo == '':
            mensaje = 'Debe especificar un Paralelo'
            messages.error(request, mensaje)
            return False
        cantidad = queryset.count()
        if cantidad > 1:
            mensaje = 'Debe seleccionar uno solo Objeto, ha seleccionado %s' % (str(cantidad))
            messages.error(request, mensaje)
            return False
        asignaturaDocente = queryset.all()[0]
        if paralelo == asignaturaDocente.asignatura.paralelo:
            mensaje = "Se trata del mismo Paralelo, debe clonar la AsignaturaDocente en uno"
            messages.error(request, mensaje)
            return False
        asignatura = asignaturaDocente.asignatura
        # Paralelo que tiene el semestre al cual pertenece AsignaturaDocente
        consulta = models.Asignatura.objects.filter(area=asignatura.area, carrera=asignatura.carrera)
        paralelos = consulta.values_list('paralelo', flat=True).distinct()
        if paralelo not in paralelos:
            mensaje = "No existe este paralelo en el Semestre {0} de la Carrera {1}".format(paralelo.semestre, asignatura.carrera)
            messages.error(request, mensaje)
            return False
        return True

    def clonar_asignaturadocente(self, request, queryset):
        """
        Permite crear una Asignatura con el Docente respectivo pero en otro paralelo
        de la misma carrera y el mismo semestre. Funcionalidad creada especialmente para
        crear aquellas unidades que no existen en el SGA en el caso de las carreras del
        Area de la Salud Humana.
        """
        if self.validar_clonar(request, queryset):

```

```

paralelo = request.POST['paralelo']
asignaturaDocente = queryset.all()[0]
asignatura = asignaturaDocente.asignatura
docente = asignaturaDocente.docente
""" Obtener los estudiantes del paralelo que no tiene esta asignatura """
consulta = models.EstudianteAsignaturaDocente.objects.filter(
    asignaturaDocente_asignatura_area=asignatura.area).filter(
    asignaturaDocente_asignatura_carrera=asignatura.carrera).filter(
    asignaturaDocente_asignatura_semestre=asignatura.semestre).filter(
    asignaturaDocente_asignatura_paralelo=paralelo # El paralelo en el cual se
)
estudiantes = set([ead.estudiante for ead in consulta.all()])
""" Creación de la Nueva Asignatura y AsignaturaDocente """
# Se clona el objeto Asignatura , será una nueva entidad
asignatura.pk = None
id_unidad = asignatura.idSGA.split(':')[0]
asignatura.idSGA = u"{0}:SEDD".format(id_unidad)
asignatura.paralelo = paralelo # El Paralelo en el cual se clonará
asignatura.nombre = u"{0} (Clonado)".format(asignatura.nombre)
# Grabado en la base de datos
asignatura.save()
# Se graba el nuevo Asignatura-Docente
models.AsignaturaDocente.objects.create(docente=docente, asignatura=asignatura)
""" Se asigna la nueva Asignatura-Docente a todos los estudiantes del Paralelo """
for e in estudiantes:
    models.EstudianteAsignaturaDocente.objects.create(estudiante=e, asignaturaDocente=asignatura)
mensaje = "Se agregó la asignatura {0} al paralelo {1} (con {2} estudiantes)".format(asignatura.nombre, paralelo, len(estudiantes))
self.message_user(request, mensaje)

clonar_asignaturadocente.short_description = u'Clonar Asignatura-Docente en otro Paralelo'

```

Clase DocentePeriodoAcademicoAdmin

```

class DocentePeriodoAcademicoAdmin(admin.ModelAdmin):
    list_display = ('cedula', '__unicode__', 'periodoAcademico')
    list_per_page = 20
    search_fields = ('usuario_cedula', 'usuario_last_name', 'usuario_first_name')
    inlines = (AsignaturaDocenteEnLinea,)
    raw_id_fields = ('usuario',)
    list_filter = ('periodoAcademico',)

```

Clase DireccionCarreraAdmin

