CoopeV3 Documentation

Release 3.4.0

Yoann Pietri

Feb 28, 2019

Contents:

CHAPTER 1

Views documentation

1.1 Gestion app views

class gestion.views.**ActiveProductsAutocomplete**(***kwargs*)
  Autocomplete view for active `products`.

  get_queryset()
    Filter the queryset with GET['q'].

class gestion.views.**KegActiveAutocomplete**(***kwargs*)
  Autocomplete view for active `kegs`.

  get_queryset()
    Filter the queryset with GET['q'].

class gestion.views.**KegPositiveAutocomplete**(***kwargs*)
  Autocomplete view for `kegs` with positive stockHold.

  get_queryset()
    Filter the queryset with GET['q'].

class gestion.views.**MenusAutocomplete**(***kwargs*)
  Autocomplete view for active `menus`.

  get_queryset()
    Filter the queryset with GET['q'].

class gestion.views.**ProductsAutocomplete**(***kwargs*)
  Autocomplete view for all `products`.

  get_queryset()
    Filter the queryset with GET['q'].

gestion.views.**addKeg**(*request*)
  Displays a `gestion.forms.KegForm` to add a `gestion.models.Keg`.

gestion.views.**addMenu**(*request*)
  Display a `gestion.forms.MenuForm` to add a `gestion.models.Menu`.

`gestion.views.`**`addProduct`**`(`*request*`)`
Displays a *`gestion.forms.ProductForm`* to add a product.

`gestion.views.`**`add_pintes`**`(`*request*`)`
Displays a *`gestion.forms.PinteForm`* to add one or more *`gestion.models.Pinte`*.

`gestion.views.`**`allocate`**`(`*pinte_pk*, *user*`)`
Allocate a *`gestion.models.Pinte`* to a user (`django.contrib.auth.models.User`) or release the pinte if user is None

`gestion.views.`**`cancel_consumption`**`(`*request*, *pk*`)`
Delete a *`consumption history`*.

pk The primary key of the consumption history to delete.

`gestion.views.`**`cancel_menu`**`(`*request*, *pk*`)`
Delete a *`menu history`*.

pk The primary key of the menu history to delete.

`gestion.views.`**`cancel_reload`**`(`*request*, *pk*`)`
Delete a *`gestion.models.Reload`*.

pk The primary key of the reload to delete.

`gestion.views.`**`closeDirectKeg`**`(`*request*, *pk*`)`
Closes a class:*gestion.models.Keg*.

pk The primary key of the class:*gestion.models.Keg* to open.

`gestion.views.`**`closeKeg`**`(`*request*`)`
Displays a *`gestion.forms.SelectPositiveKegForm`* to open a *`gestion.models.Keg`*.

`gestion.views.`**`editKeg`**`(`*request*, *pk*`)`
Displays a *`gestion.forms.KegForm`* to edit a *`gestion.models.Keg`*.

pk The primary key of the *`gestion.models.Keg`* to edit.

`gestion.views.`**`editProduct`**`(`*request*, *pk*`)`
Displays a *`gestion.forms.ProductForm`* to edit a product.

pk The primary key of the the *`gestion.models.Product`* to edit.

`gestion.views.`**`edit_menu`**`(`*request*, *pk*`)`
Displays a *`gestion.forms.MenuForm`* to edit a *`gestion.models.Menu`*.

pk The primary key of the *`gestion.models.Menu`* to edit.

`gestion.views.`**`gen_releve`**`(`*request*`)`
Displays a `forms.gestion.GenerateReleveForm` to generate a releve.

`gestion.views.`**`getProduct`**`(`*request*, *pk*`)`
Get a *`gestion.models.Product`* by barcode and return it in JSON format.

pk The primary key of the *`gestion.models.Product`* to get infos.

`gestion.views.`**`get_menu`**`(`*request*, *pk*`)`
Get a *`gestion.models.Menu`* by pk and return it in JSON format.

pk The primary key of the *`gestion.models.Menu`*.

`gestion.views.`**`kegH`**`(`*request*, *pk*`)`
Display the *`history`* of requested *`gestion.models.Keg`*.

`gestion.views.`**`kegsList`**`(`*request*`)`
Display the list of *`kegs`*.

`gestion.views.`**`manage`**`(`*request*`)`
Displays the manage view.

`gestion.views.`**`menus_list`**`(`*request*`)`
Display the list of `menus`.

`gestion.views.`**`openDirectKeg`**`(`*request*, *pk*`)`
Opens a class:*gestion.models.Keg*.

pk The primary key of the class:*gestion.models.Keg* to open.

`gestion.views.`**`openKeg`**`(`*request*`)`
Displays a `gestion.forms.SelectPositiveKegForm` to open a `gestion.models.Keg`.

`gestion.views.`**`order`**`(`*request*`)`
Processes the given order. The order is passed through POST.

`gestion.views.`**`pintes_list`**`(`*request*`)`
Displays the list of `gestion.models.Pinte`

`gestion.views.`**`pintes_user_list`**`(`*request*`)`
Displays the list of user, who have unreturned `Pinte(s)`.

`gestion.views.`**`productProfile`**`(`*request*, *pk*`)`
Displays the profile of a `gestion.models.Product`.

pk The primary key of the `gestion.models.Product` to display profile.

`gestion.views.`**`productsIndex`**`(`*request*`)`
Displays the products manage static page.

`gestion.views.`**`productsList`**`(`*request*`)`
Display the list of `products`.

`gestion.views.`**`ranking`**`(`*request*`)`
Displays the ranking page.

`gestion.views.`**`refund`**`(`*request*`)`
Displays a `Refund form`.

`gestion.views.`**`release`**`(`*request*, *pinte_pk*`)`
View to release a `gestion.models.Pinte`.

pinte_pk The primary key of the `gestion.models.Pinte` to release.

`gestion.views.`**`release_pintes`**`(`*request*`)`
Displays a `gestion.forms.PinteForm` to release one or more `gestion.models.Pinte`.

`gestion.views.`**`reload`**`(`*request*`)`
Displays a `Reload form`.

`gestion.views.`**`searchMenu`**`(`*request*`)`
Displays a `gestion.forms.SearchMenuForm` to search a `gestion.models.Menu`.

`gestion.views.`**`searchProduct`**`(`*request*`)`
Displays a `gestion.forms.SearchProduct` to search a `gestion.models.Product`.

`gestion.views.`**`switch_activate`**`(`*request*, *pk*`)`
Switch the active status of the requested `gestion.models.Product`.

pk The primary key of the `gestion.models.Product` to display profile.

