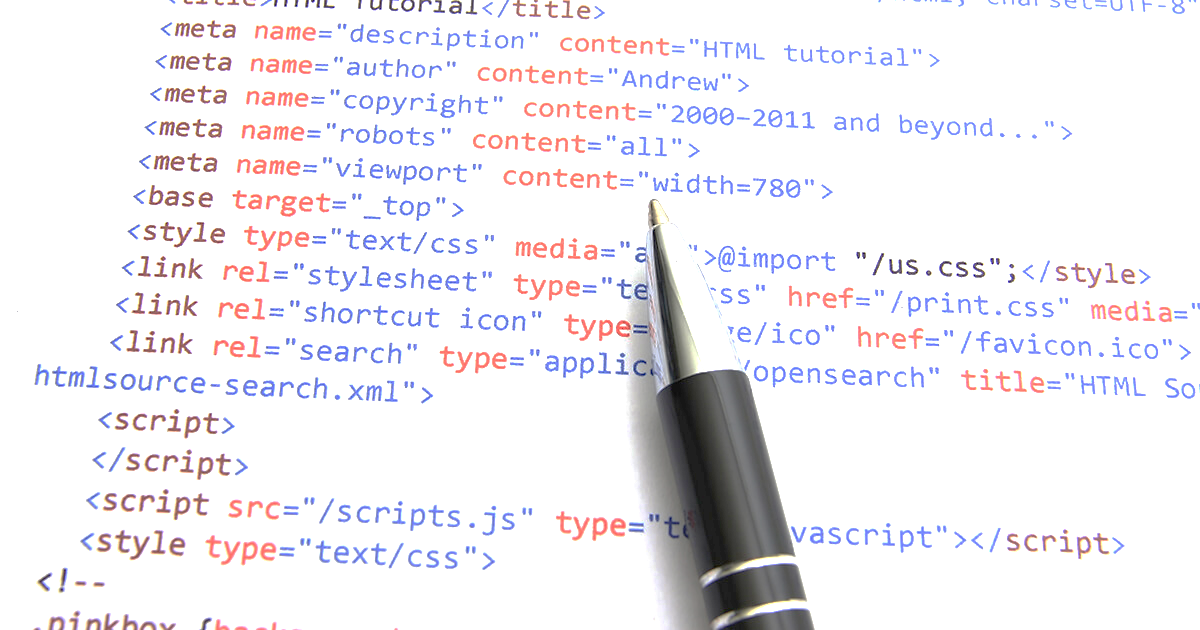
GESTION DE RECLAMOS

Reclamaciones v 1.2

Trama, Evaluaciones y Reportes

Documentación del código fuente de Reclamaciones y medidas adoptadas

*Por: Ing. Oscar David Mendoza Apaza*

****

**Contenido**

[**1.** **INTRODUCCIÓN** 3](#_Toc67306998)

[**2.** **PROPÓSITO** 3](#_Toc67306999)

[**3.** **ALCANCE** 3](#_Toc67307000)

[**4.** **CAPA DEL MODELO** 3](#_Toc67307001)

[4.1. **Clases GroupMenu** 3](#_Toc67307002)

[4.1.1. **Clase** **GroupMenu** ( django.db.models.base.Model ) 3](#_Toc67307003)

[4.1.2. **Clase Menu** ( django.db.models.base.Model ) 7](#_Toc67307004)

[4.2. **Clase Entidad** 10](#_Toc67307005)

[4.3. **Clase Periodo** 13](#_Toc67307006)

[4.4. **Clases RubroCalificacion** 17](#_Toc67307007)

[4.5. **Clases Usuario** 20](#_Toc67307008)

[4.5.1. **Clase** **Usuario** ( django.contrib.auth.models.AbstractUser ) 20](#_Toc67307009)

[4.6. **Clase ClasificacionCausa** 26](#_Toc67307010)

[4.6.1. **Clase** **ClasificacionCausa** ( django.db.models.base.Model ) 26](#_Toc67307011)

[4.7. **Clases EntidadReclamo** 29](#_Toc67307012)

[4.7.1. **Clase** **EntidadReclamo** ( django.db.models.base.Model ) 29](#_Toc67307013)

[4.8. **Clases DetalleEvaluacionAnexo1** 37](#_Toc67307014)

