
CoopeV3 Documentation

Release 3.4.0

Yoann Pietri

Feb 28, 2019

Contents:

1	Views documentation	1
1.1	Gestion app views	1
1.2	Users app views	4
1.3	Preferences app views	6
1.4	coopeV3 app views	7
2	Models documentation	9
2.1	Gestion app models	9
2.2	Users app models	38
2.3	Preferences app models	49
3	Admin documentation	59
3.1	Gestion app admin	59
3.2	Users app admin	61
3.3	Preferences app admin	61
4	Forms documentation	63
4.1	Gestion app forms	63
4.2	Users app forms	66
4.3	Preferences app forms	70
5	Utils documentation	73
5.1	ACL	73
5.2	CoopeV3 templatetags	73
5.3	Users templatetags	74
6	Django_tex documentation	75
6.1	Core	75
6.2	Engine	75
6.3	Environment	75
6.4	Exceptions	75
6.5	Filters	76
6.6	Models	76
6.7	Views	76
7	Indices and tables	77

Python Module Index

79

Index

81

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)
    Autocomplete 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 the *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*.

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 purchase.

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, his-  
tory_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: his-  
tory_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: his-
tory_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 ForwardOneToOneDescriptor 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 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.

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 ForwardOneToOneDescriptor 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:

```
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 purchase.

```
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', 'Grosse Pression'))

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 divided 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: payment-
                             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>

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, co-
                                              tisation, coopeman, history_id, his-
                                              tory_change_reason, history_date, his-
                                              tory_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 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 *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 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.

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 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 *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 ForwardOneToOneDescriptor 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 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.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: payment-
                              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>
```

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.

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', 'quantityS
    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', 'stockH
    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',)

4.1 Gestion app forms

```
class gestion.forms.GenerateReleveForm(data=None, files=None, auto_id='id_%s', pre-
    fix=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.f
```

```
declared_fields = {'begin': <django.forms.fields.DateTimeField object>, 'end': <djan
```

```
media
```

```
class gestion.forms.GestionForm(data=None, files=None, auto_id='id_%s', pre-
    fix=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, ini-
    tial=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, ini-
    tial=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', pre-
    fix=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>, 'amount
```

```
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', pre-  
fix=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', pre-  
fix=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', pre-  
fix=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', pre-
                                fix=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', pre-
                                    fix=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, ini-
                              tial=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_name
    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': <django.forms.fields.CharField object>}
```

```
    declared_fields = {'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}
```