`gestion.views.`**`switch_activate_menu`**`(`*request*, *pk*`)`
Switch active status of a `gestion.models.Menu`.

1.2 Users app views

class `users.views.`**`ActiveUsersAutocomplete`**(***kwargs*)

Autocomplete for active users (`django.contrib.auth.models.User`).

`get_queryset`()

Filter the queryset with GET['q'].

class `users.views.`**`AdherentAutocomplete`**(***kwargs*)

Autocomplete for adherents (`django.contrib.auth.models.User`).

`get_queryset`()

Filter the queryset with GET['q'].

class `users.views.`**`AllUsersAutocomplete`**(***kwargs*)

Autcomplete for all users (`django.contrib.auth.models.User`).

`get_queryset`()

Filter the queryset with GET['q'].

class `users.views.`**`NonAdminUserAutocomplete`**(***kwargs*)

Autocomplete for non-admin users (`django.contrib.auth.models.User`).

`get_queryset`()

Filter the queryset with GET['q'].

class `users.views.`**`NonSuperUserAutocomplete`**(***kwargs*)

Autocomplete for non-superuser users (`django.contrib.auth.models.User`).

`get_queryset`()

Filter the queryset with GET['q'].

`users.views.`**`addAdmin`**(*request*)

Displays a *`users.forms.SelectNonAdminUserForm`* to select a non admin user (`django.contrib.auth.models.User`) and add it to the admins.

`users.views.`**`addCotisationHistory`**(*request*, *pk*)

Displays a *`users.forms.addCotisationHistoryForm`* to add a `Cotisation History <users.models.CotisationHistory` to the requested user (`django.contrib.auth.models.User`).

pk The primary key of the user to add a cotisation history

`users.views.`**`addSuperuser`**(*request*)

Displays a *`users.forms.SelectNonAdminUserForm`* to select a non superuser user (`django.contrib.auth.models.User`) and add it to the superusers.

`users.views.`**`addWhiteListHistory`**(*request*, *pk*)

Displays a *`users.forms.addWhiteListHistoryForm`* to add a *`WhiteListHistory`* to the requested user (`django.contrib.auth.models.User`).

`users.views.`**`adminsIndex`**(*request*)

Lists the staff (`django.contrib.auth.models.User` with is_staff True)

`users.views.`**`allReloads`**(*request*, *pk*, *page*)

Display all the *`reloads`* of the requested user (`django.contrib.auth.models.User`).

`users.views.`**`all_consumptions`**(*request*, *pk*, *page*)

Display all the *consumptions <gestion.models.ConsumptionHistory>* of the requested user (`django.contrib.auth.models.User`).

`users.views.`**`all_menus`**(*request*, *pk*, *page*)
: Display all the *menus <gestion.models.MenuHistory>* of the requested user (`django.contrib.auth.models.User`).

`users.views.`**`createGroup`**(*request*)
: Displays a *`CreateGroupForm`* to create a group (`django.contrib.auth.models.Group`).

`users.views.`**`createSchool`**(*request*)
: Displays *`SchoolForm`* to add a *`School`*.

`users.views.`**`createUser`**(*request*)
: Displays a *`CreateUserForm`* to create a user (`django.contrib.auth.models.User`).

`users.views.`**`deleteCotisationHistory`**(*request*, *pk*)
: Delete the requested *`CotisationHistory`*.

  pk The primary key of tthe CotisationHistory to delete.

`users.views.`**`deleteGroup`**(*request*, *pk*)
: Deletes the requested group (`django.contrib.auth.models.Group`).

  pk The primary key of the group to delete

`users.views.`**`deleteSchool`**(*request*, *pk*)
: Deletes a *`users.models.School`*.

  pk The primary key of the School to delete.

`users.views.`**`editGroup`**(*request*, *pk*)
: Displays a *`EditGroupForm`* to edit a group (`django.contrib.auth.models.Group`).

  pk The primary key of the group to edit.

`users.views.`**`editGroups`**(*request*, *pk*)
: Displays a `users.form.GroupsEditForm` to edit the groups of a user (`django.contrib.auth.models.User`).

`users.views.`**`editPassword`**(*request*, *pk*)
: Displays a `users.form.EditPasswordForm` to edit the password of a user (`django.contrib.auth.models.User`).

`users.views.`**`editSchool`**(*request*, *pk*)
: Displays *`SchoolForm`* to edit a *`School`*.

  pk The primary key of the school to edit.

`users.views.`**`editUser`**(*request*, *pk*)
: Displays a *`CreateUserForm`* to edit a user (`django.contrib.auth.models.User`).

`users.views.`**`export_csv`**(*request*)
: Displays a *`users.forms.ExportForm`* to export csv files of users.

`users.views.`**`gen_user_infos`**(*request*, *pk*)
: Generates a latex document include adhesion certificate and list of *cotisations <users.models.CotisationHistory>*.

`users.views.`**`getUser`**(*request*, *pk*)
: Get requested user (`django.contrib.auth.models.User`) and return username, balance and is_adherent in JSON format.

  pk The primary key of the user to get infos.

`users.views.`**`groupProfile`**(*request*, *pk*)
: Displays the profile of a group (`django.contrib.auth.models.Group`).

`users.views.`**`groupsIndex`**(*request*)
Displays all the groups (`django.contrib.auth.models.Group`).

`users.views.`**`index`**(*request*)
Display the index for user related actions.

`users.views.`**`loginView`**(*request*)
Displays the *`users.forms.LoginForm`*.

`users.views.`**`logoutView`**(*request*)
Logout the logged user (`django.contrib.auth.models.User`).

`users.views.`**`profile`**(*request*, *pk*)
Displays the profile for the requested user (`django.contrib.auth.models.User`).

pk The primary key of the user (`django.contrib.auth.models.User`) to display profile

`users.views.`**`removeAdmin`**(*request*, *pk*)
Removes an user (`django.contrib.auth.models.User`) from staff.

pk The primary key of the user (`django.contrib.auth.models.User`) to remove from staff

`users.views.`**`removeRight`**(*request*, *groupPk*, *permissionPk*)
Removes a right from a given group (`django.contrib.auth.models.Group`).

`users.views.`**`removeSuperuser`**(*request*, *pk*)
Removes a user (`django.contrib.auth.models.User`) from superusers.

`users.views.`**`removeUser`**(*request*, *groupPk*, *userPk*)
Removes a user (`django.contrib.auth.models.User`) from a given group (`django.contrib.auth.models.Group`).

`users.views.`**`resetPassword`**(*request*, *pk*)
Reset the password of a user (`django.contrib.auth.models.User`).

`users.views.`**`schoolsIndex`**(*request*)
Lists the *`Schools`*.