[4.8.1. **Clase** **DetalleEvaluacionAnexo1** ( django.db.models.base.Model ) 37](#_Toc67307015)

[4.8.2. **Clase** **EvaluacionAnexo1** ( django.db.models.base.Model ) 41](#_Toc67307016)

[**5.** **CAPA DE LA VISTA** 45](#_Toc67307017)

[5.1. **Clases EntidadReclamo** 45](#_Toc67307018)

[5.1.1. **Clase** **EntidadReclamoCreate** ( django.views.generic.edit.CreateView ) 45](#_Toc67307019)

[5.1.2. **Clase** **EntidadReclamoList** ( apps.util.generic\_filters.views.FilteredListView ) 49](#_Toc67307020)

[5.1.3. **Clase** **EntidadReclamoUpdate** ( django.views.generic.edit.UpdateView ) 54](#_Toc67307021)

[5.2. **Clases User** 58](#_Toc67307022)

[5.2.1. **Clase** **UserPasswordUpdate** ( django.views.generic.edit.UpdateView ) 58](#_Toc67307023)

[5.2.2. **Clase** **UsuarioCreate** ( django.views.generic.edit.CreateView ) 63](#_Toc67307024)

[5.2.3. **Clase** **UsuarioList** ( apps.util.generic\_filters.views.FilteredListView ) 68](#_Toc67307025)

[5.2.4. **Clase** **UsuarioUpdate** ( django.views.generic.edit.UpdateView ) 74](#_Toc67307026)

[5.3. **Clases Entidad** 79](#_Toc67307027)

[5.3.1. **Clase** **EntidadCreate** ( django.views.generic.edit.CreateView ) 79](#_Toc67307028)

[5.3.2. **Clase** **EntidadList** ( apps.util.generic\_filters.views.FilteredListView ) 85](#_Toc67307029)

[5.3.3. **Clase** **EntidadUpdate** ( django.views.generic.edit.UpdateView ) 90](#_Toc67307030)

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 Sistemas, 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.

1. **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.

1. **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

1. **CAPA DEL MODELO**

|  |  |  |
| --- | --- | --- |
| * 1. **Clases GroupMenu** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **GroupMenu** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [GroupMenu](http://localhost:1234/setup.models.menu.html#GroupMenu) (\* args, \*\* kwargs)   [GroupMenu](http://localhost:1234/setup.models.menu.html#GroupMenu) (id, grupo) | | |  | Orden de resolución del método:  [GroupMenu](http://localhost:1234/setup.models.menu.html#GroupMenu)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.menu.html" \l "GroupMenu-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](http://localhost:1234/setup.models.menu.html" \l "GroupMenu-clean_fields) (),  [clean](http://localhost:1234/setup.models.menu.html" \l "GroupMenu-clean) () y  [validate\_unique](http://localhost:1234/setup.models.menu.html" \l "GroupMenu-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) :  **\_\_dict\_\_**  diccionario, por ejemplo, variables (si están definidas)  **\_\_weakref\_\_**  lista de referencias débiles al objeto (si está definido)  **paquete** |  |  |  |  | | --- | --- | --- | | * + 1. **Clase Menu** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [Menú](http://localhost:1234/setup.models.menu.html#Menu) (\* args, \*\* kwargs)   [Menú](http://localhost:1234/setup.models.menu.html#Menu) (id, nombre, icono, url, título, padre, tipo) | | |  | Orden de resolución del método:  [Menú](http://localhost:1234/setup.models.menu.html#Menu)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.menu.html#Menu-type) (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](http://localhost:1234/setup.models.menu.html" \l "Menu-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](http://localhost:1234/setup.models.menu.html" \l "Menu-clean_fields) (),  [clean](http://localhost:1234/setup.models.menu.html" \l "Menu-clean) () y  [validate\_unique](http://localhost:1234/setup.models.menu.html" \l "Menu-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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')) |

