



PERÚ

Ministerio
de Salud

Dirección de Redes Integradas
de Salud Lima Sur

Oficina del Equipo de Trabajo
de Tecnologías De Información

"Año del Bicentenario del
Perú: 200 años de
Independencia"

GESTION DE RECLAMOS


Reclamaciones v 1.2

Trama, Evaluaciones y Reportes

Documentación del código fuente de Reclamaciones
y medidas adoptadas

Por: Ing. Oscar David Mendoza Apaza

```
<meta name="description" content="HTML tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="a">@import "/us.css";</style>
<link rel="stylesheet" type="te" ss" href="/print.css" media="
<link rel="shortcut icon" type="e/ico" href="/favicon.ico">
<link rel="search" type="applic" /opensearch" title="HTML So
htmlsource-search.xml">
<script>
</script>
<script src="/scripts.js" type="t" vascript"></script>
<style type="text/css">
<!--
```





Contenido

1. INTRODUCCIÓN	3
2. PROPÓSITO	3
3. ALCANCE	3
4. CAPA DEL MODELO	3
4.1. Clases GroupMenu	3
4.1.1. Clase GroupMenu (django.db.models.base.Model)	3
4.1.2. Clase Menu (django.db.models.base.Model)	6
4.2. Clase Entidad	10
4.3. Clase Periodo	13
4.4. Clases RubroCalificacion	17
4.5. Clases Usuario	19
4.5.1. Clase Usuario (django.contrib.auth.models.AbstractUser)	19
4.6. Clase ClasificacionCausa	25
4.6.1. Clase ClasificacionCausa (django.db.models.base.Model)	26
4.7. Clases EntidadReclamo	29
4.7.1. Clase EntidadReclamo (django.db.models.base.Model)	29
4.8. Clases DetalleEvaluacionAnexo1	37
4.8.1. Clase DetalleEvaluacionAnexo1 (django.db.models.base.Model)	37
4.8.2. Clase EvaluacionAnexo1 (django.db.models.base.Model)	41
5. CAPA DE LA VISTA	45
5.1. Clases EntidadReclamo	45
5.1.1. Clase EntidadReclamoCreate (django.views.generic.edit.CreateView)	45
5.1.2. Clase EntidadReclamoList (apps.util.generic_filters.views.FilteredListView)	49
5.1.3. Clase EntidadReclamoUpdate (django.views.generic.edit.UpdateView)	53
5.2. Clases User	58
5.2.1. Clase UserPasswordUpdate (django.views.generic.edit.UpdateView)	58
5.2.2. Clase UsuarioCreate (django.views.generic.edit.CreateView)	63
5.2.3. Clase UsuarioList (apps.util.generic_filters.views.FilteredListView)	68
5.2.4. Clase UsuarioUpdate (django.views.generic.edit.UpdateView)	74
5.3. Clases Entidad	79
5.3.1. Clase EntidadCreate (django.views.generic.edit.CreateView)	79
5.3.2. Clase EntidadList (apps.util.generic_filters.views.FilteredListView)	84
5.3.3. Clase EntidadUpdate (django.views.generic.edit.UpdateView)	90

1. INTRODUCCIÓN

Tener una documentación detallada y precisa del código fuente es de gran ayuda al momento de realizar cambios, agregar nuevas funcionalidades y en general aporta a la comprensión de la codificación a los demás programadores del equipo de desarrollo. En este documento se presenta la descripción de las funcionalidades programadas en los módulos de Configuración y Mantenimiento del Sistema, y Gestión de Reclamos y Medidas Adoptadas, módulos que son parte del sistema de Reclamaciones de la DIRIS Lima Sur. El sistema está programado en Python 3.9.1 y el Framework Django 3.1.5, tecnologías actuales que fueron necesarias para desarrollar el sistema entorno Web.

2. PROPÓSITO

Las especificaciones del código fuente son necesarias para compartir la lógica de programación de los programadores, aporta a las buenas prácticas en el desarrollo de software estandarizando una idea de trabajo y que además son ideales para apoyar al Marco de Trabajo SCRUM, adoptado para el desarrollo de este proyecto informático.

3. ALCANCE

A los programadores actuales del Equipo De Equipo de Trabajo de Tecnologías de la Información de la Dirección de Redes Integradas de Salud Lima Sur, y como fuente de conocimiento para la continuidad del proyecto por parte de los futuros trabajadores de la institución

4. CAPA DEL MODELO

4.1. Clases GroupMenu

4.1.1. Clase GroupMenu ([django.db.models.base.Model](#))

[GroupMenu](#) (* args, ** kwargs)

[GroupMenu](#) (id, grupo)

Orden de resolución del método:

[GroupMenu](#)

[django.db.models.base.Model](#)

[objeto incorporado](#)

Métodos definidos aquí:

`__str__` (seft)

Devuelve str (self).

id = <objeto django.db.models.query_utils.DeferredAttribute>

Descriptores de datos definidos aquí:

grupo

Acceso al objeto relacionado en el lado delantero de una relación uno a uno.

En el ejemplo ::

```
class Restaurant (Model):  
    place = OneToOneField (Place, related_name  
= 'restaurant')
```

```
`` Restaurant.place `` es una instancia de ``  
ForwardOneToOneDescriptor ``.
```

Identificación del grupo

menús

Acceso al administrador de objetos relacionados en los lados delantero y trasero de una relación de varios a varios.

En el ejemplo ::

```
class Pizza (Modelo):  
    toppings = ManyToManyField (Topping,  
related_name = 'pizzas')
```

```
`` Pizza.toppings `` y `` Topping.pizzas `` son  
instancias de `` ManyToManyDescriptor`` .
```

La mayor parte de la implementación se delega a una clase de administrador definida dinámicamente construida por `` create_forward_many_to_many_manager () `` definido a continuación.

Datos y otros atributos definidos aquí:

DoesNotExist = <clase

'setup.models.menu.GroupMenu.DoesNotExist'>

MultipleObjectsReturned = <clase

'setup.models.menu.GroupMenu.MultipleObjectsReturned'>

objetos = <objeto django.db.models.manager.Manager>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, otro)

Devuelve self == valor.

__getstate__ (self)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (self)

Devuelve repr (self).

__setstate__ (self, state)

limpio (yo)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) ()

en cada campo por sí mismo. **clean_fields**. Cualquier ValidationError generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por NON_FIELD_ERRORS.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un ValidationError que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)

Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo. Genere un ValidationError para cualquier error que ocurra.

get_deferred_fields (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia, o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de att de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save (self, force_insert = False, force_update = False, using = None, update_fields = None)

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para

backends que no sean SQL),
respectivamente. Normalmente, no deben
configurarse.

save_base (self, raw = False, force_insert = False, force_update =
False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse
solo una vez por guardado,
pero también deben hacerse en guardados sin
formato. Esto incluye algunas
comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no
guarde ningún
modelo padre y que no haga ningún cambio en los
valores antes de guardar. Esto
es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta
instancia. Si el campo es
una clave externa, devuelve el valor de id en lugar
del objeto. Si no hay
ningún objeto Field con este nombre en el modelo,
devuelva el
valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en
el serializador o en la salida del formulario,
por ejemplo). Normalmente, solo accedería al
atributo directamente
y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y
genere ValidationError si alguna
falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :
comprobar (** kwargs) de [django.db.models.base.ModelBase](#)
from_db (db, field_names,
values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están
definidas)

__weakref__

lista de referencias débiles al objeto (si está
definido)

paquete

4.1.2. **Clase Menu** ([django.db.models.base.Model](#))

[Menú](#) (* args, ** kwargs)

[Menú](#) (id, nombre, icono, url, título, padre, tipo)

Orden de resolución del método:

[Menú](#)

[django.db.models.base.Model](#) [objeto incorporado](#)

Métodos definidos aquí:

```
__str__ (yo)
    Devuelve str (self).
get_type_display = _method (self, *, field =
<django.db.models.fields.IntegerField: type>)
icon = <django.db.models.query_utils.DeferredAttribute object> id =
<django.db.models.query_utils.DeferredAttribute object> name =
<django.db.models.query_utils.DeferredAttribute object> title =
<django.db.models.query_utils.DeferredAttribute object> type =
<django.db.models.query_utils.DeferredAttribute object> url =
<django.db.models.query_utils.DeferredAttribute object>
```

Descriptores de datos definidos aquí:

groupmenu_set

Acceso al administrador de objetos relacionados en los lados delantero y trasero de una relación de varios a varios.

En el ejemplo ::

```
class Pizza (Modelo):
    toppings = ManyToManyField (Topping,
related_name = 'pizzas')

`` Pizza.toppings `` y `` Topping.pizzas `` son
instancias de `` ManyToManyDescriptor ``.
```

La mayor parte de la implementación se delega a una clase de administrador definida dinámicamente construida por `` create_forward_many_to_many_manager () `` definido a continuación.

parent

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent `` es una instancia de ``
ForwardManyToOneDescriptor ``.
```

Identificación de los padres

Datos y otros atributos definidos aquí:

```
DoesNotExist = <clase 'setup.models.menu.Menu.DoesNotExist'>
MultipleObjectsReturned = <clase
'setup.models.menu.Menu.MultipleObjectsReturned'>
objetos = <objeto django.db.models.manager.Manager>
```


Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, other)

Devuelve self == valor.

__getstate__ (yo)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda ([escriba](#) (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (self)

Devuelve repr (self).

__setstate__ (self, state)

limpio (self)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) () en cada campo por sí mismo. **clean_fields** . Cualquier ValidationError generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por NON_FIELD_ERRORS.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un ValidationError que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)

Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo. Genere un ValidationError para cualquier error que ocurra.

get_deferred_fields (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia , o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de att de campo. Si

los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save (self, force_insert = False, force_update = False, using = None, update_fields = None)

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para backends que no sean SQL), respectivamente. Normalmente, no deben configurarse.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)
from_db (db, field_names,
values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

Datos

TIPOS = ((0, 'PERFIL_PACIENTE'), (1, 'ADMINISTRADOR'))

4.2. Clase Entidad

4.2.1. Clase Entidad ([django.db.models.base.Model](#))

[Entidad](#) (* args, ** kwargs)[Entidad](#) (id, nombre, codigo, categoria, ris, tipo)

Orden de resolución del método:

[Entidad](#)[django.db.models.base.Model](#)[objeto incorporado](#)

Métodos definidos aquí:

__str__ (self)

Devuelve str (self).

categoria = <objetodjango.db.models.query_utils.DeferredAttribute> **codigo** = <objeto

django.db.models.query_utils.DeferredAttribute>

get_categoria_display = _method (self, *, field =

<django.db.models.fields.PositiveIntegerField: categoria>)

get_ris_display = _method (self, *, field =

<django.db.models.fields.PositiveIntegerField: ris>)

get_tipo_display = _method (self, *, field =

<django.db.models.fields.PositiveIntegerField: tipo>)

id = <django.db.models.query_utils.DeferredAttributeobject> **nombre** = <django.db.models.query_utils.DeferredAttributeobject> **ris** = <django.db.models.query_utils.DeferredAttributeobject> **tipo** = <django.db.models .query_utils.DeferredAttribute

object>

Datos y otros atributos definidos aquí:

DoesNotExist = <clase 'setup.models.entidad.Entidad.DoesNotExist'>**MultipleObjectsReturned** = <clase

'setup.models.entidad.Entidad.MultipleObjectsReturned'>

objetos = <objeto django.db.models.manager.Manager>Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, otro)
Devuelve self == valor.

__getstate__ (self)
Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)
Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)
Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)
Ayudante para pepinillos.

__repr__ (self)
Devuelve repr (self).

__setstate__ (self, state)

limpio (self)
Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) () en cada campo por sí mismo. **clean_fields** . Cualquier `ValidationError` generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por `NON_FIELD_ERRORS`.

clean_fields (self, exclude = None)
Limpie todos los campos y genere un `ValidationError` que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)
Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo. Genere un `ValidationError` para cualquier error que ocurra.

get_deferred_fields (self)
Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)
Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia, o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro `using` anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de `att` de campo. Si los campos son `Ninguno`, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save (self, force_insert = False, force_update = False, using = None, update_fields = None)

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para backends que no sean SQL), respectivamente. Normalmente, no deben configurarse.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)

from_db (db, field_names,

values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

Datos

CATEGORIAS = ((1, 'I-1'), (2, 'I-2'), (3, 'I-3'), (4, 'I-4'))

RIS = ((1, ' RIS BCO-CHO-SCO '), (2, ' RIS LURIN Y BALNERARIOS '), (3, ' RIS PACHACAMAC '), (4, ' RIS SJM '), (5, ' RIS VES '), (6, ' RIS VMT'), (7, 'DIRIS SEDE ADMINISTRATIVA'))

TIPOS = ((1, 'IPRESS'), (2, 'UGIPRESS'), (3, 'IAFAS'))

4.3. Clase Periodo

4.3.1. Clase Periodo ([django.db.models.base.Model](#))

[Periodo](#) (* args, ** kwargs)

[Periodo](#) (id, periodo, fecha_inicio, fecha_fin, estado)

Orden de resolución del método:

[Periodo](#)

[django.db.models.base.Model](#)

[objeto incorporado](#)

Métodos definidos aquí:

__str__ (self)

Devuelve str (self).

estado = <objeto

django.db.models.query_utils.DeferredAttribute> **fecha_fin** =

<objeto django.db.models.query_utils.DeferredAttribute > **fecha_inicio** =

<objeto django.db.models.query_utils.DeferredAttribute>

get_next_by_fecha_fin = _method (self, *, field =

<django.db.models.fields.DateField: fecha_fin>, is_next = True, ** kwargs)

get_next_by_fecha_inicio = _method (self, *, field =

<django.db.models.fields.DateField: fecha_inicio>, is_next = True, ** kwargs)

get_previous_by_fecha_fin = _method (self, *, field =

<django.db.models.fields.DateField: fecha_fin>, is_next = False, ** kwargs)