`users.views.`**`searchUser`**(*request*)
Displays a *`SelectUserForm`* to search a user (`django.contrib.auth.models.User`).

`users.views.`**`superusersIndex`**(*request*)
Lists the superusers (`django.contrib.auth.models.User` with is_superuser True).

`users.views.`**`switch_activate_user`**(*request*, *pk*)
Switch the active status of the requested user (`django.contrib.auth.models.User`).

pk The primary key of the user to switch status

`users.views.`**`usersIndex`**(*request*)
Display the list of all users (`django.contrib.auth.models.User`).

1.3 Preferences app views

`preferences.views.`**`addCotisation`**(*request*)
View which displays a *`CotisationForm`* to create a *`Cotisation`*.

`preferences.views.`**`addPaymentMethod`**(*request*)
View which displays a *`PaymentMethodForm`* to create a *`PaymentMethod`*.

`preferences.views.`**`cotisationsIndex`**(*request*)
View which lists all the *`Cotisation`*.

preferences.views.**deleteCotisation**(*request*, *pk*)
Delete a `Cotisation`.

pk The primary key of the `Cotisation` to delete.

preferences.views.**deletePaymentMethod**(*request*, *pk*)
Delete a `PaymentMethod`.

pk The primary key of the `PaymentMethod` to delete.

preferences.views.**editCotisation**(*request*, *pk*)
View which displays a `CotisationForm` to edit a `Cotisation`.

pk The primary key of the `Cotisation` to edit.

preferences.views.**editPaymentMethod**(*request*, *pk*)
View which displays a `PaymentMethodForm` to edit a `PaymentMethod`.

pk The primary key of the `PaymentMethod` to edit.

preferences.views.**generalPreferences**(*request*)
View which displays a `GeneralPreferencesForm` to edit the `GeneralPreferences`.

preferences.views.**get_config**(*request*)
Load the `GeneralPreferences` and return it in json format (except for `statutes`, `rules` and `menu`)

preferences.views.**get_cotisation**(*request*, *pk*)
Return the requested `Cotisation` in json format.

pk The primary key of the requested `Cotisation`.

preferences.views.**inactive**(*request*)
View which displays the inactive message (if the site is inactive).

preferences.views.**paymentMethodsIndex**(*request*)
View which lists all the `PaymentMethod`.

1.4 coopeV3 app views

coopeV3.views.**coope_runner**(*request*)
Just an easter egg

coopeV3.views.**home**(*request*)
Redirect the user either to `manage()` view (if connected and staff) or `homepage()` view (if connected and not staff) or `loginView()` view (if not connected).

coopeV3.views.**homepage**(*request*)
View which displays the `home_text` and active `Kegs`.

CHAPTER 2

Models documentation

2.1 Gestion app models

class gestion.models.**Consumption**(**args*, ***kwargs*)

Stores total consumptions.

exception DoesNotExist

exception MultipleObjectsReturned

customer

Client (django.contrib.auth.models.User).

customer_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

product

A *gestion.models.Product* instance.

product_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

quantity

The total number of *gestion.models.Consumption.product* consumed by the gestion.models.Consumption.consumer.

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**ConsumptionHistory**(**args*, ***kwargs*)

Stores consumption history related to Product

exception DoesNotExist

exception MultipleObjectsReturned

amount

Price of the purchase.

coopeman

Coopeman (:class:django.contrib.auth.models.User') who collected the money.

coopeman_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer

Client (`django.contrib.auth.models.User`).

customer_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date

Date of the purhcase.

get_next_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=True*, ***kwargs*)

get_previous_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

paymentMethod

Payment Method of the product purchased.

paymentMethod_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

product

`gestion.models.product` purchased.

product_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

quantity

Quantity of *gestion.models.ConsumptionHistory.product* taken.

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**HistoricalConsumption**(*id*, *quantity*, *customer*, *product*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

customer

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of `Consumption`

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

product
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

product_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

quantity
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalConsumptionHistory**(*id*, *quantity*, *date*, *amount*, *customer*, *paymentMethod*, *product*, *coopeman*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=True, **kwargs*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=False, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *ConsumptionHistory*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

paymentMethod
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

paymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

product
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

product_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

quantity
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalKeg**(*id*, *name*, *stockHold*, *barcode*, *amount*, *capacity*, *is_active*, *pinte*, *demi*, *galopin*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

barcode
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

capacity
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

demi
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

demi_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

galopin
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

galopin_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *Keg*

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

pinte
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

pinte_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

stockHold
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class gestion.models.**HistoricalKegHistory**(*id*, *openingDate*, *quantitySold*, *amountSold*, *closingDate*, *isCurrentKegHistory*, *keg*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amountSold
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

closingDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_next_by_openingDate(**, field=<django.db.models.fields.DateTimeField: openingDate>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

get_previous_by_openingDate(**, field=<django.db.models.fields.DateTimeField: openingDate>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *KegHistory*

isCurrentKegHistory
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

keg
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

keg_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

openingDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

quantitySold
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalMenu**(*id*, *name*, *amount*, *barcode*, *is_active*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

barcode

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of `Menu`

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalMenuHistory**(*id*, *quantity*, *date*, *amount*, *customer*, *paymentMethod*, *menu*, *coopeman*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=True, **kwargs*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=False, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *MenuHistory*

menu

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

menu_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record

Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

paymentMethod

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

paymentMethod_id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record

Get the previous history record for the instance. *None* if first.

quantity

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()

URL for this change in the default admin site.

class gestion.models.**HistoricalPinte**(*id*, *last_update_date*, *current_owner*, *previous_owner*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

current_owner

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

current_owner_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_next_by_last_update_date(**, field=<django.db.models.fields.DateTimeField: last_update_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

get_previous_by_last_update_date(**, field=<django.db.models.fields.DateTimeField: last_update_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *`Pinte`*

last_update_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

previous_owner
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

previous_owner_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalProduct**(*id*, *name*, *amount*, *stockHold*, *stockBar*, *barcode*, *category*, *needQuantityButton*, *is_active*, *volume*, *deg*, *adherentRequired*, *showingMultiplier*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

adherentRequired
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

barcode
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

category
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

deg
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_category_display(**, field=<django.db.models.fields.CharField: category>*)

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *Product*

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

needQuantityButton
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

showingMultiplier
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

stockBar
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

stockHold
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

volume
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class gestion.models.**HistoricalRefund**(*id*, *date*, *amount*, *customer*, *coopeman*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=True, **kwargs*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=False, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *Refund*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class gestion.models.**HistoricalReload**(*id*, *amount*, *date*, *customer*, *PaymentMethod*, *coopeman*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