|  |  |  |
| --- | --- | --- |
| * 1. **Clase Entidad** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **Entidad** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [Entidad](http://localhost:1234/setup.models.entidad.html#Entidad) (\* args, \*\* kwargs)   [Entidad](http://localhost:1234/setup.models.entidad.html#Entidad) (id, nombre, codigo, categoria, ris, tipo) | | |  | Orden de resolución del método:  [Entidad](http://localhost:1234/setup.models.entidad.html#Entidad)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  Métodos definidos aquí:  **\_\_str\_\_** (self)  Devuelve str (self).  **categoria** = <objeto django.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.DeferredAttribute object> **nombre** = <django.db.models.query\_utils.DeferredAttribute object> **ris** = <django.db.models.query\_utils.DeferredAttribute object> **tipo** = <django.db.models .query\_utils.DeferredAttribute objeto>  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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.entidad.html" \l "Entidad-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](http://localhost:1234/setup.models.entidad.html" \l "Entidad-clean_fields) (),  [clean](http://localhost:1234/setup.models.entidad.html" \l "Entidad-clean) () y  [validate\_unique](http://localhost:1234/setup.models.entidad.html" \l "Entidad-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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')) |

|  |  |  |
| --- | --- | --- |
| * 1. **Clase Periodo** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **Periodo** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [Periodo](http://localhost:1234/setup.models.periodo.html#Periodo) (\* args, \*\* kwargs)   [Periodo](http://localhost:1234/setup.models.periodo.html#Periodo) (id, periodo, fecha\_inicio, fecha\_fin, estado) | | |  | Orden de resolución del método:  [Periodo](http://localhost:1234/setup.models.periodo.html#Periodo)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.periodo.html" \l "Periodo-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](http://localhost:1234/setup.models.periodo.html" \l "Periodo-clean_fields) (),  [clean](http://localhost:1234/setup.models.periodo.html" \l "Periodo-clean) () y  [validate\_unique](http://localhost:1234/setup.models.periodo.html" \l "Periodo-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) :  **\_\_dict\_\_**  diccionario, por ejemplo, variables (si están definidas)  **\_\_weakref\_\_**  lista de referencias débiles al objeto (si está definido)  **paquete** | |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases RubroCalificacion** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **RubroCalificacion** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [RubroCalificacion](http://localhost:1234/setup.models.rubro_calificacion.html#RubroCalificacion) (\* args, \*\* kwargs)   [RubroCalificacion](http://localhost:1234/setup.models.rubro_calificacion.html#RubroCalificacion) (id, orden, rubro) | | |  | Orden de resolución del método:  [RubroCalificación](http://localhost:1234/setup.models.rubro_calificacion.html#RubroCalificacion)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.rubro_calificacion.html" \l "RubroCalificacion-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](http://localhost:1234/setup.models.rubro_calificacion.html" \l "RubroCalificacion-clean_fields) (),  [clean](http://localhost:1234/setup.models.rubro_calificacion.html" \l "RubroCalificacion-clean) () y  [validate\_unique](http://localhost:1234/setup.models.rubro_calificacion.html" \l "RubroCalificacion-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) :  **\_\_dict\_\_**  diccionario, por ejemplo, variables (si están definidas)  **\_\_weakref\_\_**  lista de referencias débiles al objeto (si está definido)  **paquete** | |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases Usuario** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **Usuario** ( [django.contrib.auth.models.AbstractUser](http://localhost:1234/django.contrib.auth.models.html" \l "AbstractUser) ) | | | |  | [Usuario](http://localhost:1234/setup.models.usuario.html#Usuario) (\* args, \*\* kwargs)   [Usuario](http://localhost:1234/setup.models.usuario.html#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](http://localhost:1234/setup.models.usuario.html#Usuario)  [django.contrib.auth.models.AbstractUser](http://localhost:1234/django.contrib.auth.models.html#AbstractUser)  [django.contrib.auth.base\_user.AbstractBaseUser](http://localhost:1234/django.contrib.auth.base_user.html#AbstractBaseUser)  [django.contrib.auth.models.PermissionsMixin](http://localhost:1234/django.contrib.auth.models.html#PermissionsMixin)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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 `` ForwardManyToOneDescriptor ''.  **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](http://localhost:1234/django.contrib.auth.models.html" \l "AbstractUser) :  **Clean** (self)  Hook para realizar cualquier validación adicional en todo el modelo después de que  se haya llamado a [clean](http://localhost:1234/setup.models.usuario.html" \l "Usuario-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](http://localhost:1234/django.contrib.auth.models.html" \l "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](http://localhost:1234/django.contrib.auth.base_user.html" \l "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](http://localhost:1234/setup.models.usuario.html" \l "Usuario-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](http://localhost:1234/django.contrib.auth.base_user.html" \l "AbstractBaseUser) :  **get\_email\_field\_name** () de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **normalize\_username** (nombre de usuario) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.contrib.auth.base\_user.AbstractBaseUser](http://localhost:1234/django.contrib.auth.base_user.html" \l "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](http://localhost:1234/django.contrib.auth.base_user.html" \l "AbstractBaseUser) :  **is\_active** = Verdadero  Métodos heredados de [django.contrib.auth.models.PermissionsMixin](http://localhost:1234/django.contrib.auth.models.html" \l "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](http://localhost:1234/setup.models.usuario.html" \l "Usuario-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](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/setup.models.usuario.html" \l "Usuario-clean_fields) (),  [clean](http://localhost:1234/setup.models.usuario.html" \l "Usuario-clean) () y  [validate\_unique](http://localhost:1234/setup.models.usuario.html" \l "Usuario-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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')) **SEXS** = ((0, 'FEMENINO '), (1,' MASCULINO ')) **TIPOS** = ((' DNI ',' DNI '), (' PASSPORD ',' PASAPORTE ')) |