```
    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', pre-  
fix=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, ren-  
derer=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': <c  
    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
```


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)`

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`

c

coopeV3.acl, 73
coopeV3.templatetags.vip, 73
coopeV3.views, 7

d

django_tex.core, 75
django_tex.engine, 75
django_tex.environment, 75
django_tex.exceptions, 75
django_tex.filters, 76
django_tex.models, 76
django_tex.views, 76

g

gestion.admin, 59
gestion.forms, 63
gestion.models, 9
gestion.views, 1

p

preferences.admin, 61
preferences.forms, 70
preferences.models, 49
preferences.views, 6

u

users.admin, 61
users.forms, 66
users.models, 38
users.templatetags.users_extra, 74
users.views, 4

A

- abstract (*django_tex.models.TeXTemplateFile.Meta* attribute), 76
 acl_and() (*in module coopeV3.acl*), 73
 acl_or() (*in module coopeV3.acl*), 73
 active_message (*preferences.models.GeneralPreferences* attribute), 50
 active_message (*preferences.models.HistoricalGeneralPreferences* attribute), 53
 active_required() (*in module coopeV3.acl*), 73
 ActiveProductsAutocomplete (*class in gestion.views*), 1
 ActiveUsersAutocomplete (*class in users.views*), 4
 add_pintes() (*in module gestion.views*), 2
 addAdmin() (*in module users.views*), 4
 addCotisation() (*in module preferences.views*), 6
 addCotisationHistory() (*in module users.views*), 4
 addCotisationHistoryForm (*class in users.forms*), 69
 addCotisationHistoryForm.Meta (*class in users.forms*), 69
 addKeg() (*in module gestion.views*), 1
 addMenu() (*in module gestion.views*), 1
 addPaymentMethod() (*in module preferences.views*), 6
 addProduct() (*in module gestion.views*), 1
 addSuperuser() (*in module users.views*), 4
 addWhiteListHistory() (*in module users.views*), 4
 addWhiteListHistoryForm (*class in users.forms*), 69
 addWhiteListHistoryForm.Meta (*class in users.forms*), 69
 adherent_required (*gestion.models.Menu* attribute), 32
 AdherentAutocomplete (*class in users.views*), 4
 adherentRequired (*gestion.models.HistoricalProduct* attribute), 24
 adherentRequired (*gestion.models.Product* attribute), 34
 admin_required() (*in module coopeV3.acl*), 73
 adminsIndex() (*in module users.views*), 4
 affect_balance (*preferences.models.HistoricalPaymentMethod* attribute), 55
 affect_balance (*preferences.models.PaymentMethod* attribute), 56
 alcohol (*users.models.Profile* attribute), 47
 all_consumptions() (*in module users.views*), 4
 all_menus() (*in module users.views*), 4
 allocate() (*in module gestion.views*), 2
 allReloads() (*in module users.views*), 4
 AllUsersAutocomplete (*class in users.views*), 4
 amount (*gestion.models.ConsumptionHistory* attribute), 10
 amount (*gestion.models.HistoricalConsumptionHistory* attribute), 12
 amount (*gestion.models.HistoricalKeg* attribute), 15
 amount (*gestion.models.HistoricalMenu* attribute), 19
 amount (*gestion.models.HistoricalMenuHistory* attribute), 20
 amount (*gestion.models.HistoricalProduct* attribute), 24
 amount (*gestion.models.HistoricalRefund* attribute), 26
 amount (*gestion.models.HistoricalReload* attribute), 28
 amount (*gestion.models.Keg* attribute), 30
 amount (*gestion.models.Menu* attribute), 32
 amount (*gestion.models.MenuHistory* attribute), 32
 amount (*gestion.models.Product* attribute), 34
 amount (*gestion.models.Refund* attribute), 37
 amount (*gestion.models.Reload* attribute), 37
 amount (*preferences.models.Cotisation* attribute), 49
 amount (*preferences.models.HistoricalCotisation* attribute), 51

amount (*users.models.CotisationHistory* attribute), 38
 amount (*users.models.HistoricalCotisationHistory* attribute), 39
 amountSold (*gestion.models.HistoricalKegHistory* attribute), 17
 amountSold (*gestion.models.KegHistory* attribute), 31
 app_dirname (*django_tex.engine.TeXEngine* attribute), 75
 articles (*gestion.models.Menu* attribute), 32
 automatic_logout_time (*preferences.models.GeneralPreferences* attribute), 50
 automatic_logout_time (*preferences.models.HistoricalGeneralPreferences* attribute), 53

B

balance (*users.models.Profile* attribute), 47
 BalanceFilter (class in *users.admin*), 61
 barcode (*gestion.models.HistoricalKeg* attribute), 15
 barcode (*gestion.models.HistoricalMenu* attribute), 19
 barcode (*gestion.models.HistoricalProduct* attribute), 24
 barcode (*gestion.models.Keg* attribute), 30
 barcode (*gestion.models.Menu* attribute), 32
 barcode (*gestion.models.Product* attribute), 35
 base_fields (*gestion.forms.GenerateReleveForm* attribute), 63