```
get_previous_by_fecha_inicio = _method (self, *, field =  
<django.db.models.fields.DateField: fecha_inicio>, is_next = False, ** kwargs)
```

```
id = <objeto django.db.models.query_utils.DeferredAttribute> periodo =  
<objeto django.db.models.query_utils.DeferredAttribute>
```

```
guardar (uno mismo, * argumentos, ** kwargs)
```

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para backends que no sean SQL), respectivamente. Normalmente, no deben configurarse.

Datos y otros atributos definidos aquí:

```
DoesNotExist = <clase 'setup.models.periodo.Periodo.DoesNotExist'>
```

```
MultipleObjectsReturned = <clase  
'setup.models.periodo.Periodo.MultipleObjectsReturned'>
```

```
objetos = <objeto django.db.models.manager.Manager>
```

Métodos heredados de [django.db.models.base.Model](#) :

```
__eq__ (self, otro)
```

Devuelve self == valor.

```
__getstate__ (self)
```

Gancho para permitir elegir los atributos a encurtir.

```
__hash__ (self)
```

Devuelve hash (self).

```
__init__ (self, * argumentos, ** kwargs)
```

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

```
__reduce__ (auto)
```

Ayudante para pepinillos.

```
__repr__ (self)
```

Devuelve repr (self).

__setstate__ (self, state)

limpio (yo)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) () en cada campo por sí mismo. **clean_fields** . Cualquier `ValidationError` generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por `NON_FIELD_ERRORS`.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un `ValidationError` que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)

Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo.
Genere un `ValidationError` para cualquier error que ocurra.

get_deferred_fields (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia , o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro `using` anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de `att` de campo. Si los campos son `Ninguno`, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia,

la carga diferida
del campo llamará a este método.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)

from_db (db, field_names, values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

4.4. Clases RubroCalificacion

4.4.1. Clase RubroCalificacion ([django.db.models.base.Model](#))

[RubroCalificacion](#) (* args, ** kwargs)

[RubroCalificacion](#) (id, orden, rubro)

Orden de resolución del método:

[RubroCalificación](#)
[django.db.models.base.Model](#)
[objeto incorporado](#)

Métodos definidos aquí:

__str__ (self)
Devuelve str (self).
id = <objeto django.db.models.query_utils.DeferredAttribute> **orden** =
<objeto django.db.models.query_utils.DeferredAttribute> **rubro** =
<objeto django.db.models.query_utils.DeferredAttribute>

Datos y otros atributos definidos aquí:

DoesNotExist = <clase
'setup.models.rubro_calificacion.RubroCalificacion.DoesNotExist'>
MultipleObjectsReturned = <clase
'setup.models.rubro_calificacion.RubroCalificacion.MultipleObjectsReturned'>
objetos = <objeto django.db.models.manager.Manager>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, other)
Devuelve self == valor.
__getstate__ (self)
Gancho para permitir elegir los atributos a encurtir.
__hash__ (self)
Devuelve hash (self).
__init__ (self, * argumentos, ** kwargs)
Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.
__reduce__ (auto)

Ayudante para pepinillos.

__repr__(self)
Devuelve repr (self).

__setstate__(self, state)

limpio(self)
Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) () en cada campo por sí mismo. **clean_fields** . Cualquier `ValidationError` generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por `NON_FIELD_ERRORS`.

clean_fields(self, exclude = None)
Limpie todos los campos y genere un `ValidationError` que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message(self, lookup_type, field_name, unique_for)

eliminar(self, using = None, keep_parents = False)

full_clean(self, exclude = None, validate_unique = True)
Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo. Genere un `ValidationError` para cualquier error que ocurra.

get_deferred_fields(uno mismo)
Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save(self, field)

refresh_from_db(self, using = None, fields = None)
Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia , o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de att de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save(self, force_insert = False, force_update = False, using = None, update_fields = None)
Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en

que el "guardar" debe ser una inserción o actualización SQL (o equivalente para backends que no sean SQL), respectivamente. Normalmente, no deben configurarse.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)
from_db (db, field_names,
values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

4.5. Clases Usuario

4.5.1. Clase Usuario ([django.contrib.auth.models.AbstractUser](#))

[Usuario](#) (* args, ** kwargs)

[Usuario](#) (id, contraseña, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, entidad, document, celular)

Orden de resolución del método:

[Usuario](#)

[django.contrib.auth.models.AbstractUser](#)

[django.contrib.auth.base_user.AbstractBaseUser](#)

[django.contrib.auth.models.PermissionsMixin](#)

[django.db.models.base.Model](#)

[objeto incorporado](#)

Métodos definidos aquí:

__str__ (self)

Devuelve str (self).

celular = <objeto

django.db.models.query_utils.DeferredAttribute> **document** = <objeto

django.db.models.query_utils.DeferredAttribute>

get_next_by_date_joined = _method (self, *, field =

<django.db.models.fields.DateTimeField: date_joined>, is_next = True, ** kwargs)

get_previous_by_date_joined = _method (self, *, field =

<django.db.models.fields.DateTimeField: date_joined>, is_next = False, ** kwargs)

id = <objeto django.db.models.query_utils.DeferredAttribute>

Descriptores de datos definidos aquí:

entidad

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')
```

`` Child.parent `` es una instancia de `` ForwardManyToManyDescriptor ``.

entidad_id

grupos

Acceso al administrador de objetos relacionados en los lados delantero y trasero de una relación de varios a varios.

En el ejemplo ::

```
class Pizza (Modelo):
    toppings = ManyToManyField (Topping,
related_name = 'pizzas')
```

```
` Pizza.toppings ' y ` Topping.pizzas ' son  
instancias de ` ManyToManyDescriptor` .
```

La mayor parte de la implementación se delega a una clase de administrador definida dinámicamente construida por `create_forward_many_to_many_manager ()` ' definido a continuación.

Permisos de usuario

Acceso al administrador de objetos relacionados en los lados delantero y trasero de una relación de varios a varios.

En el ejemplo ::

```
class Pizza (Modelo):  
    toppings = ManyToManyField (Topping,  
related_name = 'pizzas')
```

```
` Pizza.toppings ' y ` Topping.pizzas ' son  
instancias de ` ManyToManyDescriptor` .
```

La mayor parte de la implementación se delega a una clase de administrador definida dinámicamente construida por `create_forward_many_to_many_manager ()` ' definido a continuación.

Datos y otros atributos definidos aquí:

DoesNotExist = <clase 'setup.models.usuario.Usuario.DoesNotExist'>

MultipleObjectsReturned = <clase
'setup.models.usuario.Usuario.MultipleObjectsReturned'>

Métodos heredados de [django.contrib.auth.models.AbstractUser](#) :

Clean (self)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a `clean ()`

en cada campo por sí mismo. **clean_fields** . Cualquier `ValidationError` generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por `NON_FIELD_ERRORS`.

date_joined = <objeto

`django.db.models.query_utils.DeferredAttribute`> **email** = <objeto

`django.db.models.query_utils.DeferredAttribute`>

email_user (self, asunto, mensaje, from_email = Ninguno, ** kwargs)

Envíe un correo electrónico a este usuario.

first_name = <objeto `django.db.models.query_utils.DeferredAttribute`>

get_full_name (self)

Devuelve `first_name` más `last_name`, con un espacio entre ellos.

get_short_name (self)

Devuelve el nombre corto del usuario.

```
is_staff = <django.db.models.query_utils.DeferredAttribute
object> last_name = <django.db.models.query_utils.DeferredAttribute
object> username = <django.db.models.query_utils.DeferredAttribute
object>
```

Datos y otros atributos heredados
de [django.contrib.auth.models.AbstractUser](#) :

```
EMAIL_FIELD = 'correo electrónico'
Meta = <clase 'django.contrib.auth.models.AbstractUser.Meta'>
REQUIRED_FIELDS = ['correo electrónico']
USERNAME_FIELD = 'nombre de usuario'
objetos = <objeto django.contrib.auth.models.UserManager>
username_validator = <objeto
django.contrib.auth.validators.UnicodeUsernameValidator>
```

Métodos heredados de [django.contrib.auth.base_user.AbstractBaseUser](#) :

```
check_password (self, raw_password)
    Devuelve un valor booleano que indique si
    raw_password era correcto. Maneja
    formatos hash entre bastidores.

get_session_auth_hash (self)
    Devuelve un HMAC del campo de contraseña.

get_username (self)
    Devuelve el nombre de usuario de este usuario.

has_usable_password (self)
    Devuelve False si se ha llamado
    a set\_unusable\_password () para este usuario.

last_login = <objeto django.db.models.query_utils.DeferredAttribute>
natural_key (self)
contraseña = <objeto django.db.models.query_utils.DeferredAttribute>
guardar (uno mismo, * argumentos, ** kwargs)
    Guarde la instancia actual. Reemplace esto en una
    subclase si desea
    controlar el proceso de guardado.

    Los parámetros 'force_insert' y 'force_update'
    pueden usarse para insistir en
    que el "guardar" debe ser una inserción o
    actualización SQL (o equivalente para
    backends que no sean SQL),
    respectivamente. Normalmente, no deben configurarse.

set_password (self, raw_password)
set_unusable_password (auto)
```

Métodos de clase heredados
de [django.contrib.auth.base_user.AbstractBaseUser](#) :

```
get_email_field_name () de django.db.models.base.ModelBase
normalize_username (nombre de
usuario) de django.db.models.base.ModelBase
```

Descriptores de datos heredados
de [django.contrib.auth.base_user.AbstractBaseUser](#) :
es_anónimo

Siempre devuelve False. Esta es una forma de comparar objetos de usuario con usuarios anónimos.

is_authenticated

Siempre devuelve True. Esta es una forma de saber si el usuario se ha autenticado en las plantillas.