|  |  |  |
| --- | --- | --- |
| * 1. **Clase ClasificacionCausa** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **ClasificacionCausa** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [ClasificacionCausa](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html#ClasificacionCausa) (\* args, \*\* kwargs)   [ClasificacionCausa](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html#ClasificacionCausa) (id, codigo, categoria, causa, definicion, definicion\_corta, created\_at, updated\_at) | | |  | Orden de resolución del método:  [ClasificacionCausa](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html#ClasificacionCausa)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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.MultipleObjectsReturned'>  **objetos** = <objeto django.db.models.manager.Manager>  Métodos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html" \l "ClasificacionCausa-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](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html" \l "ClasificacionCausa-clean_fields) (),  [clean](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html" \l "ClasificacionCausa-clean) () y  [validate\_unique](http://localhost:1234/apps.reclamo.models.clasificacion_causa.html" \l "ClasificacionCausa-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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')) |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases EntidadReclamo** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadReclamo** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [EntidadReclamo](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html#EntidadReclamo) (\* args, \*\* kwargs)   [EntidadReclamo](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html#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, motivodo\_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](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html#EntidadReclamo)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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\_correo** = <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\_reclamob** = <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\_utiltes. 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\_reclamob** = <django.db.models.query\_utils.DeferredAttribute object> **es\_mismo\_usuario\_afectado** = <django.db.models.query\_utils.DeferredAttribute object> **estado\_reclamob** = <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\_utildos.DeferredAttribute object> **fecha\_notificacion** = < django.db.models.query\_utils.DeferredAttribute object> **fecha\_reclamo** = <django.db.models.query\_utildos.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.MultipleObjectsReturned'>  **objetos** = <objeto django.db.models.manager.Manager>  Métodos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html" \l "EntidadReclamo-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](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html" \l "EntidadReclamo-clean_fields) (),  [clean](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html" \l "EntidadReclamo-clean) () y  [validate\_unique](http://localhost:1234/apps.reclamo.models.entidad_reclamo.html" \l "EntidadReclamo-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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, 'SETIEMBRE'), (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 QUIRDRGICO'), (5, 'CENTRO OBSTÉTRICO'), (6, 'UCI O UCIN') , (7, 'FARMACIA'), (8, 'SERVICIOS MÉDICOS DE APOYO'), (9, 'ATENCIÉN A DOMICILIO CONSULTA AMBULATORIA'), (10, 'ATENCIÉN A DOMICILIO URGENCIA A EMERGENCIA'), (11, 'OFICINAS O AREAS ADMINISTRATIVAS DE IAFAS A 1 PRENSA A UGIPRESS '), (12,' LNFRAESTRUCTURA '), (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')) |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases DetalleEvaluacionAnexo1** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **DetalleEvaluacionAnexo1** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [DetalleEvaluacionAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#DetalleEvaluacionAnexo1) (\* args, \*\* kwargs)   [DetalleEvaluacionAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#DetalleEvaluacionAnexo1) (id, evaluacion, rubro, calificacion, observacion, created\_at, updated\_at, created\_ip, updated\_ip, created\_by, updated\_by) | | |  | Orden de resolución del método:  [DetalleEvaluaciónAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#DetalleEvaluacionAnexo1)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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.evalua ... DetalleEvaluacionAnexo1.MultipleObjectsReturned'>  **objetos** = <objeto django.db.models.manager.Manager>  Métodos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "DetalleEvaluacionAnexo1-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](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "DetalleEvaluacionAnexo1-clean_fields) (),  [clean](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "DetalleEvaluacionAnexo1-clean) () y  [validate\_unique](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "DetalleEvaluacionAnexo1-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) :  **\_\_dict\_\_**  diccionario, por ejemplo, variables (si están definidas)  **\_\_weakref\_\_**  lista de referencias débiles al objeto (si está definido)  **paquete** |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EvaluacionAnexo1** ( [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) ) | | | |  | [EvaluacionAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#EvaluacionAnexo1) (\* args, \*\* kwargs)   [EvaluacionAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#EvaluacionAnexo1) (id, entidad, periodo, fecha, created\_at, updated\_at, created\_ip, updated\_ip, created\_by, updated\_by) | | |  | Orden de resolución del método:  [EvaluaciónAnexo1](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html#EvaluacionAnexo1)  [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html#Model)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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.evalua ... anexo1.EvaluacionAnexo1.MultipleObjectsReturned'>  **objetos** = <objeto django.db.models.manager.Manager>  Métodos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "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](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "EvaluacionAnexo1-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](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "EvaluacionAnexo1-clean_fields) (),  [clean](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "EvaluacionAnexo1-clean) () y  [validate\_unique](http://localhost:1234/apps.reclamo_administrador.models.evaluacion_anexo1.html" \l "EvaluacionAnexo1-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](http://localhost:1234/django.db.models.base.html" \l "Model) :  **comprobar** (\*\* kwargs) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  **from\_db** (db, field\_names, values) de [django.db.models.base.ModelBase](http://localhost:1234/django.db.models.base.html" \l "ModelBase)  Descriptores de datos heredados de [django.db.models.base.Model](http://localhost:1234/django.db.models.base.html" \l "Model) :  **\_\_dict\_\_**  diccionario, por ejemplo, variables (si están definidas)  **\_\_weakref\_\_**  lista de referencias débiles al objeto (si está definido)  **paquete** | |