PaymentMethod
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

PaymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`customer_id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`date`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_history_type_display`(**, field=<django.db.models.fields.CharField: history_type>*)

`get_next_by_date`(**, field=<django.db.models.fields.DateTimeField: date>, is_next=True, **kwargs*)

`get_next_by_history_date`(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

`get_previous_by_date`(**, field=<django.db.models.fields.DateTimeField: date>, is_next=False, **kwargs*)

`get_previous_by_history_date`(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

`history_change_reason`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`history_date`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`history_id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`history_object`

`history_type`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`history_user`
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`history_user_id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`instance`

instance_type
alias of *Reload*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class gestion.models.**Keg**(**args*, ***kwargs*)
Stores a keg.

exception DoesNotExist

exception MultipleObjectsReturned

amount
The price of the keg.

barcode
The barcode of the keg.

capacity
The capacity, in liters, of the keg.

demi
The related *Product* for demi.

demi_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

galopin
The related *Product* for galopin.

galopin_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active
If True, will be displayed on *manage()* view

keghistory_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`name`
The name of the keg.

`objects = <django.db.models.manager.Manager object>`

`pinte`
The related `Product` for pint.

`pinte_id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`save_without_historical_record`(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

`stockHold`
The number of this keg in the hold.

class `gestion.models.`**`KegHistory`**(**args*, ***kwargs*)
Stores a keg history, related to `Keg`.

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`amountSold`
The quantity, in euros, sold.

`closingDate`
The date when the keg was closed

`get_next_by_openingDate`(*, *field=<django.db.models.fields.DateTimeField: openingDate>*, *is_next=True*, ***kwargs*)

`get_previous_by_openingDate`(*, *field=<django.db.models.fields.DateTimeField: openingDate>*, *is_next=False*, ***kwargs*)

`history = <simple_history.manager.HistoryManager object>`

`id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`isCurrentKegHistory`
If True, it corresponds to the current Keg history of `Keg` instance.

`keg`
The `Keg` instance.

`keg_id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects = <django.db.models.manager.Manager object>`

`openingDate`
The date when the keg was opened.

`quantitySold`
The quantity, in liters, sold.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**Menu**(**args*, ***kwargs*)
Stores menus.

exception DoesNotExist

exception MultipleObjectsReturned

adherent_required
Test if the menu contains a restricted `Product`

amount
Price of the menu.

articles
Stores `Products` contained in the menu

barcode
Barcode of the menu.

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active
If True, the menu will be displayed on the `gestion.views.manage()` view

menuhistory_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name
Name of the menu.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**MenuHistory**(**args*, ***kwargs*)
Stores MenuHistory related to `Menu`.

exception DoesNotExist

exception MultipleObjectsReturned

amount
Price of the purchase.

coopeman
Coopeman (:class:django.contrib.auth.models.User‘) who collected the money.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
Date of the purhcase.

get_next_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=True*, ***kwargs*)

get_previous_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

menu
gestion.models.Menu purchased.

menu_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

paymentMethod
Payment Method of the Menu purchased.

paymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

quantity
Client (`django.contrib.auth.models.User`).

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you’re doing before you use this method.

class gestion.models.**Pinte**(**args*, ***kwargs*)
Stores a physical pinte

exception DoesNotExist

exception MultipleObjectsReturned

current_owner
The current owner (`django.contrib.auth.models.User`).

current_owner_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_last_update_date(*, *field=<django.db.models.fields.DateTimeField: last_update_date>*, *is_next=True*, ***kwargs*)

get_previous_by_last_update_date(*, *field=<django.db.models.fields.DateTimeField: last_update_date>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_update_date
The last update date

objects = <django.db.models.manager.Manager object>

previous_owner
The previous owner (`django.contrib.auth.models.User`).

previous_owner_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**Product**(**args*, ***kwargs*)
Stores a product.

BOTTLE = 'BT'

D_PRESSION = 'DP'

exception DoesNotExist

FOOD = 'FO'

G_PRESSION = 'GP'

exception MultipleObjectsReturned

PANINI = 'PA'

P_PRESSION = 'PP'

SOFT = 'SO'

TYPEINPUT_CHOICES_CATEGORIE = (('PP', 'Pinte Pression'), ('DP', 'Demi Pression'), ('GP

adherentRequired
If True, only adherents will be able to buy this product

amount
The price of the product.

barcode
The barcode of the product.

category
The category of the product

consumption_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

consumptionhistory_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

deg
Degree of alcohol, if relevant

futd
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

futg
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

futp

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_category_display(**, field=<django.db.models.fields.CharField: category>*)

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active

If True, will be displayed on the *`gestion.views.manage()`* view.

menu_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

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

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name

The name of the product.

needQuantityButton

If True, a javascript quantity button will be displayed

objects = <django.db.models.manager.Manager object>

ranking

Get the first 25 users with `user_ranking()`

save_without_historical_record(**args, **kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

showingMultiplier

On the graphs on *`users.views.profile()`* view, the number of total consumptions is divised by the showingMultiplier

stockBar

Number of product at the bar.

stockHold

Number of product in the hold.

user_ranking(*pk*)
Return the user ranking for the product

volume
The volume, if relevant, of the product

class gestion.models.**Refund**(**args*, ***kwargs*)
Stores refunds.

exception DoesNotExist

exception MultipleObjectsReturned

amount
Amount of the refund.

coopeman
Coopeman (`django.contrib.auth.models.User`) who realized the refund.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Client (`django.contrib.auth.models.User`).

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
Date of the refund

get_next_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=True*, ***kwargs*)

get_previous_by_date(*, *field=<django.db.models.fields.DateTimeField: date>*, *is_next=False*, ***kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class gestion.models.**Reload**(**args*, ***kwargs*)
Stores reloads.

exception DoesNotExist

exception MultipleObjectsReturned

PaymentMethod
`Payment Method` of the reload.

PaymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

amount
Amount of the reload.

coopeman
Coopeman (`django.contrib.auth.models.User`) who collected the reload.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

customer
Client (`django.contrib.auth.models.User`).

customer_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

date
Date of the reload.

get_next_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=True, **kwargs*)

get_previous_by_date(**, field=<django.db.models.fields.DateTimeField: date>, is_next=False, **kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args, **kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

`gestion.models.`**isDemi**(*id*)

`gestion.models.`**isGalopin**(*id*)

`gestion.models.`**isPinte**(*id*)

2.2 Users app models

class `users.models.`**CotisationHistory**(**args, **kwargs*)
Stores cotisation histories, related to `preferences.models.Cotisation`.

exception DoesNotExist

exception MultipleObjectsReturned

amount
Price, in euros, of the cotisation.

coopeman
User (`django.contrib.auth.models.User`) who registered the cotisation.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

cotisation
`Cotisation` related.

cotisation_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration
Duration, in days, of the cotisation.

endDate
End date of the cotisation.

get_next_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=True, **kwargs*)

get_next_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=True, **kwargs*)

get_previous_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=False, **kwargs*)

get_previous_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=False, **kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

paymentDate
Date of the payment.

paymentMethod
`Payment method` used.

paymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

save_without_historical_record(**args, **kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

user
Client (`django.contrib.auth.models.User`).