 base_fields (*gestion.forms.GestionForm* attribute), 63
 base_fields (*gestion.forms.KegForm* attribute), 64
 base_fields (*gestion.forms.MenuForm* attribute), 64
 base_fields (*gestion.forms.PinteForm* attribute), 64
 base_fields (*gestion.forms.ProductForm* attribute), 64
 base_fields (*gestion.forms.RefundForm* attribute), 65
 base_fields (*gestion.forms.ReloadForm* attribute), 65
 base_fields (*gestion.forms.SearchMenuForm* attribute), 65
 base_fields (*gestion.forms.SearchProductForm* attribute), 65
 base_fields (*gestion.forms.SelectActiveKegForm* attribute), 66
 base_fields (*gestion.forms.SelectPositiveKegForm* attribute), 66
 base_fields (*preferences.forms.CotisationForm* attribute), 70
 base_fields (*preferences.forms.GeneralPreferencesForm* attribute), 70
 base_fields (*preferences.forms.PaymentMethodForm* attribute),

71
 base_fields (*users.forms.addCotisationHistoryForm* attribute), 69
 base_fields (*users.forms.addWhiteListHistoryForm* attribute), 69
 base_fields (*users.forms.CreateGroupForm* attribute), 66
 base_fields (*users.forms.CreateUserForm* attribute), 67
 base_fields (*users.forms.EditGroupForm* attribute), 67
 base_fields (*users.forms.EditPasswordForm* attribute), 67
 base_fields (*users.forms.ExportForm* attribute), 67
 base_fields (*users.forms.GroupsEditForm* attribute), 68
 base_fields (*users.forms.LoginForm* attribute), 68
 base_fields (*users.forms.SchoolForm* attribute), 68
 base_fields (*users.forms.SelectNonAdminUserForm* attribute), 68
 base_fields (*users.forms.SelectNonSuperUserForm* attribute), 69
 base_fields (*users.forms.SelectUserForm* attribute), 69
 BOTTLE (*gestion.models.Product* attribute), 34
 brewer (*preferences.models.GeneralPreferences* attribute), 50
 brewer (*preferences.models.HistoricalGeneralPreferences* attribute), 53
 brewer () (in module *coopeV3.templatetags.vip*), 73

C

cancel_consumption () (in module *gestion.views*), 2
 cancel_menu () (in module *gestion.views*), 2
 cancel_reload () (in module *gestion.views*), 2
 capacity (*gestion.models.HistoricalKeg* attribute), 15
 capacity (*gestion.models.Keg* attribute), 30
 category (*gestion.models.HistoricalProduct* attribute), 24
 category (*gestion.models.Product* attribute), 35
 clean_password2 () (*users.forms.EditPasswordForm* method), 67
 closeDirectKeg () (in module *gestion.views*), 2
 closeKeg () (in module *gestion.views*), 2
 closingDate (*gestion.models.HistoricalKegHistory* attribute), 17
 closingDate (*gestion.models.KegHistory* attribute), 31
 compile_template_to_pdf () (in module *django_tex.core*), 75
 Consumption (class in *gestion.models*), 9
 Consumption.DoesNotExist, 9

Consumption.MultipleObjectsReturned, 9
consumption_set (*gestion.models.Product attribute*), 35
ConsumptionAdmin (*class in gestion.admin*), 59
ConsumptionHistory (*class in gestion.models*), 10
ConsumptionHistory.DoesNotExist, 10
ConsumptionHistory.MultipleObjectsReturned, 10
consumptionhistory_set (*gestion.models.Product attribute*), 35
consumptionhistory_set (*preferences.models.PaymentMethod attribute*), 56
ConsumptionHistoryAdmin (*class in gestion.admin*), 59
coope_runner () (*in module coopeV3.views*), 7
coopeman (*gestion.models.ConsumptionHistory attribute*), 10
coopeman (*gestion.models.HistoricalConsumptionHistory attribute*), 12
coopeman (*gestion.models.HistoricalMenuHistory attribute*), 20
coopeman (*gestion.models.HistoricalRefund attribute*), 26
coopeman (*gestion.models.HistoricalReload attribute*), 28
coopeman (*gestion.models.MenuHistory attribute*), 32
coopeman (*gestion.models.Refund attribute*), 37
coopeman (*gestion.models.Reload attribute*), 38
coopeman (*users.models.CotisationHistory attribute*), 38
coopeman (*users.models.HistoricalCotisationHistory attribute*), 40
coopeman (*users.models.HistoricalWhiteListHistory attribute*), 45
coopeman (*users.models.WhiteListHistory attribute*), 48
coopeman_id (*gestion.models.ConsumptionHistory attribute*), 10
coopeman_id (*gestion.models.HistoricalConsumptionHistory attribute*), 13
coopeman_id (*gestion.models.HistoricalMenuHistory attribute*), 20
coopeman_id (*gestion.models.HistoricalRefund attribute*), 26
coopeman_id (*gestion.models.HistoricalReload attribute*), 28
coopeman_id (*gestion.models.MenuHistory attribute*), 33
coopeman_id (*gestion.models.Refund attribute*), 37
coopeman_id (*gestion.models.Reload attribute*), 38
coopeman_id (*users.models.CotisationHistory attribute*), 38
coopeman_id (*users.models.HistoricalCotisationHistory attribute*), 40
coopeman_id (*users.models.HistoricalWhiteListHistory attribute*), 45
coopeman_id (*users.models.WhiteListHistory attribute*), 48