Datos y otros atributos heredados

de [django.contrib.auth.base_user.AbstractBaseUser](#) :

is_active = Verdadero

Métodos heredados de [django.contrib.auth.models.PermissionsMixin](#) :

get_all_permissions (self, obj = None)

get_group_permissions (self, obj = None)

Devuelve una lista de cadenas de permisos que este usuario tiene a través de sus grupos. Consulta todos los backends de autenticación disponibles. Si se pasa un objeto, devuelva solo los permisos que coincidan con este objeto.

get_user_permissions (self, obj = None)

Devuelve una lista de las cadenas de permisos que este usuario tiene directamente. Consulta todos los backends de autenticación disponibles. Si se pasa un objeto, devuelva solo los permisos que coincidan con este objeto.

has_module_perms (self, app_label)

Devuelve True si el usuario tiene algún permiso en la etiqueta de la aplicación dada. Utilice una lógica similar a [has_perm](#) (), anterior.

has_perm (self, permanente, obj = Ninguno)

Devuelve True si el usuario tiene el permiso especificado. Consulta todos los backends de autenticación disponibles, pero regresa inmediatamente si alguno de los backend devuelve True. Por lo tanto, se supone que un usuario que tiene permiso de un único backend de autenticación tiene permiso en general. Si se proporciona un objeto, verifique los permisos para ese objeto.

has_perms (self, perm_list, obj = None)

Devuelve True si el usuario tiene cada uno de los permisos especificados. Si se pasa el objeto, compruebe si el usuario tiene todos los permisos necesarios para ello.

is_superuser = <objeto

django.db.models.query_utils.DeferredAttribute>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, otro)

Devuelve self == valor.

__getstate__ (yo)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (self)

Devuelve repr (self).

__setstate__ (self, state)**clean_fields** (self, exclude = None)

Limpie todos los campos y genere un ValidationError que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)**eliminar** (self, using = None, keep_parents = False)**full_clean** (self, exclude = None, validate_unique = True)Llame a [clean_fields](#) (), [clean](#) () y [validate_unique](#) () en el modelo. Genere un ValidationError para cualquier error que ocurra.**get_deferred_fields** (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)**refresh_from_db** (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia , o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de att de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los

valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelve el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)**validate_unique** (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)

from_db (db, field_names,

values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

Funciones

calcular_edad (nacido)

Datos

CIVIL_STATUS = ((0, 'SOLTERO'), (1, 'CASADO'), (2, 'DIVORCIADO'), (3, 'VIUDO'))

GRADO = ((0, 'SIN ESTUDIOS'), (1, 'PRIMARIA COMPLETA'), (2, 'PRIMARIA INCOMPLETA'), (3, 'SECUNDARIA COMPLETA'), (4, 'SECUNDARIA INCOMPLETA'), (5, 'ESTUDIOS SUPERIORES'))

SEXES = ((0, 'FEMENINO '), (1, ' MASCULINO '))

TIPOS = ((' DNI ', ' DNI '), (' PASSPORD ', ' PASAPORTE '))

4.6. Clase ClasificacionCausa

4.6.1. Clase **ClasificacionCausa** ([django.db.models.base.Model](#))

[ClasificacionCausa](#) (* args, ** kwargs)

[ClasificacionCausa](#) (id, codigo, categoria, causa, definicion, definicion_corta, created_at, updated_at)

Orden de resolución del método:

[ClasificacionCausa](#)

[django.db.models.base.Model](#)

objeto incorporado

Métodos definidos aquí:

__str__ (self)

Devuelve str (self).

categoria = <objeto

django.db.models.query_utils.DeferredAttribute> **causa** = <objeto

django.db.models.query_utils.DeferredAttribute> **codigo** =

<objeto **django.db.models.query_utils.DeferredAttribute** > **created_at**

= <django.db.models .query_utils.DeferredAttribute object> **definicion** =

<django.db.models.query_utils.DeferredAttribute

object> **definicion_corta** =

<django.db.models.query_utils.DeferredAttribute object>

get_categoria_display = _method (self, *, field =

<django.db.models.fields.PositiveIntegerField: categoria>)

get_next_by_created_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: created_at>, is_next = True, **
kwargs)

get_next_by_updated_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: updated_at>, is_next = True,
** kwargs)

get_previous_by_created_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: created_at>, is_next = False, **
kwargs)

get_previous_by_updated_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: updated_at>, is_next = False,
** kwargs)

id = <objeto

django.db.models.query_utils.DeferredAttribute> **updated_at** = <objeto

django.db.models.query_utils.DeferredAttribute>

Datos y otros atributos definidos aquí:

DoesNotExist = <clase

'apps.reclamo.models.clasificacion_causa.ClasificacionCausa.DoesNotExist'
>

MultipleObjectsReturned = <clase

'apps.reclamo.models.clasificacion_causa.ClasificacionCausa.MultipleOb
jectsReturned'>

objetos = <objeto django.db.models.manager.Manager>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, other)

Devuelve self == valor.

__getstate__ (self)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (yo)

Devuelve repr (self).

__setstate__ (self, state)

limpio (yo)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado

a [clean](#) ()

en cada campo por sí mismo. **clean_fields**. Cualquier

ValidationError generado

por este método no se asociará con un campo en

particular; tendrá

una asociación de caso especial con el campo definido

por NON_FIELD_ERRORS.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un ValidationError

que contenga un dictado

de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)

Llame a [clean_fields](#) (), [clean](#) ()

y [validate_unique](#) () en el modelo.

Genere un ValidationError para cualquier error que ocurra.

get_deferred_fields (self)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se

cargó esta instancia , o por el enrutador de lectura

si esta instancia no se cargó desde

ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué

campos volver a cargar. Los campos

deben ser iterables de nombres de att de campo. Si

los campos son Ninguno, se recargan

todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save (self, force_insert = False, force_update = False, using = None, update_fields = None)

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para backends que no sean SQL), respectivamente. Normalmente, no deben configurarse.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo.

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)
from_db (db, field_names,
values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

`__weakref__``lista de referencias débiles al objeto (si está
definido)``paquete`

Datos

DERECHOS = ((1, 'ACCESO A LOS SERVICIOS DE SALUD'), (2, 'ACCESO A LA INFORMACIÓN'), (3, 'ATENCIÓN Y RECUPERACIÓN DE LA SALUD'), (4, 'CONSENTIMIENTO INFORMADO'), (5, 'PROTECCIÓN DE DERECHOS'), (6, 'OTROS'))

4.7. Clases EntidadReclamo

4.7.1. Clase EntidadReclamo ([django.db.models.base.Model](#))

[EntidadReclamo](#) (* args, ** kwargs)

[EntidadReclamo](#)(Id, entidad, periodo_declaracion, tipo_institucion, codigo_administrado, medio_presentacion, codigo_registro, tipo_documento_usuario, numero_documento_usuario, razon_social_usuario, nombres_usuario, apellido_paterno_usuario, apellido_materno_usuario, tipo_documento_presenta, numero_documento_presenta, razon_social_presenta, nombres_presenta, apellido_paterno_presenta, apellido_materno_presenta, autorizacion_notificacion_correo, correo_presenta, domicilio_presenta, celular_presenta, medio_recepcion_reclamo, fecha_reclamo, detalle_reclamo, servicio_hecho_reclamo, competencia_reclamo, clasificacion_reclamo_1, clasificacion_reclamo_2, clasificacion_reclamo_3, estado_reclamo, codigo_reclamo_primigenio, etapa_reclamo, tipo_administrado_traslado, codigo_administrado_deriva, resultado_reclamo, motivado_conclusion_resulta, comunicacion_resultado_reclamo, fecha_notificacion, es_mismo_usuario_afectado, created_at, updated_at, created_ip, updated_ip)

Orden de resolución del método:

[EntidadReclamo](#)

[django.db.models.base.Model](#)

[objeto incorporado](#)

Métodos definidos aquí:

apellido_materno_presenta =

<django.db.models.query_utils.DeferredAttribute

object> **apellido_materno_usuario** =

<django.db.models.query_utils.DeferredAttribute

object> **apellido_paterno_presenta** =

<django.db.models.query_utils.DeferredAttribute

object> **apellido_paterno_usuario** = <django Objeto

.db.models.query_utils.DeferredAttribute> **autorizacion_notificacion_co**

rreo = <objeto

django.db.models.query_utils.DeferredAttribute> **celular_presenta** =


```
<django.db.models.query_utils.DeferredAttribute
objeto> codigo_administrado = <django.DeferredAttribute objeto
objeto> codigo_administrado_deriva=
<django.db.models.query_utils.DeferredAttribute
objeto> codigo_reclamo_primigenio =
<django.db.models.query_utils.DeferredAttribute
object> codigo_registro =
<django.db.models.query_utils.DeferredAttribute
object> competencia_reclamob =
<django.db.models.query_utils.DeferredAttribute object> competencia_r
eclamob =
<django.db.models.query_utils.DeferredAttribute object> codigo_registro
= <django.db.models.query_utils.DeferredAttribute
object> competencia_reclamob =
<django.db.models.query_utils.Objective objeto
query_utils.DeferredAttribute> comunicacion_resultado_reclamo =
<objeto
django.db.models.query_utils.DeferredAttribute> correo_presenta =
<objeto django.db.models.query_utils.DeferredAttribute > created_at
= <django.db.models.query_utils. object> created_tiptributes . Objeto
django.db.models.query_utils.DeferredAttribute> detalle_reclamo=
<objeto
django.db.models.query_utils.DeferredAttribute> domicilio_presenta =
<objeto django.db.models.query_utils.DeferredAttribute > es_mismo_
usuario_afectado = <django.db.models.query_utils.DeferredAttribute
object> estado_reclamob =
<django.db.models.query_utils.DeferredAttribute object> estado_reclamo
b = <django.db.models.query_utils.DeferredAttribute
object> es_mismo_usuario_afectado =
<django.db.models.query_utils.DeferredAttribute object> estado_reclamo
b = <django.db. query_utils.DeferredAttribute object> etapa_reclamo =
<django.db.models.query_utils.DeferredAttribute
object> fecha_notificacion =
<django.db.models.query_utils.DeferredAttribute
object> fecha_reclamo =
<django.db.models.query_utilidos.DeferredAttribute
object> fecha_notificacion =
< django.db.models.query_utils.DeferredAttribute
object> fecha_reclamo =
<django.db.models.query_utilidos.DeferredAttribute objeto Objeto
django.db.models.query_utils.DeferredAttribute>
get_autorizacion_notificacion_correo_display = _method (self, *, field
= <django.db.models.fields.IntegerField:
autorizacion_notificacion_correo>)
get_competencia_reclamo_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: competencia_reclamo>)
get_comunicacion_resultado_reclamo_display = _method (self, *, field
= <django.db.models.fields.PositiveIntegerField:
comunicacion_resultado_reclamo>)
get_estado (yo)
```

```
get_estado_reclamo_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: estado_reclamo>)
get_etapa_reclamo_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: etapa_reclamo>)
get_medio_presentacion_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: medio_presentacion>)
get_medio_recepcion_reclamo_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField:
medio_recepcion_reclamo>)
get_motivo_conclusion_anticipada_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField:
motivo_conclusion_anticipada>)
get_next_by_created_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: created_at>, is_next = True, **
kwargs)
get_next_by_fecha_reclamo = _method (self, *, field =
<django.db.models.fields.DateField: fecha_reclamo>, is_next = True, **
kwargs)
get_next_by_updated_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: updated_at>, is_next = True, **
kwargs)
get_previous_by_created_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: created_at>, is_next = False, **
kwargs)
get_previous_by_fecha_reclamo = _method (self, *, field =
<django.db.models.fields.DateField: fecha_reclamo>, is_next = False, **
kwargs)
get_previous_by_updated_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: updated_at>, is_next = False, **
kwargs)
get_resultado_reclamo_display = _method (self, *, field =
<django.db.models.fields.IntegerField: resultado_reclamo>)
get_servicio_hecho_reclamo_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: servicio_hecho_reclamo>)
get_tipo_administrado_traslado_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField:
tipo_administrado_traslado>)
get_tipo_documento_presenta_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField:
tipo_documento_presenta>)
get_tipo_documento_usuario_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: tipo_documento_usuario>)
get_tipo_institucion_display = _method (self, *, field =
<django.db.models.fields.PositiveIntegerField: tipo_institucion>)
get_usuario (yo)
ID = <django.db.models.query_utils.DeferredAttribute
object> medio_presentacion =
<django.db.models.query_utils.DeferredAttribute
object> medio_recepcion_reclamo =
<django.db.models.query_utils.DeferredAttribute
```

```
objeto> motivo_conclusion_anticipada = <django.db.models Objeto
.query_utils.DeferredAttribute> nombres_presenta = <objeto
django.db.models.query_utils.DeferredAttribute> nombres_usuario =
<objeto django.db.models.query_utils.DeferredAttribute > numero_documento_presenta = <objeto django.db.modelsDeferredAttributes .
<objeto
django.db.models.query_utils.DeferredAttribute> razon_social_presenta
= <objeto
django.db.models.query_utils.DeferredAttribute> razon_social_usuario =
<objeto
django.db.models.query_utils.DeferredAttribute> resultado_reclamo =
<objeto django.db.models.query_utils.DeferredAttribute>
guardar (uno mismo, * argumentos, ** kwargs)
    Guarde la instancia actual. Reemplace esto en una
    subclase si desea
    controlar el proceso de guardado.

    Los parámetros 'force_insert' y 'force_update' pueden
    usarse para insistir en
    que el "guardar" debe ser una inserción o
    actualización SQL (o equivalente para
    backends que no sean SQL),
    respectivamente. Normalmente, no deben configurarse.

servicio_hecho_reclamo =
<django.db.models.query_utils.DeferredAttribute
object> tipo_administrado_traslado =
<django.db.models.query_utils.DeferredAttribute
object> tipo_documento_presenta =
<django.db.models.query_utils.DeferredAttribute
objeto> tipo_documento_usuario = <django.db.models Objeto
.query_utils.DeferredAttribute> tipo_institucion = <objeto
django.db.models.query_utils.DeferredAttribute> updated_at =
<objeto django.db.models.query_utils.DeferredAttribute > updated_ip
= <objeto django.db.models.query_utils.DeferredAttribute
```