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class users.models.**HistoricalCotisationHistory**(*id, amount, duration, paymentDate, endDate, user, paymentMethod, cotisation, coopeman, history_id, history_change_reason, history_date, history_user, history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

cotisation
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

cotisation_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

endDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_endDate(*, *field=<django.db.models.fields.DateTimeField: endDate>, is_next=True, **kwargs*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_next_by_paymentDate(*, *field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=True, **kwargs*)

get_previous_by_endDate(*, *field=<django.db.models.fields.DateTimeField: endDate>, is_next=False, **kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

get_previous_by_paymentDate(*, *field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *CotisationHistory*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

paymentDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

paymentMethod
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

paymentMethod_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class users.models.**HistoricalProfile**(*id*, *credit*, *debit*, *cotisationEnd*, *user*, *school*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

cotisationEnd
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

credit
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

debit
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *`Profile`*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

school
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

school_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOne-ToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class users.models.**HistoricalSchool**(*id*, *name*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *School*

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class users.models.**HistoricalWhiteListHistory**(*id*, *paymentDate*, *endDate*, *duration*, *user*, *coopeman*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

coopeman
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

endDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=True, **kwargs*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_next_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=True, **kwargs*)

get_previous_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=False, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

get_previous_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *WhiteListHistory*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

paymentDate
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class users.models.**Profile**(**args*, ***kwargs*)
Stores user profile.

exception DoesNotExist

exception MultipleObjectsReturned

alcohol
Computes ingerated alcohol.

balance
Computes client balance (`gestion.models.Profile.credit` - `gestion.models.Profile.debit`).

cotisationEnd
Date of end of cotisation for the client

credit
Amount of money, in euros, put on the account

debit
Amount of money, in euros, spent form the account

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_adherent
Test if a client is adherent.

nb_pintes
Return the number of *`Pinte(s)`* currently owned.

objects = <django.db.models.manager.Manager object>

positiveBalance()
Test if the client balance is positive or null.

rank
Computes the rank (by `gestion.models.Profile.debit`) of the client.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

school
School of the client

school_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user
Client (`django.contrib.auth.models.User`).

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `users.models.`**School**(**args*, ***kwargs*)
Stores school.

exception DoesNotExist

exception MultipleObjectsReturned

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
The name of the school

objects = <django.db.models.manager.Manager object>

profile_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class `users.models.`**WhiteListHistory**(**args*, ***kwargs*)
Stores whitelist history.

exception DoesNotExist

exception MultipleObjectsReturned

coopeman
User (`django.contrib.auth.models.User`) who registered the cotisation.

coopeman_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration
Duration, in days, of the whitelist

endDate
End date of the whitelist.

get_next_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=True, **kwargs*)

get_next_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=True, **kwargs*)

get_previous_by_endDate(**, field=<django.db.models.fields.DateTimeField: endDate>, is_next=False, **kwargs*)

get_previous_by_paymentDate(**, field=<django.db.models.fields.DateTimeField: paymentDate>, is_next=False, **kwargs*)

history = <simple_history.manager.HistoryManager object>

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

paymentDate
Date of the beginning of the whitelist.

save_without_historical_record(**args, **kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

user
Client (`django.contrib.auth.models.User`).

user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`users.models.`**create_user_profile**(*sender, instance, created, **kwargs*)
Create profile when user (`django.contrib.auth.models.User`) is created.

`users.models.`**save_user_profile**(*sender, instance, **kwargs*)
Save profile when user (`django.contrib.auth.models.User`) is saved.

`users.models.`**str_user**(*self*)
Rewrite str method for user (`django.contrib.auth.models.User`).

2.3 Preferences app models

class `preferences.models.`**Cotisation**(**args, **kwargs*)
Stores cotisations.

exception DoesNotExist

exception MultipleObjectsReturned

amount
Price of the cotisation.

cotisationhistory_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

duration

Duration (in days) of the cotisation

history = <simple_history.manager.HistoryManager object>

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

save_without_historical_record(**args*, ***kwargs*)

Save model without saving a historical record

Make sure you know what you're doing before you use this method.

class preferences.models.**GeneralPreferences**(**args*, ***kwargs*)

Stores a unique line of general preferences

exception DoesNotExist

exception MultipleObjectsReturned

active_message

Message displayed on the *inactive()*

automatic_logout_time

Duration after which the user is automatically disconnected if inactive

brewer

The name of the brewer

floating_buttons

If True, displays floating paymentButtons on the *manage()* view.

global_message

List of messages, separated by a carriage return. One will be chosen randomly at each request on displayed in the header

grocer

The name of the grocer

history = <simple_history.manager.HistoryManager object>

home_text

Text display on the home page

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active
If True, the site will be accessible. If False, all the requests are redirect to `inactive()`.

lost_pintes_allowed
If > 0, a user will be blocked if he has losted more pints than the value

menu
The file of the menu

objects = <django.db.models.manager.Manager object>

president
The name of the president

rules
The file of the internal rules

save_without_historical_record(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

secretary
The name of the secretary

statutes
The file of the statutes

treasurer
The name of the treasurer

use_pinte_monitoring
If True, alert will be displayed to allocate pints during order

vice_president
The name of the vice-president

class preferences.models.**HistoricalCotisation**(*id*, *amount*, *duration*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

amount
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

duration
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(*, *field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=True*, ***kwargs*)

get_previous_by_history_date(*, *field=<django.db.models.fields.DateTimeField: history_date>*, *is_next=False*, ***kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *Cotisation*

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class preferences.models.**HistoricalGeneralPreferences**(*id*, *is_active*, *active_message*, *global_message*, *president*, *vice_president*, *treasurer*, *secretary*, *brewer*, *grocer*, *use_pinte_monitoring*, *lost_pintes_allowed*, *floating_buttons*, *home_text*, *automatic_logout_time*, *statutes*, *rules*, *menu*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

active_message
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

automatic_logout_time
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

brewer
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

floating_buttons
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

global_message
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

grocer
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

home_text
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *GeneralPreferences*