coopeV3.acl (*module*), 73
coopeV3.templatetags.vip (*module*), 73
coopeV3.views (*module*), 7
Cotisation (*class in preferences.models*), 49
cotisation (*users.models.CotisationHistory attribute*), 38
cotisation (*users.models.HistoricalCotisationHistory attribute*), 40
Cotisation.DoesNotExist, 49
Cotisation.MultipleObjectsReturned, 49
cotisation_id (*users.models.CotisationHistory attribute*), 39
cotisation_id (*users.models.HistoricalCotisationHistory attribute*), 40
CotisationAdmin (*class in preferences.admin*), 61
cotisationEnd (*users.models.HistoricalProfile attribute*), 42
cotisationEnd (*users.models.Profile attribute*), 47
CotisationForm (*class in preferences.forms*), 70
CotisationForm.Meta (*class in preferences.forms*), 70
CotisationHistory (*class in users.models*), 38
CotisationHistory.DoesNotExist, 38
CotisationHistory.MultipleObjectsReturned, 38
cotisationhistory_set (*preferences.models.Cotisation attribute*), 49
cotisationhistory_set (*preferences.models.PaymentMethod attribute*), 57
CotisationHistoryAdmin (*class in users.admin*), 61
cotisationsIndex () (*in module preferences.views*), 6
create_user_profile () (*in module users.models*), 49
createGroup () (*in module users.views*), 5
CreateGroupForm (*class in users.forms*), 66
CreateGroupForm.Meta (*class in users.forms*), 66
createSchool () (*in module users.views*), 5
createUser () (*in module users.views*), 5
CreateUserForm (*class in users.forms*), 66
CreateUserForm.Meta (*class in users.forms*), 66
credit (*users.models.HistoricalProfile attribute*), 42
credit (*users.models.Profile attribute*), 47
current_owner (*gestion.models.HistoricalPinte attribute*), 22
current_owner (*gestion.models.Pinte attribute*), 34
current_owner_id (*gestion.models.HistoricalPinte*

attribute), 23
 current_owner_id (*gestion.models.Pinte attribute*), 34
 customer (*gestion.models.Consumption attribute*), 9
 customer (*gestion.models.ConsumptionHistory attribute*), 10
 customer (*gestion.models.HistoricalConsumption attribute*), 11
 customer (*gestion.models.HistoricalConsumptionHistory attribute*), 13
 customer (*gestion.models.HistoricalMenuHistory attribute*), 20
 customer (*gestion.models.HistoricalRefund attribute*), 26
 customer (*gestion.models.HistoricalReload attribute*), 28
 customer (*gestion.models.MenuHistory attribute*), 33
 customer (*gestion.models.Refund attribute*), 37
 customer (*gestion.models.Reload attribute*), 38
 customer_id (*gestion.models.Consumption attribute*), 9
 customer_id (*gestion.models.ConsumptionHistory attribute*), 10
 customer_id (*gestion.models.HistoricalConsumption attribute*), 11
 customer_id (*gestion.models.HistoricalConsumptionHistory attribute*), 13
 customer_id (*gestion.models.HistoricalMenuHistory attribute*), 20
 customer_id (*gestion.models.HistoricalRefund attribute*), 26
 customer_id (*gestion.models.HistoricalReload attribute*), 29
 customer_id (*gestion.models.MenuHistory attribute*), 33
 customer_id (*gestion.models.Refund attribute*), 37
 customer_id (*gestion.models.Reload attribute*), 38

D

D_PRESSION (*gestion.models.Product attribute*), 34
 date (*gestion.models.ConsumptionHistory attribute*), 10
 date (*gestion.models.HistoricalConsumptionHistory attribute*), 13
 date (*gestion.models.HistoricalMenuHistory attribute*), 21
 date (*gestion.models.HistoricalRefund attribute*), 27
 date (*gestion.models.HistoricalReload attribute*), 29
 date (*gestion.models.MenuHistory attribute*), 33
 date (*gestion.models.Refund attribute*), 37
 date (*gestion.models.Reload attribute*), 38
 debit (*users.models.HistoricalProfile attribute*), 42
 debit (*users.models.Profile attribute*), 47
 declared_fields (*gestion.forms.GenerateReleveForm attribute*), 63
 declared_fields (*gestion.forms.GestionForm attribute*), 63
 declared_fields (*gestion.forms.KegForm attribute*), 64
 declared_fields (*gestion.forms.MenuForm attribute*), 64
 declared_fields (*gestion.forms.PinteForm attribute*), 64
 declared_fields (*gestion.forms.ProductForm attribute*), 64
 declared_fields (*gestion.forms.RefundForm attribute*), 65
 declared_fields (*gestion.forms.ReloadForm attribute*), 65
 declared_fields (*gestion.forms.SearchMenuForm attribute*), 65
 declared_fields (*gestion.forms.SearchProductForm attribute*), 65
 declared_fields (*gestion.forms.SelectActiveKegForm attribute*), 66
 declared_fields (*gestion.forms.SelectPositiveKegForm attribute*), 66
 declared_fields (*preferences.forms.CotisationForm attribute*), 70
 declared_fields (*preferences.forms.GeneralPreferencesForm attribute*), 70
 declared_fields (*preferences.forms.PaymentMethodForm attribute*), 71
 declared_fields (*users.forms.addCotisationHistoryForm attribute*), 69