Descriptores de datos definidos aquí:

clasificacion_reclamo_1

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent `` es una instancia de ``
ForwardManyToOneDescriptor ``.
```

clasificacion_reclamo_1_id

clasificacion_reclamo_2

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o

uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

clasificacion_reclamo_2_id

clasificacion_reclamo_3

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

clasificacion_reclamo_3_id

entidad

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

entidad_id

periodo_declaracion

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

periodo_declaracion_id

Datos y otros atributos definidos aquí:

```
DoesNotExist = <clase  
'apps.reclamo.models.entidad_reclamo.EntidadReclamo.DoesNotExist'>  
MultipleObjectsReturned = <clase  
'apps.reclamo.models.entidad_reclamo.EntidadReclamo.MultipleObjectsR  
eturned'>  
objetos = <objeto django.db.models.manager.Manager>
```

Métodos heredados de [django.db.models.base.Model](#) :

```
__eq__ (yo, otro)  
    Devuelve self == valor.  
__getstate__ (yo)  
    Gancho para permitir elegir los atributos a encurtir.  
__hash__ (yo)  
    Devuelve hash (self).  
__init__ (self, * argumentos, ** kwargs)  
    Inicializar uno mismo. Consulte la ayuda (escriba  
    (auto)) para obtener una firma precisa.  
__reduce__ (auto)  
    Ayudante para pepinillos.  
__repr__ (yo)  
    Devuelve repr (self).  
__setstate__ (self, state)  
__str__ (yo)  
    Devuelve str (self).
```

```
limpio (yo)  
    Hook para realizar cualquier validación adicional en  
    todo el modelo después de que se haya llamado  
    a clean ()  
    en cada campo por sí mismo. clean_fields . Cualquier  
    ValidationError generado  
    por este método no se asociará con un campo en  
    particular; tendrá  
    una asociación de caso especial con el campo definido  
    por NON_FIELD_ERRORS.
```

```
clean_fields (self, exclude = None)  
    Limpie todos los campos y genere un ValidationError  
    que contenga un dictado  
    de todos los errores de validación, si ocurre alguno.
```

```
date_error_message (self, lookup_type, field_name, unique_for)
```

```
eliminar (self, using = None, keep_parents = False)
```

```
full_clean (self, exclude = None, validate_unique = True)
```

```
    Llame a clean\_fields (), clean ()  
    y validate\_unique () en el modelo.  
    Genere un ValidationError para cualquier error que  
    ocurra.
```

```
get_deferred_fields (self)
```

```
    Devuelve un conjunto que contiene nombres de campos  
    diferidos para esta instancia.
```

```
prepare_database_save (self, field)
```

```
refresh_from_db (self, using = None, fields = None)
```

```
    Vuelva a cargar los valores de campo de la base de  
    datos.
```

```
    De forma predeterminada, la recarga ocurre desde la  
    base de datos desde la que se
```

cargó esta instancia , o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro using anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de att de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento 'crudo' le dice a save_base que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelva el valor del atributo del modelo .

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)

from_db (db, field_names,

values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

Datos

AUTORIZACION = ((0, 'SI'), (1, 'NO'))

COMPETENCIAS = ((1, 'SI'), (2, 'NO'), (3, 'COMPARTIDA'))

COMUNICACIONES_RESULTADO_RECLAMOS = ((1, 'DOMICILIO CONSIGNADO EN EL LIBRO DE RECLAMACIONES EN SALUD'), (2, 'CORREO ELECTRÓNICO'), (3, 'OTRA DIRECCIÓN'))

ESTADOS_RECLAMO = ((1, 'RESUELTO'), (2, 'EN TRAMITE '), (3, 'TRASLADADO '), (4, 'ARCHIVADO POR DUPLICIDAD '), (5, 'ACUMULADO '), (6, 'CONCLUIDO (CUANDO EL RESULTADO ES COMUNICADO AL RECLAMANTE O TERCERO LEGITIMADO) '))

ETAPAS_RECLAMO = ((1, 'ADMISIÓN Y REGISTRO'), (2, 'EVALUACIÓN E INVESTIGACIÓN'), (3, 'RESULTADO Y NOTIFICACIÓN'), (4, 'ARCHIVO Y CUSTODIA DEL EXPEDIENTE '))

MEDIOS = ((1, 'VIRTUAL '), (2, 'FÍSICO '))

MEDIOS_RECLAMO = ((1, 'LIBRO DE RECLAMACIONES VIRTUAL'), (2, 'LIBRO DE RECLAMACIONES FÍSICO'), (3, 'LLAMADA TELEFÓNICA'), (4, 'RECLAMO PRESENCIAL'), (5, 'DOCUMENTO ESCRITO'), (6, 'RECLAMO TRASLADO DE OTRA ADMINISTRADA'), (7, 'RECLAMO COPARTICIPADO CON OTRA ADMINISTRADA'))

MESES = ((1, 'ENERO'), (2, 'FEBRERO'), (3, 'MARZO'), (4, 'ABRIL'), (5, 'MAYO'), (6, 'JUNIO'), (7, 'JULIO'), (8, 'AGOSTO'), (9, 'SEPTIEMBRE'), (10, 'OCTUBRE'), (11, 'NOVIEMBRE'), (12, 'DICIEMBRE'))

MOTIVOS_CONCLUSION_ANTICIPADA = ((1, 'DESISTIMIENTO POR ESCRITO'), (2, 'TRATO DIRECTO'), (3, 'CONCILIACIÓN '), (4, 'TRANSACCIÓN EXTRAJUDICIAL '), (5, 'LAUDO ARBITRAL '))

RESULTADO_NOTIFICACION_RECLAMO = ((0, 'PENDIENTE'), (1, 'FUNDADO'), (2, 'FUNDADO PARCIAL'), (3, 'INFUNDADO'), (4, 'IMPROCEDENTE'), (5, 'CONCLUIDO ANTICIPADAMENTE'))

RESULTADO_RECLAMO = ((1, 'FUNDADO'), (2, 'EVALUACIÓN E INVESTIGACIÓN'), (3, 'RESULTADO Y NOTIFICACIÓN'), (4, 'ARCHIVO Y CUSTODIA DEL EXPEDIENTE'))

SERVICIOS = ((1, 'CONSULTA EXTERNA'), (2, 'HOSPITALIZACION'), (3, 'EMERGENCIA'), (4, 'CENTRO QUIRÚRGICO'), (5, 'CENTRO OBSTÉTRICO'), (6, 'UCI O UCIN'), (7, 'FARMACIA'), (8, 'SERVICIOS MÉDICOS DE APOYO'), (9, 'ATENCIÓN A DOMICILIO CONSULTA AMBULATORIA'), (10, 'ATENCIÓN A DOMICILIO URGENCIA A EMERGENCIA'), (11, 'OFICINAS O ÁREAS ADMINISTRATIVAS DE IAFAS A 1 PUNTA A UGIPRESS '), (12, 'INFRAESTRUCTURA '), (13, 'REFERENCIA Y CONTRAREFERENCIA '))

TIPOS_ADMINISTRADOR = ((1, 'IPRESS'), (2, 'UGIPRESS'), (3, 'IAFAS'))

TIPOS_DOCUMENTO = ((1, 'DNI'), (2, 'CARNÉ DE EXTRANJERÍA'), (3, 'PASAPORTE'), (4, 'DOCUMENTO DE IDENTIDAD EXTRANJERO'), (5, 'CÓDIGO ÚNICO DE IDENTIFICACIÓN (CUI)'), (6, 'CERTIFICADO DE NACIDO VIVO'), (7, 'PERMISO TEMPORAL DE PERMANENCIA '), (8, 'RUC'), (9, 'OTROS'))

4.8. Clases DetalleEvaluacionAnexo1

4.8.1. Clase DetalleEvaluacionAnexo1 ([django.db.models.base.Model](#))

[DetalleEvaluacionAnexo1](#) (* args, ** kwargs)

[DetalleEvaluacionAnexo1](#) (id, evaluacion, rubro, calificacion, observacion, created_at, updated_at, created_ip, updated_ip, created_by, updated_by)

Orden de resolución del método:

[DetalleEvaluaciónAnexo1](#)
[django.db.models.base.Model](#)
[objeto incorporado](#)

Métodos definidos aquí:

```
__str__ (self)
    Devuelve str (self).

calificacion = <objeto
django.db.models.query_utils.DeferredAttribute> created_at =
<objeto django.db.models.query_utils.DeferredAttribute > created_ip
= <objeto django.db.models.query_utils.DeferredAttribute>
get_next_by_created_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: created_at>, is_next = True, **
kwargs)
get_next_by_updated_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: updated_at>, is_next = True, **
kwargs)
get_previous_by_created_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: created_at>, is_next = False, **
kwargs)
get_previous_by_updated_at = _method (self, *, field =
<django.db.models.fields.DateTimeField: updated_at>, is_next = False, **
kwargs)
id = <objeto
django.db.models.query_utils.DeferredAttribute> observacion = <objeto
django.db.models.query_utils.DeferredAttribute>
guardar (uno mismo, * argumentos, ** kwargs)
    Guarde la instancia actual. Reemplace esto en una
    subclase si desea
    controlar el proceso de guardado.