```

class DireccionCarreraAdmin(admin.ModelAdmin):
    actions = ['actualizar_periodo_academico']
    list_filter = ('director_periodoAcademico',)
    raw_id_fields = ('director',)

    def actualizar_periodo_academico(self, request, queryset):
        """
        Promueve los directores de carrera al presente Periodo Academico
        seleccionando el mismo docente pero del periodo actual.
        """
        periodoAcademico = models.Configuracion.getPeriodoAcademicoActual()
        cont = 0
        for dc in queryset.all():
            try:
                docente_actual = models.DocentePeriodoAcademico.objects.get(

```

```

        usuario=dc.director.usuario, periodoAcademico=periodoAcademico)
    if docente_actual:
        dc.director = docente_actual
        cont = cont + 1
        dc.save()
    except models.DocentePeriodoAcademico.DoesNotExist:
        logg.error('Docente {0} no encontrado en periodo Actual {1}'.format(
            dc.director, periodoAcademico))
    self.message_user(request, 'Actualizadas {0} Direcciones de Carrera'.format(cont))

actualizar_periodo_academico.short_description = u'Actualizar Coordinadores al Periodo A'

```

Clase AsignaturaAdmin

```

class AsignaturaAdmin(admin.ModelAdmin):
    list_display = ('carrera', 'semestre', 'paralelo', '__unicode__')
    list_display_links = ('__unicode__',)
    search_fields = ('idSGA', 'area', 'carrera', 'semestre', 'paralelo', 'nombre',)
    list_filter = ('periodoAcademico', 'area', 'carrera', 'semestre', 'tipo',)
    list_per_page = 20
    # TODO: combobox
    #readonly_fields = ('area', 'carrera', 'semestre', 'paralelo', 'tipo',)
    fieldsets = (
        ('Datos Académicos', {
            'fields': ('area', 'carrera', 'semestre', 'paralelo')
        }),
        ('Asignatura', {
            'fields': ('nombre', 'tipo', 'creditos')
        }),
        ('Duracion', {
            'fields': ('duracion', 'inicio', 'fin', 'periodoAcademico')
        }),
    )

```

Clase UsuarioAdmin

```

class UsuarioAdmin(admin.ModelAdmin):
    search_fields = ('username', 'cedula')
    list_display = ('username', 'cedula', 'get_full_name',)
    list_display_links = ('cedula', 'get_full_name',)
    list_per_page = 20

```

Clase ConfiguracionAdmin

```

class ConfiguracionAdmin(admin.ModelAdmin):
    actions = None

    def has_add_permission(self, request):
        return False

    def has_delete_permission(self, request, obj=None):
        return False

```

Clase EvaluacionAdmin

```
class EvaluacionAdmin(admin.ModelAdmin):
    # Si no se especifica fields no se muestra la vista de modificación
    search_fields = ('estudianteAsignaturaDocente_estudiante_usuario_cedula',
                     'estudianteAsignaturaDocente_asignaturaDocente_docente_usuario_cedula',
                     'docentePeriodoAcademico_usuario_cedula',
                     'directorCarrera_usuario_cedula')
    list_per_page = 30
    list_filter = ('cuestionario_periodoEvaluacion_periodoAcademico', 'cuestionario_períodoEvaluacion_informante',
                   'cuestionario_informante', 'fechaInicio', 'fechaFin',)
    date_hierarchy = 'fechaFin'
    fields = ('cuestionario', 'fechaFin', 'horaFin')
    readonly_fields = ('cuestionario', 'fechaFin', 'horaFin')

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

Clase ResultadosAdmin

```
class ResultadosAdmin(admin.ModelAdmin):
    pass
```

Clase TabulacionAdmin

```
class TabulacionAdmin(admin.ModelAdmin):
    list_filter = ('periodoEvaluacion_periodoAcademico', 'periodoEvaluacion')
```

Clase TipoInformanteAdmin

```
class TipoInformanteAdmin(admin.ModelAdmin):
    pass
```

Clases y Ofertas en EnLinea

A continuación una lista de clases que se caracterizan por adquirir [Los Modelos](#) del paquete `models` con el fin de adicionar ciertos atributos