 declared_fields (*users.forms.addWhiteListHistoryForm attribute*), 70
 declared_fields (*users.forms.CreateGroupForm attribute*), 66
 declared_fields (*users.forms.CreateUserForm attribute*), 67
 declared_fields (*users.forms.EditGroupForm attribute*), 67
 declared_fields (*users.forms.EditPasswordForm attribute*), 67
 declared_fields (*users.forms.ExportForm attribute*), 67
 declared_fields (*users.forms.GroupsEditForm attribute*), 68
 declared_fields (*users.forms.LoginForm attribute*), 68
 declared_fields (*users.forms.SchoolForm attribute*), 68

- declared_fields (*users.forms.SelectNonAdminUserForm* attribute), 68
- declared_fields (*users.forms.SelectNonSuperUserForm* attribute), 69
- declared_fields (*users.forms.SelectUserForm* attribute), 69
- deg (*gestion.models.HistoricalProduct* attribute), 24
- deg (*gestion.models.Product* attribute), 35
- deleteCotisation() (in module *preferences.views*), 6
- deleteCotisationHistory() (in module *users.views*), 5
- deleteGroup() (in module *users.views*), 5
- deletePaymentMethod() (in module *preferences.views*), 7
- deleteSchool() (in module *users.views*), 5
- demi (*gestion.models.HistoricalKeg* attribute), 15
- demi (*gestion.models.Keg* attribute), 30
- demi_id (*gestion.models.HistoricalKeg* attribute), 15
- demi_id (*gestion.models.Keg* attribute), 30
- django_tex.core (module), 75
- django_tex.engine (module), 75
- django_tex.environment (module), 75
- django_tex.exceptions (module), 75
- django_tex.filters (module), 76
- django_tex.models (module), 76
- django_tex.views (module), 76
- do_linebreaks() (in module *django_tex.filters*), 76
- duration (*preferences.models.Cotisation* attribute), 50
- duration (*preferences.models.HistoricalCotisation* attribute), 51
- duration (*users.models.CotisationHistory* attribute), 39
- duration (*users.models.HistoricalCotisationHistory* attribute), 40
- duration (*users.models.HistoricalWhiteListHistory* attribute), 45
- duration (*users.models.WhiteListHistory* attribute), 48
- ## E
- edit_menu() (in module *gestion.views*), 2
- editCotisation() (in module *preferences.views*), 7
- editGroup() (in module *users.views*), 5
- EditGroupForm (class in *users.forms*), 67
- EditGroupForm.Meta (class in *users.forms*), 67
- editGroups() (in module *users.views*), 5
- editKeg() (in module *gestion.views*), 2
- editPassword() (in module *users.views*), 5
- EditPasswordForm (class in *users.forms*), 67
- editPaymentMethod() (in module *preferences.views*), 7
- editProduct() (in module *gestion.views*), 2
- editSchool() (in module *users.views*), 5
- editUser() (in module *users.views*), 5
- endDate (*users.models.CotisationHistory* attribute), 39
- endDate (*users.models.HistoricalCotisationHistory* attribute), 40
- endDate (*users.models.HistoricalWhiteListHistory* attribute), 45
- endDate (*users.models.WhiteListHistory* attribute), 49
- environment() (in module *django_tex.environment*), 75
- exclude (*gestion.forms.KegForm.Meta* attribute), 63
- export_csv() (in module *users.views*), 5
- ExportForm (class in *users.forms*), 67
- ## F
- fields (*gestion.forms.MenuForm.Meta* attribute), 64
- fields (*gestion.forms.ProductForm.Meta* attribute), 64
- fields (*gestion.forms.RefundForm.Meta* attribute), 65
- fields (*gestion.forms.ReloadForm.Meta* attribute), 65
- fields (*preferences.forms.CotisationForm.Meta* attribute), 70
- fields (*preferences.forms.GeneralPreferencesForm.Meta* attribute), 70
- fields (*preferences.forms.PaymentMethodForm.Meta* attribute), 71
- fields (*users.forms.addCotisationHistoryForm.Meta* attribute), 69
- fields (*users.forms.addWhiteListHistoryForm.Meta* attribute), 69
- fields (*users.forms.CreateGroupForm.Meta* attribute), 66
- fields (*users.forms.CreateUserForm.Meta* attribute), 67
- fields (*users.forms.EditGroupForm.Meta* attribute), 67
- fields (*users.forms.GroupsEditForm.Meta* attribute), 68
- fields (*users.forms.SchoolForm.Meta* attribute), 68
- FIELDS_CHOICES (*users.forms.ExportForm* attribute), 67
- floating_buttons (*preferences.models.GeneralPreferences* attribute), 50
- floating_buttons (*preferences.models.HistoricalGeneralPreferences* attribute), 53
- FOOD (*gestion.models.Product* attribute), 34
- futd (*gestion.models.Product* attribute), 35
- futg (*gestion.models.Product* attribute), 35
- futp (*gestion.models.Product* attribute), 35
- ## G
- G_PRESSION (*gestion.models.Product* attribute), 34
- galopin (*gestion.models.HistoricalKeg* attribute), 15
- galopin (*gestion.models.Keg* attribute), 30

- galopin_id (*gestion.models.HistoricalKeg* attribute), 15
- galopin_id (*gestion.models.Keg* attribute), 30