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

lost_pintes_allowed
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

menu
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

president
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

rules
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

secretary
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

statutes
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

treasurer
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

use_pinte_monitoring
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

vice_president
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class preferences.models.**HistoricalPaymentMethod**(*id*, *name*, *is_active*, *is_usable_in_cotisation*, *is_usable_in_reload*, *affect_balance*, *icon*, *history_id*, *history_change_reason*, *history_date*, *history_user*, *history_type*)

exception DoesNotExist

exception MultipleObjectsReturned

affect_balance
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_history_type_display(**, field=<django.db.models.fields.CharField: history_type>*)

get_next_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=True, **kwargs*)

get_previous_by_history_date(**, field=<django.db.models.fields.DateTimeField: history_date>, is_next=False, **kwargs*)

history_change_reason
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_date
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_object

history_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

history_user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

history_user_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

icon
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instance

instance_type
alias of *PaymentMethod*

is_active
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_usable_in_cotisation
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_usable_in_reload
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

next_record
Get the next history record for the instance. *None* if last.

objects = <django.db.models.manager.Manager object>

prev_record
Get the previous history record for the instance. *None* if first.

revert_url()
URL for this change in the default admin site.

class preferences.models.**PaymentMethod**(**args*, ***kwargs*)
Stores payment methods.

exception DoesNotExist

exception MultipleObjectsReturned

affect_balance
If true, the PaymentMethod will decrease the user's balance when used.

consumptionhistory_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

cotisationhistory_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

history = <simple_history.manager.HistoryManager object>

icon

A font awesome icon (without the fa)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_active

If False, the PaymentMethod can't be use anywhere.

is_usable_in_cotisation

If true, the PaymentMethod can be used to pay cotisation.

is_usable_in_reload

If true, the PaymentMethod can be used to reload an user account.

menuhistory_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

name

The name of the PaymentMethod.

objects = <django.db.models.manager.Manager object>

reload_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

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

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`save_without_historical_record`(**args*, ***kwargs*)
Save model without saving a historical record

Make sure you know what you're doing before you use this method.

CHAPTER 3

Admin documentation

3.1 Gestion app admin

class gestion.admin.**ConsumptionAdmin**(*model*, *admin_site*)

The admin class for *Consumptions*.

list_display = ('customer', 'product', 'quantity')

media

ordering = ('-quantity',)

search_fields = ('customer', 'product')

class gestion.admin.**ConsumptionHistoryAdmin**(*model*, *admin_site*)

The admin class for *Consumption Histories*.

list_display = ('customer', 'product', 'quantity', 'paymentMethod', 'date', 'amount')

list_filter = ('paymentMethod',)

media

ordering = ('-date',)

search_fields = ('customer', 'product')

class gestion.admin.**KegAdmin**(*model*, *admin_site*)

The admin class for *Kegs*.

list_display = ('name', 'stockHold', 'capacity', 'is_active')

list_filter = ('capacity', 'is_active')

media

ordering = ('name',)

search_fields = ('name',)

class gestion.admin.**KegHistoryAdmin**(*model*, *admin_site*)

The admin class for *Keg Histories*.

list_display = ('keg', 'openingDate', 'closingDate', 'isCurrentKegHistory', 'quantitySo

list_filter = ('isCurrentKegHistory', 'keg')

media

ordering = ('-openingDate', 'quantitySold')

search_fields = ('keg',)

class gestion.admin.**MenuAdmin**(*model*, *admin_site*)

The admin class for *Menu*.

list_display = ('name', 'amount', 'is_active')

list_filter = ('is_active',)

media

ordering = ('name', 'amount')

search_fields = ('name',)

class gestion.admin.**MenuHistoryAdmin**(*model*, *admin_site*)

The admin class for *Menu Histories*.

list_display = ('customer', 'menu', 'paymentMethod', 'date', 'quantity', 'amount')

media

ordering = ('-date',)

search_fields = ('customer', 'menu')

class gestion.admin.**ProductAdmin**(*model*, *admin_site*)

The admin class for *Products*.

list_display = ('name', 'amount', 'is_active', 'category', 'adherentRequired', 'stockHo

list_filter = ('is_active', 'adherentRequired', 'category')

media

ordering = ('name', 'amount', 'stockHold', 'stockBar', 'deg')

search_fields = ('name',)

class gestion.admin.**RefundAdmin**(*model*, *admin_site*)

The admin class for *Refunds*.

list_display = ('customer', 'amount', 'date')

media

ordering = ('-date', 'amount', 'customer')

search_fields = ('customer',)

class gestion.admin.**ReloadAdmin**(*model*, *admin_site*)

The admin class for *Reloads*.

list_display = ('customer', 'amount', 'date', 'PaymentMethod')

list_filter = ('PaymentMethod',)

media

ordering = ('-date', 'amount', 'customer')

search_fields = ('customer',)

3.2 Users app admin

class `users.admin.`**BalanceFilter**(*request*, *params*, *model*, *model_admin*)

A filter which filters according to the sign of the balance

lookups(*request*, *model_admin*)

Must be overridden to return a list of tuples (value, verbose value)

parameter_name = 'solde'

queryset(*request*, *queryset*)

Return the filtered queryset.

title = 'Solde'

class `users.admin.`**CotisationHistoryAdmin**(*model*, *admin_site*)

The admin class for `Consumptions`.

list_display = ('user', 'amount', 'duration', 'paymentDate', 'endDate', 'paymentMethod

list_filter = ('paymentMethod',)

media

ordering = ('user', 'amount', 'duration', 'paymentDate', 'endDate')

search_fields = ('user',)

class `users.admin.`**ProfileAdmin**(*model*, *admin_site*)

The admin class for `Consumptions`.

list_display = ('user', 'credit', 'debit', 'balance', 'school', 'cotisationEnd', 'is_a

list_filter = ('school', <class 'users.admin.BalanceFilter'>)

media

ordering = ('user', '-credit', '-debit')

search_fields = ('user',)

class `users.admin.`**WhiteListHistoryAdmin**(*model*, *admin_site*)

The admin class for `Consumptions`.

list_display = ('user', 'paymentDate', 'endDate', 'duration')

media

ordering = ('user', 'duration', 'paymentDate', 'endDate')

search_fields = ('user',)

3.3 Preferences app admin

class `preferences.admin.`**CotisationAdmin**(*model*, *admin_site*)

The admin class for `Consumptions`.

list_display = ('__str__', 'amount', 'duration')

media

ordering = ('-duration', '-amount')

class preferences.admin.**GeneralPreferencesAdmin**(*model*, *admin_site*)
The admin class for *Consumptions*.