    Los parámetros 'force_insert' y 'force_update' pueden
    usarse para insistir en
    que el "guardar" debe ser una inserción o
    actualización SQL (o equivalente para
    backends que no sean SQL),
    respectivamente. Normalmente, no deben configurarse.

updated_at = <objeto
django.db.models.query_utils.DeferredAttribute> updated_ip =
<django.db.models.query_utils.DeferredAttribute object>
```

Descriptores de datos definidos aquí:

creado por

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')
```

```
` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

created_by_id

evaluacion

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')
```

```
` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

evaluacion_id

rubro

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')
```

```
` Child.parent '' es una instancia de ``
ForwardManyToOneDescriptor ''.
```

rubro_id

Actualizado por

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')
```

```
`` Child.parent `` es una instancia de ``  
ForwardManyToOneDescriptor ``.
```

updated_by_id

Datos y otros atributos definidos aquí:

DoesNotExist = <clase

'apps.reclamo_administrador.models.evaluacion_anexo1.DetalleEvaluacionAnexo1.DoesNotExist'>

MultipleObjectsReturned = <clase

'apps.reclamo_administrador.models.evaluacion_anexo1.DetalleEvaluacionAnexo1.MultipleObjectsReturned'>

objetos = <objeto django.db.models.manager.Manager>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, otro)

Devuelve self == valor.

__getstate__ (self)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (yo)

Devuelve repr (self).

__setstate__ (self, state)

limpio (self)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a [clean](#) ()

en cada campo por sí mismo. **clean_fields** . Cualquier ValidationError generado

por este método no se asociará con un campo en particular; tendrá

una asociación de caso especial con el campo definido por NON_FIELD_ERRORS.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un ValidationError que contenga un dictado

de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)

eliminar (self, using = None, keep_parents = False)

full_clean (self, exclude = None, validate_unique = True)

Llame a [clean_fields](#) (), [clean](#) ()

y [validate_unique](#) () en el modelo.

Genere un ValidationError para cualquier error que ocurra.

get_deferred_fields (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)

refresh_from_db (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia, o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro `using` anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de `att` de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

`save_base` (`self`, `raw = False`, `force_insert = False`, `force_update = False`, `using = None`, `update_fields = None`)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento '`crudo`' le dice a `save_base` que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

`serializable_value` (`self`, `field_name`)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de `id` en lugar del objeto. Si no hay ningún objeto `Field` con este nombre en el modelo, devuelva el valor del atributo del modelo.

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

`mensaje_error_unico` (`self`, `clase_modelo`, `verificación_unica`)

`validate_unique` (`self`, `exclude = None`)

Verifique las restricciones únicas en el modelo y genere `ValidationError` si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

`comprobar` (`** kwargs`) de [django.db.models.base.ModelBase](#)
`from_db` (`db`, `field_names`,
`values`) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

4.8.2. **Clase EvaluacionAnexo1** ([django.db.models.base.Model](#))

[EvaluacionAnexo1](#) (* args, ** kwargs)

[EvaluacionAnexo1](#) (id, entidad, periodo, fecha, created_at, updated_at, created_ip, updated_ip, created_by, updated_by)

Orden de resolución del método:

[EvaluaciónAnexo1](#)

[django.db.models.base.Model](#)

[objeto incorporado](#)

Métodos definidos aquí:

__str__ (yo)

Devuelve str (self).

created_at = <objeto

django.db.models.query_utils.DeferredAttribute> **created_ip** = <objeto

django.db.models.query_utils.DeferredAttribute> **fecha** = <objeto

django.db.models.query_utils.DeferredAttribute>

get_next_by_created_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: created_at>, is_next = True, ** kwargs)

get_next_by_fecha = _method (self, *, field =

<django.db.models.fields.DateField: fecha>, is_next = True, ** kwargs)

get_next_by_updated_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: updated_at>, is_next = True, ** kwargs)

get_previous_by_created_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: created_at>, is_next = False, ** kwargs)

get_previous_by_fecha = _method (self, *, field =

<django.db.models.fields.DateField: fecha>, is_next = False, ** kwargs)

get_previous_by_updated_at = _method (self, *, field =

<django.db.models.fields.DateTimeField: updated_at>, is_next = False, ** kwargs)

id = <objeto django.db.models.query_utils.DeferredAttribute>

guardar (uno mismo, * argumentos, ** kwargs)

Guarde la instancia actual. Reemplace esto en una subclase si desea controlar el proceso de guardado.

Los parámetros 'force_insert' y 'force_update' pueden usarse para insistir en que el "guardar" debe ser una inserción o actualización SQL (o equivalente para

backends que no sean SQL),
respectivamente. Normalmente, no deben configurarse.

updated_at = <objeto

django.db.models.query_utils.DeferredAttribute> **updated_ip** =

<django.db.models.query_utils.DeferredAttribute object>

Descriptores de datos definidos aquí:

creado por

Acceso al objeto relacionado en el lado delantero de
una relación de muchos a uno o
uno a uno (a través de la subclase
ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):  
    parent = ForeignKey (Parent, related_name =  
    'children')
```

```
` Child.parent '' es una instancia de ``  
ForwardManyToOneDescriptor ''.
```

created_by_id

detalleevaluacionanexo1_set

Acceso al administrador de objetos relacionados en el
reverso de una relación de
muchos a uno.

En el ejemplo ::

```
class Child (Model):  
    parent = ForeignKey (Parent, related_name =  
    'children')
```

```
` Parent.children '' es una instancia de ``  
ReverseManyToOneDescriptor ''.
```

La mayor parte de la implementación se delega a una
clase de administrador definida
dinámicamente construida por ``
create_forward_many_to_many_manager () '' definido a
continuación.

entidad

Acceso al objeto relacionado en el lado delantero de
una relación de muchos a uno o
uno a uno (a través de la subclase
ForwardOneToOneDescriptor).

En el ejemplo ::

```
class Child (Model):  
    parent = ForeignKey (Parent, related_name =  
    'children')
```

```
` Child.parent '' es una instancia de ``  
ForwardManyToOneDescriptor ''.
```

entidad_id

periodo

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase `ForwardOneToOneDescriptor`).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent `` es una instancia de ``
ForwardManyToOneDescriptor ``.
```

periodo_id

Actualizado por

Acceso al objeto relacionado en el lado delantero de una relación de muchos a uno o uno a uno (a través de la subclase `ForwardOneToOneDescriptor`).

En el ejemplo ::

```
class Child (Model):
    parent = ForeignKey (Parent, related_name =
'children')

`` Child.parent `` es una instancia de ``
ForwardManyToOneDescriptor ``.
```

updated_by_id

Datos y otros atributos definidos aquí:

DoesNotExist = <clase

'apps.reclamo_administrador.models.evaluacion_anexo1.EvaluacionAnexo1.DoesNotExist'>

MultipleObjectsReturned = <clase

'apps.reclamo_administrador.models.evaluacion_anexo1.EvaluacionAnexo1.MultipleObjectsReturned'>

objetos = <objeto django.db.models.manager.Manager>

Métodos heredados de [django.db.models.base.Model](#) :

__eq__ (self, other)

Devuelve self == valor.

__getstate__ (self)

Gancho para permitir elegir los atributos a encurtir.

__hash__ (self)

Devuelve hash (self).

__init__ (self, * argumentos, ** kwargs)

Inicializar uno mismo. Consulte la ayuda (escriba (auto)) para obtener una firma precisa.

__reduce__ (auto)

Ayudante para pepinillos.

__repr__ (self)

Devuelve repr (self).

__setstate__ (self, state)

limpio (self)

Hook para realizar cualquier validación adicional en todo el modelo después de que se haya llamado a `clean()` en cada campo por sí mismo. **clean_fields**. Cualquier `ValidationError` generado por este método no se asociará con un campo en particular; tendrá una asociación de caso especial con el campo definido por `NON_FIELD_ERRORS`.

clean_fields (self, exclude = None)

Limpie todos los campos y genere un `ValidationError` que contenga un dictado de todos los errores de validación, si ocurre alguno.

date_error_message (self, lookup_type, field_name, unique_for)**eliminar** (self, using = None, keep_parents = False)**full_clean** (self, exclude = None, validate_unique = True)

Llame a `clean_fields()`, `clean()` y `validate_unique()` en el modelo. Genere un `ValidationError` para cualquier error que ocurra.

get_deferred_fields (uno mismo)

Devuelve un conjunto que contiene nombres de campos diferidos para esta instancia.

prepare_database_save (self, field)**refresh_from_db** (self, using = None, fields = None)

Vuelva a cargar los valores de campo de la base de datos.

De forma predeterminada, la recarga ocurre desde la base de datos desde la que se cargó esta instancia, o por el enrutador de lectura si esta instancia no se cargó desde ninguna base de datos. El parámetro `using` anulará el predeterminado.

Los campos se pueden utilizar para especificar qué campos volver a cargar. Los campos deben ser iterables de nombres de `att` de campo. Si los campos son Ninguno, se recargan todos los campos no diferidos.

Al acceder a los campos diferidos de una instancia, la carga diferida del campo llamará a este método.

save_base (self, raw = False, force_insert = False, force_update = False, using = None, update_fields = None)

Maneje las partes del guardado que deben hacerse solo una vez por guardado, pero también deben hacerse en guardados sin formato. Esto incluye algunas comprobaciones de cordura y envío de señales.

El argumento `'crudo'` le dice a `save_base` que no guarde ningún modelo padre y que no haga ningún cambio en los valores antes de guardar. Esto es utilizado por la carga de dispositivos.

serializable_value (self, field_name)

Devuelve el valor del nombre del campo para esta instancia. Si el campo es una clave externa, devuelve el valor de id en lugar del objeto. Si no hay ningún objeto Field con este nombre en el modelo, devuelve el valor del atributo del modelo.

Se utiliza para serializar el valor de un campo (en el serializador o en la salida del formulario, por ejemplo). Normalmente, solo accedería al atributo directamente y no usaría este método.

mensaje_error_unico (self, clase_modelo, verificación_unica)

validate_unique (self, exclude = None)

Verifique las restricciones únicas en el modelo y genere ValidationError si alguna falla.

Métodos de clase heredados de [django.db.models.base.Model](#) :

comprobar (** kwargs) de [django.db.models.base.ModelBase](#)

from_db (db, field_names,

values) de [django.db.models.base.ModelBase](#)

Descriptores de datos heredados de [django.db.models.base.Model](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

paquete

5. CAPA DE LA VISTA

5.1. Clases EntidadReclamo

5.1.1. Clase EntidadReclamoCreate ([django.views.generic.edit.CreateView](#))

[EntidadReclamoCreate](#) (** kwargs)

Vista para crear un nuevo objeto, con una respuesta renderizada por una plantilla.

Orden de resolución del método:

[EntidadReclamoCrear](#)

[django.views.generic.edit.CreateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseCreateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)
[django.views.generic.base.ContextMixin](#)
[django.views.generic.edit.ProcessFormView](#)
[django.views.generic.base.View](#)
[objeto incorporado](#)

Métodos definidos aquí:

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

get_context_data (self, ** kwargs)

Inserte el formulario en el diccionario de contexto.

get_form_kwargs (self, * argumentos, ** kwargs)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

Datos y otros atributos definidos aquí:

form_class = <clase

'apps.reclamo.forms.entidad_reclamo.EntidadReclamoForm'>

model = <clase 'apps.reclamo.models.entidad_reclamo.EntidadReclamo'>

EntidadReclamo (id, entidad, periodo_declaracion,
tipo_institucion, codigo_administrado,
medio_presentacion, codigo_registro,
tipo_documento_usuario, numero_documento_usuario,
razon_social_usuario, nombres_usuario,
apellido_paterno_usuario, apellido_materno_usuario,
tipo_documento_presenta, numero_documento_presenta,
razon_social_presenta, nombres_presenta,
apellido_paterno_presenta, apellido_materno_presenta,
autorizacion_notificacion_correo, correo_presenta,
domicilio_presenta, celular_presenta,
medio_recepcion_reclamo, fecha_reclamo,
detalle_reclamo, servicio_hecho_reclamo,
competencia_reclamo, clasificacion_reclamo_1,
clasificacion_reclamo_2, clasificacion_reclamo_3,
estado_reclamo, codigo_reclamo_primigenio,
etapa_reclamo, tipo_administrado_traslado,
codigo_administrado_deriva, resultado_reclamo,
motivo_conicipadofecha_resultado_reclamo,
comunicacion_resultado_reclamo, fecha_notificacion,
es_mismo_usuario_afectado, created_at, updated_at,
created_ip, updated_ip)

Success_url = '/ reclamo / entidad-reclamo / list'

Datos y otros atributos heredados

de [django.views.generic.edit.CreateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede

llamar si se [anula render to response](#) (). Devuelve la siguiente lista:

- * el valor de `` template_name '' en la vista (si se proporciona)
- * el contenido del campo `` template_name_field '' en la instancia del objeto sobre el que opera la vista (si está disponible)