- gen_releve () (*in module gestion.views*), 2
- gen_user_infos () (*in module users.views*), 5
- GeneralPreferences (*class in preferences.models*), 50
- generalPreferences () (*in module preferences.views*), 7
- GeneralPreferences.DoesNotExist, 50
- GeneralPreferences.MultipleObjectsReturned, 50
- GeneralPreferencesAdmin (*class in preferences.admin*), 62
- GeneralPreferencesForm (*class in preferences.forms*), 70
- GeneralPreferencesForm.Meta (*class in preferences.forms*), 70
- GenerateReleveForm (*class in gestion.forms*), 63
- gestion.admin (*module*), 59
- gestion.forms (*module*), 63
- gestion.models (*module*), 9
- gestion.views (*module*), 1
- GestionForm (*class in gestion.forms*), 63
- get_category_display () (*gestion.models.HistoricalProduct* method), 24
- get_category_display () (*gestion.models.Product* method), 36
- get_config () (*in module preferences.views*), 7
- get_cotisation () (*in module preferences.views*), 7
- get_history_type_display () (*gestion.models.HistoricalConsumption* method), 11
- get_history_type_display () (*gestion.models.HistoricalConsumptionHistory* method), 13
- get_history_type_display () (*gestion.models.HistoricalKeg* method), 15
- get_history_type_display () (*gestion.models.HistoricalKegHistory* method), 17
- get_history_type_display () (*gestion.models.HistoricalMenu* method), 19
- get_history_type_display () (*gestion.models.HistoricalMenuHistory* method), 21
- get_history_type_display () (*gestion.models.HistoricalPinte* method), 23
- get_history_type_display () (*gestion.models.HistoricalProduct* method), 24
- get_history_type_display () (*gestion.models.HistoricalRefund* method), 27
- get_history_type_display () (*gestion.models.HistoricalReload* method), 29
- get_history_type_display () (*preferences.models.HistoricalCotisation* method), 51
- get_history_type_display () (*preferences.models.HistoricalGeneralPreferences* method), 53
- get_history_type_display () (*preferences.models.HistoricalPaymentMethod* method), 55
- get_history_type_display () (*users.models.HistoricalCotisationHistory* method), 40
- get_history_type_display () (*users.models.HistoricalProfile* method), 42
- get_history_type_display () (*users.models.HistoricalSchool* method), 44
- get_history_type_display () (*users.models.HistoricalWhiteListHistory* method), 45
- get_menu () (*in module gestion.views*), 2
- get_message () (*django_text.exceptions.TextError* method), 75
- get_next_by_date () (*gestion.models.ConsumptionHistory* method), 10
- get_next_by_date () (*gestion.models.HistoricalConsumptionHistory* method), 13
- get_next_by_date () (*gestion.models.HistoricalMenuHistory* method), 21
- get_next_by_date () (*gestion.models.HistoricalRefund* method), 27
- get_next_by_date () (*gestion.models.HistoricalReload* method), 29
- get_next_by_date () (*gestion.models.MenuHistory* method), 33
- get_next_by_date () (*gestion.models.Refund* method), 37
- get_next_by_date () (*gestion.models.Reload* method), 38
- get_next_by_endDate () (*users.models.CotisationHistory* method), 39
- get_next_by_endDate () (*users.models.HistoricalCotisationHistory* method), 40
- get_next_by_endDate () (*users.models.HistoricalWhiteListHistory* method), 45

`get_next_by_endDate()` (*users.models.WhiteListHistory* method), 49
`get_next_by_history_date()` (*gestion.models.HistoricalConsumption* method), 11
`get_next_by_history_date()` (*gestion.models.HistoricalConsumptionHistory* method), 13
`get_next_by_history_date()` (*gestion.models.HistoricalKeg* method), 15
`get_next_by_history_date()` (*gestion.models.HistoricalKegHistory* method), 17
`get_next_by_history_date()` (*gestion.models.HistoricalMenu* method), 19
`get_next_by_history_date()` (*gestion.models.HistoricalMenuHistory* method), 21
`get_next_by_history_date()` (*gestion.models.HistoricalPinte* method), 23
`get_next_by_history_date()` (*gestion.models.HistoricalProduct* method), 25
`get_next_by_history_date()` (*gestion.models.HistoricalRefund* method), 27
`get_next_by_history_date()` (*gestion.models.HistoricalReload* method), 29
`get_next_by_history_date()` (*preferences.models.HistoricalCotisation* method), 51
`get_next_by_history_date()` (*preferences.models.HistoricalGeneralPreferences* method), 53
`get_next_by_history_date()` (*preferences.models.HistoricalPaymentMethod* method), 55
`get_next_by_history_date()` (*users.models.HistoricalCotisationHistory* method), 40
`get_next_by_history_date()` (*users.models.HistoricalProfile* method), 42
`get_next_by_history_date()` (*users.models.HistoricalSchool* method), 44
`get_next_by_history_date()` (*users.models.HistoricalWhiteListHistory* method), 45