list_display = ('is_active', 'president', 'vice_president', 'treasurer', 'secretary',

media

class preferences.admin.**PaymentMethodAdmin**(*model*, *admin_site*)
The admin class for *Consumptions*.

list_display = ('name', 'is_active', 'is_usable_in_cotisation', 'is_usable_in_reload',

list_filter = ('is_active', 'is_usable_in_cotisation', 'is_usable_in_reload', 'affect_

media

ordering = ('name',)

search_fields = ('name',)

CHAPTER 4

Forms documentation

4.1 Gestion app forms

class gestion.forms.**GeneratReleveForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to generate a releve.

base_fields = {'begin': <django.forms.fields.DateTimeField object>, 'end': <django.fo

declared_fields = {'begin': <django.forms.fields.DateTimeField object>, 'end': <djan

media

class gestion.forms.**GestionForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form for the `manage()` view.

base_fields = {'client': <django.forms.models.ModelChoiceField object>, 'product': <

declared_fields = {'client': <django.forms.models.ModelChoiceField object>, 'product'

media

class gestion.forms.**KegForm**(**args*, ***kwargs*)

A form to create and edit a `Keg`.

class Meta

exclude = ('is_active',)

model
alias of *gestion.models.Keg*

widgets = {'amount': <class 'django.forms.widgets.TextInput'>}

base_fields = {'amount': <django.forms.fields.DecimalField object>, 'barcode': <djan

declared_fields = {}

media

class gestion.forms.**MenuForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

A form to create and edit a *Menu*.

class Meta

fields = '__all__'

model
alias of *gestion.models.Menu*

widgets = {'amount': <class 'django.forms.widgets.TextInput'>}

base_fields = {'amount': <django.forms.fields.DecimalField object>, 'articles': <dja

declared_fields = {}

media

class gestion.forms.**PinteForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to free *Pints*.

base_fields = {'begin': <django.forms.fields.IntegerField object>, 'end': <django.fo

declared_fields = {'begin': <django.forms.fields.IntegerField object>, 'end': <djang

media

class gestion.forms.**ProductForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

A form to create and edit a *Product*.

class Meta

fields = '__all__'

model
alias of *gestion.models.Product*

widgets = {'amount': <class 'django.forms.widgets.TextInput'>}

base_fields = {'adherentRequired': <django.forms.fields.BooleanField object>, 'amount

declared_fields = {}

media

class gestion.forms.**RefundForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

A form to create a `Refund`.

class Meta

fields = ('customer', 'amount')

model

alias of `gestion.models.Refund`

widgets = {'amount': <class 'django.forms.widgets.TextInput'>, 'customer': <dal_s

base_fields = {'amount': <django.forms.fields.DecimalField object>, 'customer': <dja

declared_fields = {}

media

class gestion.forms.**ReloadForm**(**args*, ***kwargs*)

A form to create a `Reload`.

class Meta

fields = ('customer', 'amount', 'PaymentMethod')

model

alias of `gestion.models.Reload`

widgets = {'amount': <class 'django.forms.widgets.TextInput'>, 'customer': <dal_s

base_fields = {'PaymentMethod': <django.forms.models.ModelChoiceField object>, 'amoun

declared_fields = {}

media

class gestion.forms.**SearchMenuForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to search a `Menu`.

base_fields = {'menu': <django.forms.models.ModelChoiceField object>}

declared_fields = {'menu': <django.forms.models.ModelChoiceField object>}

media

class gestion.forms.**SearchProductForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to search a `Product`.

base_fields = {'product': <django.forms.models.ModelChoiceField object>}

declared_fields = {'product': <django.forms.models.ModelChoiceField object>}

media

class gestion.forms.**SelectActiveKegForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to search an active `Keg`.

base_fields = {'keg': <django.forms.models.ModelChoiceField object>}

declared_fields = {'keg': <django.forms.models.ModelChoiceField object>}

media

class gestion.forms.**SelectPositiveKegForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

A form to search a `Keg` with a positive stockhold.

base_fields = {'keg': <django.forms.models.ModelChoiceField object>}

declared_fields = {'keg': <django.forms.models.ModelChoiceField object>}

media

4.2 Users app forms

class users.forms.**CreateGroupForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to create a new group (`django.contrib.auth.models.Group`).

class Meta

fields = ('name',)

model

alias of `django.contrib.auth.models.Group`

base_fields = {'name': <django.forms.fields.CharField object>}

declared_fields = {}

media

class users.forms.**CreateUserForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to create a new user (`django.contrib.auth.models.User`).

class Meta

fields = ('username', 'last_name', 'first_name', 'email')

model
alias of `django.contrib.auth.models.User`

base_fields = {'email': <django.forms.fields.EmailField object>, 'first_name': <djan

declared_fields = {'school': <django.forms.models.ModelChoiceField object>}

media

class `users.forms.`**EditGroupForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Form to edit a group (`django.contrib.auth.models.Group`).

class Meta

fields = '__all__'

model
alias of `django.contrib.auth.models.Group`

base_fields = {'name': <django.forms.fields.CharField object>, 'permissions': <djang

declared_fields = {}

media

class `users.forms.`**EditPasswordForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)
Form to change the password of a user (`django.contrib.auth.models.User`).

base_fields = {'password': <django.forms.fields.CharField object>, 'password1': <dja

clean_password2()
Verify if the two new passwords are identical

declared_fields = {'password': <django.forms.fields.CharField object>, 'password1':

media

class `users.forms.`**ExportForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)
Form to export list of users (`django.contrib.auth.models.User`) to csv file

FIELDS_CHOICES = (('username', "Nom d'utilisateur"), ('last_name', 'Nom'), ('first_nam

QUERY_TYPE_CHOICES = (('all', 'Tous les comptes'), ('all_active', 'Tous les comptes ac

base_fields = {'fields': <django.forms.fields.MultipleChoiceField object>, 'group':

declared_fields = {'fields': <django.forms.fields.MultipleChoiceField object>, 'group

media

class users.forms.**GroupsEditForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to edit a user's list of groups (`django.contrib.auth.models.User` and `django.contrib.auth.models.Group`).

class Meta

fields = ('groups',)

model

alias of `django.contrib.auth.models.User`

base_fields = {'groups': <django.forms.models.ModelMultipleChoiceField object>}

declared_fields = {}

media

class users.forms.**LoginForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form to log in.

base_fields = {'password': <django.forms.fields.CharField object>, 'username': <djan

declared_fields = {'password': <django.forms.fields.CharField object>, 'username': <

media

class users.forms.**SchoolForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to add and edit a `users.models.School`.