1. **CAPA DE LA VISTA**

|  |  |  |
| --- | --- | --- |
| * 1. **Clases EntidadReclamo** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadReclamoCreate** ( [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html#CreateView) ) | | | |  | [EntidadReclamoCreate](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoCreate) (\*\* kwargs)   Vista para crear un nuevo objeto, con una respuesta renderizada por una plantilla. | | |  | Orden de resolución del método:  [EntidadReclamoCrear](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoCreate)  [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html#CreateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseCreateView](http://localhost:1234/django.views.generic.edit.html#BaseCreateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "CreateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoCreate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoCreate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoCreate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadReclamoList** ( [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView) ) | | | |  | [EntidadReclamoList](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoList) (\*\* kwargs)   Un ListView genérico utilizado para filtrar y ordenar objetos. | | |  | Orden de resolución del método:  [EntidadReclamoList](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoList)  [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.list.ListView](http://localhost:1234/django.views.generic.list.html#ListView)  [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.list.BaseListView](http://localhost:1234/django.views.generic.list.html#BaseListView)  [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  Métodos definidos aquí:  **get\_context\_data** (self \*\* kwargs)  Agregue una lista de filtros y uno mismo. F**ormulario** al contexto que será representado por la vista.  **get\_queryset** (self)  Devuelve el conjunto de consultas filtrado. Utiliza  [form\_valid](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoList-form_valid) () o  [form\_invalid](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoList-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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/apps.util.generic_filters.views.html" \l "FilteredListView) :  **default\_filter** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  **success\_url** = Ninguno  Métodos heredados de [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#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](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin) :  **template\_name\_suffix** = '\_list'  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "BaseListView) :  **get** (self, request, \* args, \*\* kwargs)  Métodos heredados de [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadReclamoUpdate** ( [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView) ) | | | |  | [EntidadReclamoUpdate](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoUpdate) (\*\* kwargs)   Vista para actualizar un objeto, con una respuesta renderizada por una plantilla. | | |  | Orden de resolución del método:  [EntidadReclamoUpdate](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoUpdate)  [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseUpdateView](http://localhost:1234/django.views.generic.edit.html#BaseUpdateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html#EntidadReclamoUpdate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoUpdate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/apps.reclamo.views.entidad_reclamo.html" \l "EntidadReclamoUpdate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "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'] |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases User** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **UserPasswordUpdate** ( [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView) ) | | | |  | [UserPasswordUpdate](http://localhost:1234/setup.views.usuario.html#UserPasswordUpdate) (\*\* kwargs) | | |  | Orden de resolución del método:  [UserPasswordUpdate](http://localhost:1234/setup.views.usuario.html#UserPasswordUpdate)  [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseUpdateView](http://localhost:1234/django.views.generic.edit.html#BaseUpdateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/setup.views.usuario.html#UserPasswordUpdate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "TemplateResponseMixin) :  **content\_type** = Ninguno  **response\_class** = <clase 'django.template.response.TemplateResponse'>  **template\_engine** = Ninguno  Métodos heredados de [django.views.generic.edit.BaseUpdateView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  **success\_url** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/setup.views.usuario.html" \l "UserPasswordUpdate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/setup.views.usuario.html" \l "UserPasswordUpdate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **UsuarioCreate** ( [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html" \l "CreateView) ) | | | |  | [UsuarioCreate](http://localhost:1234/setup.views.usuario.html#UsuarioCreate) (\*\* kwargs) | | |  | Orden de resolución del método:  [UsuarioCreate](http://localhost:1234/setup.views.usuario.html#UsuarioCreate)  [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html#CreateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseCreateView](http://localhost:1234/django.views.generic.edit.html#BaseCreateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "CreateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/setup.views.usuario.html#UsuarioCreate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/setup.views.usuario.html" \l "UsuarioCreate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/setup.views.usuario.html" \l "UsuarioCreate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **UsuarioList** ( [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView) ) | | | |  | [UsuarioList](http://localhost:1234/setup.views.usuario.html#UsuarioList) (\*\* kwargs) | | |  | Orden de resolución del método:  [UsuarioList](http://localhost:1234/setup.views.usuario.html#UsuarioList)  [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.list.ListView](http://localhost:1234/django.views.generic.list.html#ListView)  [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.list.BaseListView](http://localhost:1234/django.views.generic.list.html#BaseListView)  [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/setup.views.usuario.html" \l "UsuarioList-form_valid) () o  [form\_invalid](http://localhost:1234/setup.views.usuario.html" \l "UsuarioList-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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/apps.util.generic_filters.views.html" \l "FilteredListView) :  **default\_filter** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  **success\_url** = Ninguno  Métodos heredados de [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#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](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin) :  **template\_name\_suffix** = '\_list'  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "BaseListView) :  **get** (self, request, \* args, \*\* kwargs)  Métodos heredados de [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **UsuarioUpdate** ( [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) ) | | | |  | [UsuarioUpdate](http://localhost:1234/setup.views.usuario.html#UsuarioUpdate) (\*\* kwargs) | | |  | Orden de resolución del método:  [UsuarioUpdate](http://localhost:1234/setup.views.usuario.html#UsuarioUpdate)  [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseUpdateView](http://localhost:1234/django.views.generic.edit.html#BaseUpdateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/setup.views.usuario.html#UsuarioUpdate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/setup.views.usuario.html" \l "UsuarioUpdate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/setup.views.usuario.html" \l "UsuarioUpdate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] | |