```
* `` < app_label> / <model_name>  
<template_name_suffix> .html ''
```

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :
template_name_field = Ninguno

Métodos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
render_to_response (yo, contexto, ** respuesta_kwargs)
Devuelve una respuesta, usando el `response_class` para esta vista, con una plantilla renderizada con el contexto dado.

Pase response_kwargs al constructor de la clase de respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
__dict__
diccionario, por ejemplo, variables (si están definidas)
__weakref__
lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
content_type = Ninguno
response_class = <clase 'django.template.response.TemplateResponse'>
template_engine = Ninguno
template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseCreateView](#) :
get (self, request, * args, ** kwargs)
Manejar solicitudes GET: instancia una versión en blanco del formulario.
publicar (self, request, * args, ** kwargs)
Maneje las solicitudes POST: cree una instancia de formulario con las variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :
get_form_class (self)
Devuelve la clase de formulario para usar en esta vista.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados

de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados

de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir `self. queryset` y un argumento `pk` o `slug` en la URLconf.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (self)

Devuelve el `QuerySet` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de `get_object ()` y no puede ser llamado si `get_object ()` está anulado.

get_slug_field (self)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados

de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

despacho (self, request, * args, ** kwargs)

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

5.1.2. Clase EntidadReclamoList ([apps.util.generic_filters.views.FilteredListView](#))

[EntidadReclamoList](#) (** kwargs)

Un ListView genérico utilizado para filtrar y ordenar objetos.

Orden de resolución del método:

[EntidadReclamoList](#)

[apps.util.generic_filters.views.FilteredListView](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.list.ListView](#)

[django.views.generic.list.MultipleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.list.BaseListView](#)

[django.views.generic.list.MultipleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.base.View](#)

[objeto incorporado](#)

Métodos definidos aquí:

get_context_data (self ** kwargs)

Agregue una lista de filtros y uno mismo.
Formulario al contexto que será representado por la vista.

get_queryset (self)

Devuelve el conjunto de consultas filtrado. Utiliza [form_valid \(\)](#) o [form_invalid \(\)](#).

Datos y otros atributos definidos aquí:

default_order = '-id'

filter_fields = ['clasificacion_reclamo_1_categoria']

form_class = <clase

'apps.reclamo.forms.entidad_reclamo.EntidadReclamoListFilter'>

model = <clase 'apps.reclamo.models.entidad_reclamo.EntidadReclamo'>

EntidadReclamo (id, entidad, periodo_declaracion,
tipo_institucion, codigo_administrado,
medio_presentacion, codigo_registro,
tipo_documento_usuario, numero_documento_usuario,
razon_social_usuario, nombres_usuario,
apellido_paterno_usuario, apellido_materno_usuario,
tipo_documento_presenta, numero_documento_presenta,
razon_social_presenta, nombres_presenta,
apellido_paterno_presenta, apellido_materno_presenta,
autorizacion_notificacion_correo, correo_presenta,
domicilio_presenta, celular_presenta,
medio_recepcion_reclamo, fecha_reclamo,
detalle_reclamo, servicio_hecho_reclamo,
competencia_reclamo, clasificacion_reclamo_1,
clasificacion_reclamo_2, clasificacion_reclamo_3,
estado_reclamo, codigo_reclamo_primigenio,
etapa_reclamo, tipo_administrado_traslado,
codigo_administrado_deriva, resultado_reclamo,
motivo_conicipadofecha_resultado_reclamo,
comunicacion_resultado_reclamo, fecha_notificacion,
es_mismo_usuario_afectado, created_at, updated_at,
created_ip, updated_ip)

paginate_by = 30

search_fields = ['codigo_administrado', 'razon_social_usuario',

'nombres_usuario', 'apellido_paterno_usuario', 'apellido_materno_usuario']

Métodos heredados de [apps.util.generic_filters.views.FilteredListView](#) :

clean_qs_filter_field (self, key, value)

form_empty (yo)

Devuelve el conjunto de consultas utilizado cuando no se envía el formulario.

form_invalid (self, form)

Devuelve el conjunto de consultas cuando el formulario enviado no es válido.

La implementación predeterminada utiliza: meth:
'form_empty'.

: param: `django.forms.Forms`
: return: `django.db.models.query.QuerySet`

form_valid (self, form)

Form_valid es responsable de filtrar y ordenar el conjunto de consultas base. Devuelve un conjunto de consultas.

```
: param: `django.forms.Forms`  
: return: `django.db.models.query.QuerySet`
```

get_filters (uno mismo)

Convierta algunos ChoiceField en una lista de opciones en la plantilla.

get_form_kwargs (self)

Lea los datos GET para devolver argumentos de palabras clave para el formulario.

get_initial (yo)

agregue "order_by" y "order_reverse" a las iniciales.

get_qs_filters (self)

recupere los filtros de "qs_filter_fields" o "filter_fields" y devuélvalos como un dictado para ser utilizado por uno mismo. **form_valid**

is_form_submitted (auto)

Devuelve True si el formulario ya está enviado. Falso de lo contrario

Descriptores de datos heredados

de [apps.util.generic_filters.views.FilteredListView](#) :

formulario

adivine el formulario que se utilizará y oculte los campos filter_field en la plantilla

Datos y otros atributos heredados

de [apps.util.generic_filters.views.FilteredListView](#) :

default_filter = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :**get_form (self, form_class = Ninguno)**

Devuelve una instancia del formulario que se utilizará en esta vista.

get_form_class (uno mismo)

Devuelve la clase de formulario para usar.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

get_success_url (yo)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados

de [django.views.generic.edit.FormMixin](#) :

inicial = { }

prefijo = Ninguno

success_url = Ninguno

Métodos heredados

de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. Debe devolver

una lista. No se puede llamar si se anula
`render_to_response`.

Datos y otros atributos heredados
de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :
template_name_suffix = '_list'

Métodos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
render_to_response (yo, contexto, ** respuesta_kwargs)
Devuelve una respuesta, usando el `response_class`
para esta vista, con una
plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de
respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
__dict__
diccionario, por ejemplo, variables (si están
definidas)
__weakref__
lista de referencias débiles al objeto (si está
definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :
content_type = Ninguno
response_class = <clase 'django.template.response.TemplateResponse'>
template_engine = Ninguno
template_name = Ninguno

Métodos heredados de [django.views.generic.list.BaseListView](#) :
get (self, request, * args, ** kwargs)

Métodos heredados de [django.views.generic.list.MultipleObjectMixin](#) :
get_allow_empty (auto)
Devuelve `True` si la vista debe mostrar
listas vacías y `False`
si se debe generar un 404 en su lugar.
get_context_object_name (self, object_list)
Obtenga el nombre del elemento que se utilizará en el
contexto.
get_ordering (auto)
Devuelve el campo o los campos que se utilizarán para
ordenar el conjunto de consultas.
get_paginate_by (self, queryset)
Obtenga la cantidad de elementos para paginar o
Ninguno para no paginar.
get_paginate_orphans (self)
Devuelve el número máximo de huérfanos, extiende la
última página al
paginar.

get_paginador (self, queryset, per_page, huérfanos = 0,
allow_empty_first_page = True, ** kwargs)
Devuelve una instancia del paginador para esta vista.

paginate_queryset (self, queryset, page_size)
Pagina el conjunto de consultas, si es necesario.

Datos y otros atributos heredados
de [django.views.generic.list.MultipleObjectMixin](#) :
allow_empty = Verdadero
context_object_name = Ninguno
ordering = Ninguno
page_kwarg = 'página'
paginate_orphans = 0
paginator_class = <clase 'django.core.paginator.Paginator'>
queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :
extra_context = Ninguno

Métodos heredados de [django.views.generic.base.View](#) :
__init__ (self, ** kwargs)
Constructor. Llamado en la URLconf; puede contener
argumentos de palabras clave adicionales útiles y
otras cosas.

despacho (self, request, * args, ** kwargs)
http_method_not_allowed (self, request, * args, ** kwargs)
opciones (self, request, * args, ** kwargs)
Manejar la respuesta a las solicitudes del verbo
OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)
Inicialice los atributos compartidos por todos los
métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :
as_view (** initkwargs) de [builtins.type](#)
Punto de entrada principal para un proceso de
solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :
http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options',
'trace']

5.1.3. Clase EntidadReclamoUpdate ([django.views.generic.edi t.UpdateView](#))

[EntidadReclamoUpdate](#) (** kwargs)

Vista para actualizar un objeto, con una respuesta
renderizada por una plantilla.

Orden de resolución del método:

[EntidadReclamoUpdate](#)
[django.views.generic.edit.UpdateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)
[django.views.generic.base.TemplateResponseMixin](#)
[django.views.generic.edit.BaseUpdateView](#)
[django.views.generic.edit.ModelFormMixin](#)
[django.views.generic.edit.FormMixin](#)
[django.views.generic.detail.SingleObjectMixin](#)
[django.views.generic.base.ContextMixin](#)
[django.views.generic.edit.ProcessFormView](#)
[django.views.generic.base.View](#)
[objeto incorporado](#)

Métodos definidos aquí:

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

get_context_data (self, ** kwargs)

Inserte el formulario en el diccionario de contexto.

get_form_kwargs (self, * argumentos, ** kwargs)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

Datos y otros atributos definidos aquí:

form_class = <clase

'apps.reclamo.forms.entidad_reclamo.EntidadReclamoForm'>

model = <clase 'apps.reclamo.models.entidad_reclamo.EntidadReclamo'>

EntidadReclamo (id, entidad, periodo_declaracion,
tipo_institucion, codigo_administrado,
medio_presentacion, codigo_registro,
tipo_documento_usuario, numero_documento_usuario,
razon_social_usuario, nombres_usuario,
apellido_paterno_usuario, apellido_materno_usuario,
tipo_documento_presenta, numero_documento_presenta,
razon_social_presenta, nombres_presenta,
apellido_paterno_presenta, apellido_materno_presenta,
autorizacion_notificacion_correo, correo_presenta,
domicilio_presenta, celular_presenta,
medio_recepcion_reclamo, fecha_reclamo,
detalle_reclamo, servicio_hecho_reclamo,
competencia_reclamo, clasificacion_reclamo_1,
clasificacion_reclamo_2, clasificacion_reclamo_3,
estado_reclamo, codigo_reclamo_primigenio,
etapa_reclamo, tipo_administrado_traslado,
codigo_administrado_deriva, resultado_reclamo,
motivo_conicipadofecha_resultado_reclamo,
comunicacion_resultado_reclamo, fecha_notificacion,
es_mismo_usuario_afectado, created_at, updated_at,
created_ip, updated_ip)

Success_url = '/ reclamo / entidad-reclamo / list'

Datos y otros atributos heredados

de [django.views.generic.edit.UpdateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render to response](#) (). Devuelve la siguiente lista:

```
* el valor de `` template_name '' en la vista (si se proporciona)
* el contenido del campo `` template_name_field '' en la
  instancia del objeto sobre el que opera la vista (si está disponible)
* `` < app_label> / <model_name>
  <template_name_suffix> .html ''
```

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el `response_class` para esta vista, con una plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de respuesta.

Descriptores de datos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseUpdateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de formulario con las variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (self)

Devuelve la clase de formulario para usar en esta vista.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados

de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados

de [django.views.generic.edit.ModelFormMixin](#) :

inicial = { }

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir `self. queryset` y un argumento `pk` o `slug` en la URLconf.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (self)

Devuelve el `QuerySet` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de [get_object \(\)](#) y no puede ser llamado si [get_object \(\)](#) está anulado.

get_slug_field (self)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso
queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :
extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :
poner (self, * argumentos, ** kwargs)
PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :
__init__ (self, ** kwargs)
Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.
despacho (self, request, * args, ** kwargs)
http_method_not_allowed (self, request, * args, ** kwargs)
opciones (self, request, * args, ** kwargs)
Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.
configuración (self, request, * args, ** kwargs)
Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :
as_view (** initkwargs) de [builtins.type](#)
Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :
http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

Datos

input_hidden = ['created_at', 'updated_at', 'created_ip', 'updated_ip']
paso_cinco = ['servicio_hecho_reclamo', 'competencia_reclamo', 'clasificacion_reclamo_1', 'clasificacion_reclamo_2', 'clasificacion_reclamo_3', 'estado_reclamo', 'codigen_reclamo', 'codigen_reclamo', 'etapa_reclamo', 'tipo_administrado_traslado', 'codigo_administrado_deriva']
paso_cuatro = ['medio_recepcion_reclamo', 'fecha_reclamo', 'detalle_reclamo']
paso_dos = ['tipo_documento_usuario', 'numero_documento_usuario', 'razon_social_usuario', 'apellido_materno_usuario']
paso_seis = ['resultado_reclamo', 'motivo_conclusion_anticipada', 'fecha_resultado_reclamo', 'comunicacion_resultado_reclamo', 'fecha_notificacion']
paso_tres = ['tipo_documento_presenta', 'numero_documento_presenta',

'razon_social_presenta', 'nombres_presenta', 'apellido_paterno_presenta',
'apellido_materno_presenta', 'autorizacion_notificacion_correo',
'correo_presenta', 'domicilio_presenta', 'celular_presenta']
paso_uno = ['periodo_declaracion', 'tipo_administrador',
'codigo_administrador', 'codigo_administrador', 'codigo_ugipress',
'tipo_institucion', 'codigo_administrado', 'codigo_administrado',
'medio_presentacion', 'codigo_registro']

5.2. Clases User

5.2.1. Clase **UserPasswordUpdate** ([django.views.generic.edit.UpdateView](#))

[UserPasswordUpdate](#) (** kwargs)

Orden de resolución del método:

[UserPasswordUpdate](#)

[django.views.generic.edit.UpdateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseUpdateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.edit.ProcessFormView](#)

[django.views.generic.base.View](#)

[objeto incorporado](#)

Métodos definidos aquí:

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

get_context_data (uno mismo, ** kwargs)

Inserte el formulario en el diccionario de contexto.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos definidos aquí:

form_class = <clase 'setup.forms.usuario.UsuarioChangePasswordForm'>

Un formulario que crea un usuario, sin privilegios, a partir del nombre de usuario y la contraseña dados .

modelo = <clase 'setup.models.usuario.Usuario'>

Usuario (id, contraseña, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, entidad, document, celular)

template_name = 'setup / change_password.html'

Datos y otros atributos heredados de [django.views.generic.edit.UpdateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (uno mismo)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render_to_response](#) (). Devuelve la siguiente lista:

- * el valor de `` template_name '' en la vista (si se proporciona)
- * el contenido del campo `` template_name_field '' en la instancia del objeto sobre el que opera la vista (si está disponible)
- * `` < app_label> / <model_name> <template_name_suffix> .html ''

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el ``response_class`` para esta vista, con una plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de respuesta.

Descriptores de datos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

Métodos heredados de [django.views.generic.edit.BaseUpdateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de formulario con las variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (uno mismo)

Devuelve la clase de formulario para usar en esta vista.

get_form_kwargs (uno mismo)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

Datos y otros atributos heredados de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

success_url = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir `self. queryset` y un argumento `pk` o `slug` en la URLconf.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (self)

Devuelve el `QuerySet` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de [get_object \(\)](#) y no puede ser llamado si [get_object \(\)](#) está anulado.

get_slug_field (self)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

despacho (self, request, * args, ** kwargs)

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo
OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los
métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de
solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options',
'trace']

5.2.2. **Clase UsuarioCreate** ([django.views.generic.edit.CreateView](#))

[UsuarioCreate](#) (** kwargs)

Orden de resolución del método:

[UsuarioCreate](#)

[django.views.generic.edit.CreateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseCreateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.edit.ProcessFormView](#)

[django.views.generic.base.View](#)

objeto incorporado

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

get_context_data (uno mismo, ** kwargs)

Inserte el formulario en el diccionario de contexto.

Datos y otros atributos definidos aquí:

form_class = <clase 'setup.forms.usuario.UsuarioForm'>

modelo = <clase 'setup.models.usuario.Usuario'>

Usuario (id, contraseña, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, entidad, document, celular)

Success_url = '/ setup / usuario / list'

Datos y otros atributos heredados de [django.views.generic.edit.CreateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render_to_response](#) (). Devuelve la siguiente lista:

- * el valor de `` template_name `` en la vista (si se proporciona)
- * el contenido del campo `` template_name_field `` en la instancia del objeto sobre el que opera la vista (si está disponible)
- * `` < app_label > / < model_name > < template_name_suffix > .html ``

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el `response_class` para esta vista, con una plantilla renderizada con el contexto dado.

Pase response_kwargs al constructor de la clase de respuesta.

Descriptores de datos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseCreateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de formulario con las

variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (self)

Devuelve la clase de formulario para usar en esta vista.

get_form_kwargs (self)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

get_success_url (yo)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados
de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir ``self. queryset`` y un argumento ``pk`` o ``slug`` en la URLconf.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (yo)

Devuelve el ``QuerySet`` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de [get_object](#) () y no puede ser llamado si [get_object](#) () está anulado.

get_slug_field (yo)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

5.2.3. **Clase UsuarioList** ([apps.util.generic_filters.views.FilteredListView](#))

[UsuarioList](#) (** kwargs)

Orden de resolución del método:

[UsuarioList](#)

[apps.util.generic_filters.views.FilteredListView](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.list.ListView](#)

[django.views.generic.list.MultipleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.list.BaseListView](#)[django.views.generic.list.MultipleObjectMixin](#)[django.views.generic.base.ContextMixin](#)[django.views.generic.base.View](#)[objeto incorporado](#)

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

get_context_data (uno mismo, ** kwargs)

Agregue una lista de filtros y uno mismo. **formulario** al contexto que será representado por la vista.

get_queryset (self)

Devuelve el conjunto de consultas filtrado. Utiliza [form valid](#) () o [form invalid](#) () .

Datos y otros atributos definidos aquí:

default_order = 'id'

form_class = <clase 'setup.forms.usuario.UsuarioListFilter'>

modelo = <clase 'setup.models.usuario.Usuario'>

Usuario (id, contraseña, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, entidad, document, celular)

paginate_by = 30

search_fields = ['username', 'first_name', 'last_name', 'document']

Métodos heredados de [apps.util.generic_filters.views.FilteredListView](#) :

clean_qs_filter_field (self, key, value)

form_empty (yo)

Devuelve el conjunto de consultas utilizado cuando no se envía el formulario.

form_invalid (self, form)

Devuelve el conjunto de consultas cuando el formulario enviado no es válido.

La implementación predeterminada utiliza: `meth: `form_empty``.

```
: param: `django.forms.Forms`  
: return: ` django.db.models.query.QuerySet`
```

form_valid (self, form)

`Form_valid` es responsable de filtrar y ordenar el conjunto de consultas base. Devuelve un conjunto de consultas.

```
: param: `django.forms.Forms`  
: return: ` django.db.models.query.QuerySet`
```

get_filters (self)

Convierta algunos `ChoiceField` en una lista de opciones en la plantilla.

get_form_kwargs (self)

Lea los datos GET para devolver argumentos de palabras clave para el formulario.

get_initial (self)

agregue `"order_by"` y `"order_reverse"` a las iniciales.

get_qs_filters (self)

recupere los filtros de `"qs_filter_fields"` o `"filter_fields"` y devuélvalos como un dictado para ser utilizado por uno mismo. **form_valid**

is_form_submitted (auto)

Devuelve `True` si el formulario ya está enviado. Falso de lo contrario

Propiedades de solo lectura heredadas
de [apps.util.generic_filters.views.FilteredListView](#) :

formulario

adivine el formulario que se utilizará y oculte los campos `filter_field` en la plantilla

Datos y otros atributos heredados
de [apps.util.generic_filters.views.FilteredListView](#) :

default_filter = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_form_class (uno mismo)

Devuelve la clase de formulario para usar.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

success_url = Ninguno

Métodos heredados

de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :

get_template_names (uno mismo)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. Debe devolver una lista. No se puede llamar si se anula `render_to_response`.

Datos y otros atributos heredados

de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :

template_name_suffix = '_list'

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el ``response_class`` para esta vista, con una plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

`__dict__`

diccionario, por ejemplo, variables (si están definidas)

`__weakref__`

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

`content_type` = Ninguno

`response_class` = <clase 'django.template.response.TemplateResponse'>

`template_engine` = Ninguno

`template_name` = Ninguno

Métodos heredados de [django.views.generic.list.BaseListView](#) :

`get` (self, request, * args, ** kwargs)

Métodos heredados de [django.views.generic.list.MultipleObjectMixin](#) :

`get_allow_empty` (auto)

Devuelve `` Verdadero `` si la vista debe mostrar listas vacías y `` Falso `` si se debe generar un 404 en su lugar.

`get_context_object_name` (self, object_list)

Obtenga el nombre del elemento que se utilizará en el contexto.

`get_ordering` (auto)

Devuelve el campo o los campos que se utilizarán para ordenar el conjunto de consultas.

`get_paginate_by` (self, queryset)

Obtenga la cantidad de elementos para paginar o ``
Ninguno `` para no paginar.

get_paginate_orphans (yo)

Devuelve el número máximo de huérfanos, extiende la
última página al
paginar.

**get_paginator (self, queryset, per_page, huérfanos = 0,
allow_empty_first_page = True, ** kwargs)**

Devuelve una instancia del paginador para esta vista.

paginate_queryset (self, queryset, page_size)

Pagina el conjunto de consultas, si es necesario.

Datos y otros atributos heredados
de [django.views.generic.list.MultipleObjectMixin](#) :

allow_empty = Verdadero

context_object_name = Ninguno

ordering = Ninguno

page_kwarg = 'página'

paginate_orphans = 0

paginator_class = <clase 'django.core.paginator.Paginator'>

queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (yo, ** kwargs)

Constructor. Llamado en la URLconf; puede contener
argumentos de palabras clave adicionales útiles y
otras cosas.

http_method_not_allowed (self, request, * args, ** kwargs)**opciones (self, request, * args, ** kwargs)**

Manejar la respuesta a las solicitudes del verbo
OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

5.2.4. Clase UsuarioUpdate ([django.views.generic.edit.UpdateView](#))

[UsuarioUpdate](#) (** kwargs)

Orden de resolución del método:

[UsuarioUpdate](#)

[django.views.generic.edit.UpdateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseUpdateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.edit.ProcessFormView](#)

[django.views.generic.base.View](#)

[objeto incorporado](#)

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

get_context_data (uno mismo, ** kwargs)

Inserte el formulario en el diccionario de contexto.

Datos y otros atributos definidos aquí:

form_class = <clase 'setup.forms.usuario.UsuarioChangeForm'>

modelo = <clase 'setup.models.usuario.Usuario'>

Usuario (id, contraseña, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, entidad, document, celular)

Success_url = '/ setup / usuario / list'

Datos y otros atributos heredados de [django.views.generic.edit.UpdateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (uno mismo)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render to response](#) (). Devuelve la siguiente lista:

- * el valor de `` template_name '' en la vista (si se proporciona)
- * el contenido del campo `` template_name_field '' en la instancia del objeto sobre el que opera la vista (si está disponible)
- * `` < app_label> / <model_name> <template_name_suffix> .html ''

Datos y otros atributos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el ``response_class`` para esta vista, con una plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de respuesta.

Descriptores de datos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados

de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseUpdateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de formulario con las variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (self)

Devuelve la clase de formulario para usar en esta vista.

get_form_kwargs (self)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

get_success_url (yo)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados
de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (yo)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir ``self. queryset`` y un argumento ``pk`` o ``slug`` en la `URLconf`.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (self)

Devuelve el ``QuerySet`` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de `get_object ()` y no puede ser llamado si `get_object ()` está anulado.

get_slug_field (yo)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados
de [django.view.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

5.3. Clases Entidad

5.3.1. Clase EntidadCreate ([django.views.generic.edit.CreateView](#))

[EntidadCreate](#) (** kwargs)

Orden de resolución del método:

[EntidadCreate](#)

[django.views.generic.edit.CreateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseCreateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.edit.ProcessFormView](#)

[django.views.generic.base.View](#)

[objeto incorporado](#)

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

Datos y otros atributos definidos aquí:

form_class = <clase 'setup.forms.entidad.EntidadForm'>

modelo = <clase 'setup.models.entidad.Entidad'>

Entidad (id, nombre, codigo, categoria, ris, tipo)

Success_url = '/ setup / entidad / list'

Datos y otros atributos heredados de [django.views.generic.edit.CreateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render to response](#) (). Devuelve la siguiente lista:

- * el valor de `` template_name '' en la vista (si se proporciona)
- * el contenido del campo `` template_name_field '' en la instancia del objeto sobre el que opera la vista (si está disponible)
- * `` < app_label > / < model_name > < template_name_suffix > .html ''

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el `response_class`
para esta vista, con una
plantilla renderizada con el contexto dado.

Pase response_kwargs al constructor de la clase de
respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están
definidas)

__weakref__

lista de referencias débiles al objeto (si está
definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseCreateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en
blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de formulario con las variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (self)

Devuelve la clase de formulario para usar en esta vista.

get_form_kwargs (self)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados
de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_context_data (self, ** kwargs)

Inserte el formulario en el diccionario de contexto.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir `self. **queryset** `y un argumento` pk` o
`slug` en la URLconf.

Las subclases pueden anular esto para devolver
cualquier objeto.

get_queryset (self)

Devuelve el `QuerySet` que se utilizará para buscar
el objeto.

Este método es llamado por la implementación
predeterminada de [get_object](#) () y
no puede ser llamado si [get_object](#) () está anulado.

get_slug_field (self)

Obtenga el nombre de un campo slug que se utilizará
para buscar por slug.

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