Clase ItemPreguntaEnLinea

Esta clase obtiene el modelo de la [Clase ItemPregunta](#) y se aumenta ciertos atributos:

```
class ItemPreguntaEnLinea(admin.StackedInline):
    model = models.ItemPregunta
    extra = 1
```

Clase PreguntaEnLinea

Esta clase obtiene el modelo de la [Clase Pregunta](#) y se aumenta ciertos atributos:

```
class PreguntaEnLinea(admin.TabularInline):
    model = models.Pregunta
    extra = 1
```

Clase SubSeccionEnLinea

Esta clase obtiene el modelo de la [Clase Seccion](#) y se aumenta ciertos atributos:

```
class SubSeccionEnLinea(admin.TabularInline):
    model = models.Seccion
    extra = 1
```

```
verbose_name = u'Subsección'  
verbose_name_plural = 'Subsecciones'  
exclude = ('cuestionario',)
```

Clase OfertaAcademicaSGAEnLinea

Esta clase obtiene el modelo de la [Clase OfertaAcademicaSGA](#) y se aumenta ciertos atributos:

```
class OfertaAcademicaSGAEnLinea(admin.TabularInline):  
    model = models.OfertaAcademicaSGA  
    extra = 1
```

Clase EstudianteAsignaturaDocenteEnLinea

Esta clase obtiene el modelo de la [Clase EstudianteAsignaturaDocente](#) y se aumenta ciertos atributos:

```
class EstudianteAsignaturaDocenteEnLinea(admin.StackedInline):  
    model = models.EstudanteAsignaturaDocente  
    extra = 1  
    verbose_name = 'Asignaturas de Estudiante'  
    raw_id_fields = ('asignaturaDocente',)  
    # fields = ('carrera', 'semestre', 'paralelo', 'asignaturaDocente')
```

Clase AsignaturaDocenteEnLinea

Esta clase obtiene el modelo de la [Clase AsignaturaDocente](#) y se aumenta ciertos atributos:

```
class AsignaturaDocenteEnLinea(admin.TabularInline):  
    model = models.AsignaturaDocente  
    form = AsignaturaDocenteAdminForm  
    extra = 1  
    verbose_name = 'Asignaturas de Docente'  
    raw_id_fields = ('asignatura',)
```

Clase ContestacionEnLinea

Esta clase obtiene el modelo de la [Clase Contestacion](#) y se aumenta ciertos atributos:

```
class ContestacionEnLinea(admin.TabularInline):  
    model = models.Contestacion  
    extra = 15  
    verbose_name = 'Respuesta'
```

Agregación de Clases a la Administración

En esta sección se asignan las distintas clases utilizadas para realizar operaciones de administración:

```
admin.site.register(models.Cuestionario, CuestionarioAdmin)  
admin.site.register(models.Seccion, SeccionAdmin)  
admin.site.register(models.Pregunta, PreguntaAdmin)  
admin.site.register(models.PeriodoAcademico, PeriodoAcademicoAdmin)  
admin.site.register(models.PeriodoEvaluacion, PeriodoEvaluacionAdmin)  
admin.site.register(models.EstudantePeriodoAcademico, EstudantePeriodoAcademicoAdmin)  
admin.site.register(models.DocentePeriodoAcademico, DocentePeriodoAcademicoAdmin)  
admin.site.register(models.DireccionCarrera, DireccionCarreraAdmin)  
admin.site.register(models.EstudanteAsignaturaDocente, EstudanteAsignaturaDocenteAdmin)  
admin.site.register(models.AsignaturaDocente, AsignaturaDocenteAdmin)  
admin.site.register(models.Asignatura, AsignaturaAdmin)  
admin.site.register(models.Usuario, UsuarioAdmin)  
admin.site.register(models.Configuracion, ConfiguracionAdmin)  
admin.site.register(models.Evaluacion, EvaluacionAdmin)  
admin.site.register(models.Tabulacion, TabulacionAdmin)
```

```
admin.site.register(models.TipoInformante, TipoInformanteAdmin)
admin.site.register(models.Resultados, ResultadosAdmin)
```