|  |  |  |
| --- | --- | --- |
| * 1. **Clases Entidad** | | |
|  |  | |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadCreate** ( [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html" \l "CreateView) ) | | | |  | [EntidadCreate](http://localhost:1234/setup.views.entidad.html#EntidadCreate) (\*\* kwargs) | | |  | Orden de resolución del método:  [EntidadCreate](http://localhost:1234/setup.views.entidad.html#EntidadCreate)  [django.views.generic.edit.CreateView](http://localhost:1234/django.views.generic.edit.html#CreateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseCreateView](http://localhost:1234/django.views.generic.edit.html#BaseCreateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "CreateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/setup.views.entidad.html#EntidadCreate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/setup.views.entidad.html" \l "EntidadCreate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/setup.views.entidad.html" \l "EntidadCreate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadList** ( [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView) ) | | | |  | [EntidadList](http://localhost:1234/setup.views.entidad.html#EntidadList) (\*\* kwargs) | | |  | Orden de resolución del método:  [EntidadList](http://localhost:1234/setup.views.entidad.html#EntidadList)  [apps.util.generic\_filters.views.FilteredListView](http://localhost:1234/apps.util.generic_filters.views.html#FilteredListView)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.list.ListView](http://localhost:1234/django.views.generic.list.html#ListView)  [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.list.BaseListView](http://localhost:1234/django.views.generic.list.html#BaseListView)  [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html#MultipleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/setup.views.entidad.html" \l "EntidadList-form_valid) () o  [form\_invalid](http://localhost:1234/setup.views.entidad.html" \l "EntidadList-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](http://localhost:1234/apps.util.generic_filters.views.html" \l "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](http://localhost:1234/apps.util.generic_filters.views.html" \l "FilteredListView) :  **default\_filter** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  **success\_url** = Ninguno  Métodos heredados de [django.views.generic.list.MultipleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.list.html#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](http://localhost:1234/django.views.generic.list.html#MultipleObjectTemplateResponseMixin) :  **template\_name\_suffix** = '\_list'  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "BaseListView) :  **get** (self, request, \* args, \*\* kwargs)  Métodos heredados de [django.views.generic.list.MultipleObjectMixin](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.list.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] |  |  |  |  | | --- | --- | --- | | * + 1. **Clase** **EntidadUpdate** ( [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) ) | | | |  | [EntidadUpdate](http://localhost:1234/setup.views.entidad.html#EntidadUpdate) (\*\* kwargs) | | |  | Orden de resolución del método:  [EntidadUpdate](http://localhost:1234/setup.views.entidad.html#EntidadUpdate)  [django.views.generic.edit.UpdateView](http://localhost:1234/django.views.generic.edit.html#UpdateView)  [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin)  [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html#TemplateResponseMixin)  [django.views.generic.edit.BaseUpdateView](http://localhost:1234/django.views.generic.edit.html#BaseUpdateView)  [django.views.generic.edit.ModelFormMixin](http://localhost:1234/django.views.generic.edit.html#ModelFormMixin)  [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html#FormMixin)  [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html#SingleObjectMixin)  [django.views.generic.base.ContextMixin](http://localhost:1234/django.views.generic.base.html#ContextMixin)  [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html#ProcessFormView)  [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html#View)  [objeto incorporado](http://localhost:1234/builtins.html#object)  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](http://localhost:1234/django.views.generic.edit.html" \l "UpdateView) :  **template\_name\_suffix** = '\_form'  Métodos heredados de [django.views.generic.detail.SingleObjectTemplateResponseMixin](http://localhost:1234/django.views.generic.detail.html#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](http://localhost:1234/setup.views.entidad.html#EntidadUpdate-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](http://localhost:1234/django.views.generic.detail.html#SingleObjectTemplateResponseMixin) :  **template\_name\_field** = Ninguno  Métodos heredados de [django.views.generic.base.TemplateResponseMixin](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "ModelFormMixin) :  **campos** = Ninguno  Métodos heredados de [django.views.generic.edit.FormMixin](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.edit.html" \l "FormMixin) :  **inicial** = {}  **prefijo** = Ninguno  Métodos heredados de [django.views.generic.detail.SingleObjectMixin](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/setup.views.entidad.html" \l "EntidadUpdate-get_object) () y no puede ser llamado si  [get\_object](http://localhost:1234/setup.views.entidad.html" \l "EntidadUpdate-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](http://localhost:1234/django.views.generic.detail.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "ContextMixin) :  **extra\_context** = Ninguno  Métodos heredados de [django.views.generic.edit.ProcessFormView](http://localhost:1234/django.views.generic.edit.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "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](http://localhost:1234/django.views.generic.base.html" \l "View) :  **as\_view** (\*\* initkwargs) de [builtins.type](http://localhost:1234/builtins.html" \l "type)  Punto de entrada principal para un proceso de solicitud-respuesta.  Datos y otros atributos heredados de [django.views.generic.base.View](http://localhost:1234/django.views.generic.base.html" \l "View) :  **http\_method\_names** = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace'] | |