```
# PUT es un verbo HTTP válido para crear (con una
URL conocida) o editar un
# objeto, tenga en cuenta que los navegadores solo
admiten POST por ahora.
```

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

```
Constructor. Llamado en la URLconf; puede contener
argumentos de palabras clave adicionales útiles y
otras cosas.
```

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

```
Manejar la respuesta a las solicitudes del verbo
OPTIONS HTTP.
```

configuración (self, request, * args, ** kwargs)

```
Inicialice los atributos compartidos por todos los
métodos de vista.
```

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

```
Punto de entrada principal para un proceso de
solicitud-respuesta.
```

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

```
http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options',
'trace']
```

5.3.2. Clase EntidadList ([apps.util.generic_filters.views.FilteredListView](#))

[EntidadList](#) (** kwargs)

Orden de resolución del método:

[EntidadList](#)

[apps.util.generic_filters.views.FilteredListView](#)

[django.views.generic.edit.FormMixin](#)[django.views.generic.list.ListView](#)[django.views.generic.list.MultipleObjectTemplateResponseMixin](#)[django.views.generic.base.TemplateResponseMixin](#)[django.views.generic.list.BaseListView](#)[django.views.generic.list.MultipleObjectMixin](#)[django.views.generic.base.ContextMixin](#)[django.views.generic.base.View](#)[objeto incorporado](#)

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

get_context_data (self, ** kwargs)

Agregue una lista de filtros y uno mismo. **formulario** al contexto que será representado por la vista.

Datos y otros atributos definidos aquí:

default_order = '-id'

filter_fields = []

form_class = <clase 'setup.forms.entidad.EntidadListFilter'>

modelo = <clase 'setup.models.entidad.Entidad'>

Entidad (id, nombre, codigo, categoria, ris, tipo)

paginate_by = 30

search_fields = ['nombre']

Métodos heredados de [apps.util.generic_filters.views.FilteredListView](#) :

clean_qs_filter_field (self, key, value)

form_empty (self)

Devuelve el conjunto de consultas utilizado cuando no se envía el formulario.

form_invalid (self, form)

Devuelve el conjunto de consultas cuando el formulario enviado no es válido.

La implementación predeterminada utiliza: `meth: `form_empty``.

```
: param: `django.forms.Forms`  
: return: ` django.db.models.query.QuerySet`
```

form_valid (self, form)

`Form_valid` es responsable de filtrar y ordenar el conjunto de consultas base. Devuelve un conjunto de consultas.

```
: param: `django.forms.Forms`  
: return: ` django.db.models.query.QuerySet`
```

get_filters (self)

Convierta algunos `ChoiceField` en una lista de opciones en la plantilla.

get_form_kwargs (self)

Lea los datos GET para devolver argumentos de palabras clave para el formulario.

get_initial (self)

agregue `"order_by"` y `"order_reverse"` a las iniciales.

get_qs_filters (uno mismo)

recupere los filtros de `"qs_filter_fields"` o `"filter_fields"` y devuélvalos como un dictado para ser utilizado por uno mismo. **form_valid**

get_queryset (self)

Devuelve el conjunto de consultas filtrado. Utiliza [form_valid \(\)](#) o [form_invalid \(\)](#).

is_form_submitted (auto)

Devuelve `True` si el formulario ya está enviado. Falso de lo contrario

Propiedades de solo lectura heredadas
de [apps.util.generic_filters.views.FilteredListView](#) :

formulario

adivine el formulario que se utilizará y oculte los campos `filter_field` en la plantilla

Datos y otros atributos heredados
de [apps.util.generic.filters.views.FilteredListView](#) :

default_filter = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_form_class (self)

Devuelve la clase de formulario para usar.

get_prefix (self)

Devuelve el prefijo para usar en formularios.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

success_url = Ninguno

Métodos heredados
de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :

get_template_names (uno mismo)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. Debe devolver una lista. No se puede llamar si se anula `render_to_response`.

Datos y otros atributos heredados
de [django.views.generic.list.MultipleObjectTemplateResponseMixin](#) :

template_name_suffix = '_list'

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el ``response_class`` para esta vista, con una plantilla renderizada con el contexto dado.

Pase `response_kwargs` al constructor de la clase de respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están definidas)

__weakref__

lista de referencias débiles al objeto (si está definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.list.BaseListView](#) :

get (self, request, * args, ** kwargs)

Métodos heredados de [django.views.generic.list.MultipleObjectMixin](#) :

get_allow_empty (auto)

Devuelve ``Verdadero`` si la vista debe mostrar listas vacías y ``Falso`` si se debe generar un 404 en su lugar.

get_context_object_name (self, object_list)

Obtenga el nombre del elemento que se utilizará en el contexto.

get_ordering (auto)

Devuelve el campo o los campos que se utilizarán para ordenar el conjunto de consultas.

get_paginate_by (self, queryset)

Obtenga la cantidad de elementos para paginar o ``Ninguno `` para no paginar.

get_paginate_orphans (self)

Devuelve el número máximo de huérfanos, extiende la última página al paginar.

get_paginator (self, queryset, per_page, huérfanos = 0, allow_empty_first_page = True, ** kwargs)

Devuelve una instancia del paginador para esta vista.

paginate_queryset (self, queryset, page_size)

Pagina el conjunto de consultas, si es necesario.

Datos y otros atributos heredados de [django.views.generic.list.MultipleObjectMixin](#) :

allow_empty = Verdadero

context_object_name = Ninguno

ordering = Ninguno

page_kwarg = 'página'

paginate_orphans = 0

paginator_class = <clase 'django.core.paginator.Paginator'>

queryset = Ninguno

Datos y otros atributos heredados de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']

5.3.3. **Clase EntidadUpdate** ([django.views.generic.edit.UpdateView](#))

[EntidadUpdate](#) (** kwargs)

Orden de resolución del método:

[EntidadUpdate](#)

[django.views.generic.edit.UpdateView](#)

[django.views.generic.detail.SingleObjectTemplateResponseMixin](#)

[django.views.generic.base.TemplateResponseMixin](#)

[django.views.generic.edit.BaseUpdateView](#)

[django.views.generic.edit.ModelFormMixin](#)

[django.views.generic.edit.FormMixin](#)

[django.views.generic.detail.SingleObjectMixin](#)

[django.views.generic.base.ContextMixin](#)

[django.views.generic.edit.ProcessFormView](#)

[django.views.generic.base.View](#)

[objeto incorporado](#)

Métodos definidos aquí:

despacho (self, * args, ** kwargs)

form_valid (self, form)

Si el formulario es válido, guarde el modelo asociado.

Datos y otros atributos definidos aquí:

form_class = <clase 'setup.forms.entidad.EntidadForm'>

modelo = <clase 'setup.models.entidad.Entidad'>

Entidad (id, nombre, codigo, categoria, ris, tipo)

Success_url = '/ setup / entidad / list'

Datos y otros atributos heredados de [django.views.generic.edit.UpdateView](#) :

template_name_suffix = '_form'

Métodos heredados

de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

get_template_names (self)

Devuelve una lista de nombres de plantillas que se utilizarán para la solicitud. No se puede llamar si se [anula render to response](#) (). Devuelve la siguiente lista:

* el valor de `` template_name '' en la vista (si se proporciona)

* el contenido del campo `` template_name_field '' en la

instancia del objeto sobre el que opera la vista (si está disponible)

* `` < app_label > / < model_name >

< template_name_suffix > .html ''

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectTemplateResponseMixin](#) :

template_name_field = Ninguno

Métodos heredados de [django.views.generic.base.TemplateResponseMixin](#) :

render_to_response (self, contexto, ** respuesta_kwargs)

Devuelve una respuesta, usando el `response_class`
para esta vista, con una
plantilla renderizada con el contexto dado.

Pase response_kwargs al constructor de la clase de
respuesta.

Descriptores de datos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

__dict__

diccionario, por ejemplo, variables (si están
definidas)

__weakref__

lista de referencias débiles al objeto (si está
definido)

Datos y otros atributos heredados
de [django.views.generic.base.TemplateResponseMixin](#) :

content_type = Ninguno

response_class = <clase 'django.template.response.TemplateResponse'>

template_engine = Ninguno

template_name = Ninguno

Métodos heredados de [django.views.generic.edit.BaseUpdateView](#) :

get (self, request, * args, ** kwargs)

Manejar solicitudes GET: instancia una versión en
blanco del formulario.

publicar (self, request, * args, ** kwargs)

Maneje las solicitudes POST: cree una instancia de
formulario con las

variables POST pasadas y luego verifique si es válida.

Métodos heredados de [django.views.generic.edit.ModelFormMixin](#) :

get_form_class (self)

Devuelve la clase de formulario para usar en esta vista.

get_form_kwargs (self)

Devuelve los argumentos de la palabra clave para crear instancias del formulario.

get_success_url (self)

Devuelve la URL a la que redireccionar después de procesar un formulario válido.

Datos y otros atributos heredados
de [django.views.generic.edit.ModelFormMixin](#) :

campos = Ninguno

Métodos heredados de [django.views.generic.edit.FormMixin](#) :

form_invalid (self, form)

Si el formulario no es válido, renderice el formulario no válido.

get_context_data (uno mismo, ** kwargs)

Inserte el formulario en el diccionario de contexto.

get_form (self, form_class = Ninguno)

Devuelve una instancia del formulario que se utilizará en esta vista.

get_initial (self)

Devuelve los datos iniciales para utilizarlos en los formularios de esta vista.

get_prefix (auto)

Devuelve el prefijo para usar en formularios.

Datos y otros atributos heredados de [django.views.generic.edit.FormMixin](#) :

inicial = {}

prefijo = Ninguno

Métodos heredados de [django.views.generic.detail.SingleObjectMixin](#) :

get_context_object_name (self, obj)

Obtenga el nombre que se utilizará para el objeto.

get_object (self, queryset = None)

Devuelve el objeto que muestra la vista.

Requerir `self. queryset` y un argumento `pk` o `slug` en la URLconf.

Las subclases pueden anular esto para devolver cualquier objeto.

get_queryset (self)

Devuelve el `QuerySet` que se utilizará para buscar el objeto.

Este método es llamado por la implementación predeterminada de [get_object](#) () y no puede ser llamado si [get_object](#) () está anulado.

get_slug_field (self)

Obtenga el nombre de un campo slug que se utilizará para buscar por slug.

Datos y otros atributos heredados
de [django.views.generic.detail.SingleObjectMixin](#) :

context_object_name = Ninguno

pk_url_kwarg = 'pk'

query_pk_and_slug = Falso

queryset = Ninguno

Datos y otros atributos heredados
de [django.views.generic.base.ContextMixin](#) :

extra_context = Ninguno

Métodos heredados de [django.views.generic.edit.ProcessFormView](#) :

poner (self, * argumentos, ** kwargs)

PUT es un verbo HTTP válido para crear (con una URL conocida) o editar un
objeto, tenga en cuenta que los navegadores solo admiten POST por ahora.

Métodos heredados de [django.views.generic.base.View](#) :

__init__ (self, ** kwargs)

Constructor. Llamado en la URLconf; puede contener argumentos de palabras clave adicionales útiles y otras cosas.

http_method_not_allowed (self, request, * args, ** kwargs)

opciones (self, request, * args, ** kwargs)

Manejar la respuesta a las solicitudes del verbo OPTIONS HTTP.

configuración (self, request, * args, ** kwargs)

Inicialice los atributos compartidos por todos los métodos de vista.

Métodos de clase heredados de [django.views.generic.base.View](#) :

as_view (** initkwargs) de [builtins.type](#)

Punto de entrada principal para un proceso de solicitud-respuesta.

Datos y otros atributos heredados de [django.views.generic.base.View](#) :

http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']