class Meta

fields = '__all__'

model

alias of `users.models.School`

base_fields = {'name': <django.forms.fields.CharField object>}

declared_fields = {}

media

class users.forms.**SelectNonAdminUserForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form to select a user from all non-staff users (`django.contrib.auth.models.User`).

base_fields = {'user': <django.forms.models.ModelChoiceField object>}

declared_fields = {'user': <django.forms.models.ModelChoiceField object>}

media

class users.forms.**SelectNonSuperUserForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form to select a user from all non-superuser users (`django.contrib.auth.models.User`).

base_fields = {'user': <django.forms.models.ModelChoiceField object>}

declared_fields = {'user': <django.forms.models.ModelChoiceField object>}

media

class users.forms.**SelectUserForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Form to select a user from all users (`django.contrib.auth.models.User`).

base_fields = {'user': <django.forms.models.ModelChoiceField object>}

declared_fields = {'user': <django.forms.models.ModelChoiceField object>}

media

class users.forms.**addCotisationHistoryForm**(**args*, ***kwargs*)

Form to add a `users.models.CotisationHistory` to user (`django.contrib.auth.models.User`).

class Meta

fields = ('cotisation', 'paymentMethod')

model

alias of `users.models.CotisationHistory`

base_fields = {'cotisation': <django.forms.models.ModelChoiceField object>, 'paymentM

declared_fields = {}

media

class users.forms.**addWhiteListHistoryForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to add a `users.models.WhiteListHistory` to user (`django.contrib.auth.models.User`).

class Meta

fields = ('duration',)

model

alias of `users.models.WhiteListHistory`

base_fields = {'duration': <django.forms.fields.IntegerField object>}

declared_fields = {}

media

4.3 Preferences app forms

class preferences.forms.**CotisationForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to add and edit `Cotisation`.

class Meta

fields = '__all__'

model

alias of `preferences.models.Cotisation`

base_fields = {'amount': <django.forms.fields.DecimalField object>, 'duration': <dja

declared_fields = {}

media

class preferences.forms.**GeneralPreferencesForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to edit the `GeneralPreferences`.

class Meta

fields = '__all__'

model

alias of `preferences.models.GeneralPreferences`

widgets = {'active_message': <django.forms.widgets.Textarea object>, 'brewer': <

base_fields = {'active_message': <django.forms.fields.CharField object>, 'automatic_l

declared_fields = {}

media

class preferences.forms.**PaymentMethodForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Form to add and edit `PaymentMethod`.

class Meta

fields = '__all__'

model
alias of *preferences.models.PaymentMethod*

base_fields = {'affect_balance': <django.forms.fields.BooleanField object>, 'icon':

declared_fields = {}

media

CHAPTER 5

Utils documentation

5.1 ACL

`coopeV3.acl.`**`acl_and`**`(`**perms*`)`
  Test if a user has all perms

`coopeV3.acl.`**`acl_or`**`(`**perms*`)`
  Test if a user has one of perms

`coopeV3.acl.`**`active_required`**`(`*view*`)`
  Test if the site is active (*`preferences.models.GeneralPreferences.is_active`*).

`coopeV3.acl.`**`admin_required`**`(`*view*`)`
  Test if the user is staff.

`coopeV3.acl.`**`self_or_has_perm`**`(`*pkName*, *perm*`)`
  Test if the user is the request user (pk) or has perm permission.

`coopeV3.acl.`**`superuser_required`**`(`*view*`)`
  Test if the user is superuser.

5.2 CoopeV3 templatetags

`coopeV3.templatetags.vip.`**`brewer`**`()`
  A tag which returns *`preferences.models.GeneralPreferences.brewer`*.

`coopeV3.templatetags.vip.`**`global_message`**`()`
  A tag which returns *`preferences.models.GeneralPreferences.global_message`*.

`coopeV3.templatetags.vip.`**`grocer`**`()`
  A tag which returns *`preferences.models.GeneralPreferences.grocer`*.

`coopeV3.templatetags.vip.`**`logout_time`**`()`
  A tag which returns *`preferences.models.GeneralPreferences.automatic_logout_time`*.

`coopeV3.templatetags.vip.`**`menu`**`()`
A tag which returns *`preferences.models.GeneralPreferences.menu`*.

`coopeV3.templatetags.vip.`**`president`**`()`
A tag which returns *`preferences.models.GeneralPreferences.president`*.

`coopeV3.templatetags.vip.`**`rules`**`()`
A tag which returns *`preferences.models.GeneralPreferences.rules`*.

`coopeV3.templatetags.vip.`**`secretary`**`()`
A tag which returns *`preferences.models.GeneralPreferences.secretary`*.

`coopeV3.templatetags.vip.`**`statutes`**`()`
A tag which returns *`preferences.models.GeneralPreferences.statutes`*.

`coopeV3.templatetags.vip.`**`treasurer`**`()`
A tag which returns *`preferences.models.GeneralPreferences.treasurer`*.

`coopeV3.templatetags.vip.`**`vice_president`**`()`
A tag which returns *`preferences.models.GeneralPreferences.vice_president`*.

5.3 Users templatetags

`users.templatetags.users_extra.`**`random_filter`**`(`*a*, *b*`)`

CHAPTER 6

Django_tex documentation

6.1 Core

django_tex.core.**compile_template_to_pdf**(*template_name*, *context*)
    Compile the source with *render_template_with_context()* and *run_tex()*.

django_tex.core.**render_template_with_context**(*template_name*, *context*)
    Render the template

django_tex.core.**run_tex**(*source*)
    Copy the source to temp dict and run latex.

6.2 Engine

class django_tex.engine.**TeXEngine**(*params*)

    app_dirname = 'templates'

6.3 Environment

django_tex.environment.**environment**(***options*)

6.4 Exceptions

exception django_tex.exceptions.**TexError**(*log*, *source*)

    get_message()

`django_tex.exceptions.`**`prettify_message`**`(`*message*`)`
Helper methods that removes consecutive whitespaces and newline characters

`django_tex.exceptions.`**`tokenizer`**`(`*code*`)`

6.5 Filters

`django_tex.filters.`**`do_linebreaks`**`(`*value*`)`

6.6 Models

class `django_tex.models.`**`TeXTemplateFile`**`(`**args*, ***kwargs*`)`

class Meta

abstract = False

name
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

title
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`django_tex.models.`**`validate_template_path`**`(`*name*`)`

6.7 Views

class `django_tex.views.`**`PDFResponse`**`(`*content*, *filename=None*`)`

`django_tex.views.`**`render_to_pdf`**`(`*request*, *template_name*, *context=None*, *filename=None*`)`

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index