`get_next_by_last_update_date()` (*gestion.models.HistoricalPinte* method), 23
`get_next_by_last_update_date()` (*gestion.models.Pinte* method), 34
`get_next_by_openingDate()` (*gestion.models.HistoricalKegHistory* method), 17
`get_next_by_openingDate()` (*gestion.models.KegHistory* method), 31
`get_next_by_paymentDate()` (*users.models.CotisationHistory* method), 39
`get_next_by_paymentDate()` (*users.models.HistoricalCotisationHistory* method), 40
`get_next_by_paymentDate()` (*users.models.HistoricalWhiteListHistory* method), 45
`get_next_by_paymentDate()` (*users.models.WhiteListHistory* method), 49
`get_previous_by_date()` (*gestion.models.ConsumptionHistory* method), 10
`get_previous_by_date()` (*gestion.models.HistoricalConsumptionHistory* method), 13
`get_previous_by_date()` (*gestion.models.HistoricalMenuHistory* method), 21
`get_previous_by_date()` (*gestion.models.HistoricalRefund* method), 27
`get_previous_by_date()` (*gestion.models.HistoricalReload* method), 29
`get_previous_by_date()` (*gestion.models.MenuHistory* method), 33
`get_previous_by_date()` (*gestion.models.Refund* method), 37
`get_previous_by_date()` (*gestion.models.Reload* method), 38
`get_previous_by_endDate()` (*users.models.CotisationHistory* method), 39
`get_previous_by_endDate()` (*users.models.HistoricalCotisationHistory* method), 40
`get_previous_by_endDate()` (*users.models.HistoricalWhiteListHistory* method), 45
`get_previous_by_endDate()` (*users.models.WhiteListHistory* method), 49
`get_previous_by_history_date()` (*gestion.models.HistoricalConsumption* method), 11
`get_previous_by_history_date()` (*gestion.models.HistoricalConsumptionHistory* method), 13
`get_previous_by_history_date()` (*ges-*

tion.models.HistoricalKeg method), 15
get_previous_by_history_date() (*gestion.models.HistoricalKegHistory* method), 17
get_previous_by_history_date() (*gestion.models.HistoricalMenu* method), 19
get_previous_by_history_date() (*gestion.models.HistoricalMenuHistory* method), 21
get_previous_by_history_date() (*gestion.models.HistoricalPinte* method), 23
get_previous_by_history_date() (*gestion.models.HistoricalProduct* method), 25
get_previous_by_history_date() (*gestion.models.HistoricalRefund* method), 27
get_previous_by_history_date() (*gestion.models.HistoricalReload* method), 29
get_previous_by_history_date() (*preferences.models.HistoricalCotisation* method), 51
get_previous_by_history_date() (*preferences.models.HistoricalGeneralPreferences* method), 53
get_previous_by_history_date() (*preferences.models.HistoricalPaymentMethod* method), 55
get_previous_by_history_date() (*users.models.HistoricalCotisationHistory* method), 40
get_previous_by_history_date() (*users.models.HistoricalProfile* method), 42
get_previous_by_history_date() (*users.models.HistoricalSchool* method), 44
get_previous_by_history_date() (*users.models.HistoricalWhiteListHistory* method), 46
get_previous_by_last_update_date() (*gestion.models.HistoricalPinte* method), 23
get_previous_by_last_update_date() (*gestion.models.Pinte* method), 34
get_previous_by_openingDate() (*gestion.models.HistoricalKegHistory* method), 17
get_previous_by_openingDate() (*gestion.models.KegHistory* method), 31
get_previous_by_paymentDate() (*users.models.CotisationHistory* method), 39
get_previous_by_paymentDate() (*users.models.HistoricalCotisationHistory* method), 40
get_previous_by_paymentDate() (*users.models.HistoricalWhiteListHistory* method), 46
get_previous_by_paymentDate() (*users.models.WhiteListHistory* method), 49
get_queryset() (*gestion.views.ActiveProductsAutocomplete* method), 1
get_queryset() (*gestion.views.KegActiveAutocomplete* method), 1
get_queryset() (*gestion.views.KegPositiveAutocomplete* method), 1
get_queryset() (*gestion.views.MenusAutocomplete* method), 1
get_queryset() (*gestion.views.ProductsAutocomplete* method), 1
get_queryset() (*users.views.ActiveUsersAutocomplete* method), 4
get_queryset() (*users.views.AdherentAutocomplete* method), 4
get_queryset() (*users.views.AllUsersAutocomplete* method), 4
get_queryset() (*users.views.NonAdminUserAutocomplete* method), 4
get_queryset() (*users.views.NonSuperUserAutocomplete* method), 4
getProduct() (in module *gestion.views*), 2
getUser() (in module *users.views*), 5
global_message (*preferences.models.GeneralPreferences* attribute), 50
global_message (*preferences.models.HistoricalGeneralPreferences* attribute), 53
global_message() (in module *coopeV3.templatetags.vip*), 73
grocer (*preferences.models.GeneralPreferences* attribute), 50
grocer (*preferences.models.HistoricalGeneralPreferences* attribute), 53
grocer() (in module *coopeV3.templatetags.vip*), 73
groupProfile() (in module *users.views*), 5
GroupsEditForm (class in *users.forms*), 67
GroupsEditForm.Meta (class in *users.forms*), 68
groupsIndex() (in module *users.views*), 5

H

HistoricalConsumption (class in *gestion.models*), 10
HistoricalConsumption.DoesNotExist, 11

[HistoricalConsumption.MultipleObjectsReturned](#), 11
[HistoricalConsumptionHistory \(class in gestion.models\)](#), 12
[HistoricalConsumptionHistory.DoesNotExist](#), 12
[HistoricalConsumptionHistory.MultipleObjectsReturned](#), 12
[HistoricalCotisation \(class in preferences.models\)](#), 51
[HistoricalCotisation.DoesNotExist](#), 51
[HistoricalCotisation.MultipleObjectsReturned](#), 51
[HistoricalCotisationHistory \(class in users.models\)](#), 39
[HistoricalCotisationHistory.DoesNotExist](#), 39
[HistoricalCotisationHistory.MultipleObjectsReturned](#), 39
[HistoricalGeneralPreferences \(class in preferences.models\)](#), 52
[HistoricalGeneralPreferences.DoesNotExist](#), 52
[HistoricalGeneralPreferences.MultipleObjectsReturned](#), 53
[HistoricalKeg \(class in gestion.models\)](#), 14
[HistoricalKeg.DoesNotExist](#), 15
[HistoricalKeg.MultipleObjectsReturned](#), 15
[HistoricalKegHistory \(class in gestion.models\)](#), 17
[HistoricalKegHistory.DoesNotExist](#), 17
[HistoricalKegHistory.MultipleObjectsReturned](#), 17
[HistoricalMenu \(class in gestion.models\)](#), 18
[HistoricalMenu.DoesNotExist](#), 19
[HistoricalMenu.MultipleObjectsReturned](#), 19
[HistoricalMenuHistory \(class in gestion.models\)](#), 20
[HistoricalMenuHistory.DoesNotExist](#), 20
[HistoricalMenuHistory.MultipleObjectsReturned](#), 20
[HistoricalPaymentMethod \(class in preferences.models\)](#), 55
[HistoricalPaymentMethod.DoesNotExist](#), 55
[HistoricalPaymentMethod.MultipleObjectsReturned](#), 55
[HistoricalPinte \(class in gestion.models\)](#), 22
[HistoricalPinte.DoesNotExist](#), 22
[HistoricalPinte.MultipleObjectsReturned](#), 22
[HistoricalProduct \(class in gestion.models\)](#), 24
[HistoricalProduct.DoesNotExist](#), 24
[HistoricalProduct.MultipleObjectsReturned](#), 24
[HistoricalProfile \(class in users.models\)](#), 42
[HistoricalProfile.DoesNotExist](#), 42
[HistoricalProfile.MultipleObjectsReturned](#), 42
[HistoricalRefund \(class in gestion.models\)](#), 26
[HistoricalRefund.DoesNotExist](#), 26
[HistoricalRefund.MultipleObjectsReturned](#), 26
[HistoricalReload \(class in gestion.models\)](#), 28
[HistoricalReload.DoesNotExist](#), 28
[HistoricalReload.MultipleObjectsReturned](#), 28
[HistoricalSchool \(class in users.models\)](#), 44
[HistoricalSchool.DoesNotExist](#), 44
[HistoricalSchool.MultipleObjectsReturned](#), 44
[HistoricalWhiteListHistory \(class in users.models\)](#), 45
[HistoricalWhiteListHistory.DoesNotExist](#), 45
[HistoricalWhiteListHistory.MultipleObjectsReturned](#), 45
[history \(gestion.models.Consumption attribute\)](#), 9
[history \(gestion.models.ConsumptionHistory attribute\)](#), 10
[history \(gestion.models.Keg attribute\)](#), 30
[history \(gestion.models.KegHistory attribute\)](#), 31
[history \(gestion.models.Menu attribute\)](#), 32
[history \(gestion.models.MenuHistory attribute\)](#), 33
[history \(gestion.models.Pinte attribute\)](#), 34
[history \(gestion.models.Product attribute\)](#), 36
[history \(gestion.models.Refund attribute\)](#), 37
[history \(gestion.models.Reload attribute\)](#), 38
[history \(preferences.models.Cotisation attribute\)](#), 50
[history \(preferences.models.GeneralPreferences attribute\)](#), 50
[history \(preferences.models.PaymentMethod attribute\)](#), 57
[history \(users.models.CotisationHistory attribute\)](#), 39
[history \(users.models.Profile attribute\)](#), 47
[history \(users.models.School attribute\)](#), 48
[history \(users.models.WhiteListHistory attribute\)](#), 49
[history_change_reason \(gestion.models.HistoricalConsumption attribute\)](#), 11
[history_change_reason \(gestion.models.HistoricalConsumptionHistory attribute\)](#), 13
[history_change_reason \(gestion.models.HistoricalKeg attribute\)](#), 15
[history_change_reason \(gestion.models.HistoricalProduct attribute\)](#), 24

<i>tion.models.HistoricalKegHistory</i> attribute), 17	<i>history_date</i> (<i>gestion.models.HistoricalPinte</i> attribute), 23
<i>history_change_reason</i> (<i>gestion.models.HistoricalMenu</i> attribute), 19	<i>history_date</i> (<i>gestion.models.HistoricalProduct</i> attribute), 25
<i>history_change_reason</i> (<i>gestion.models.HistoricalMenuHistory</i> attribute), 21	<i>history_date</i> (<i>gestion.models.HistoricalRefund</i> attribute), 27
<i>history_change_reason</i> (<i>gestion.models.HistoricalPinte</i> attribute), 23	<i>history_date</i> (<i>gestion.models.HistoricalReload</i> attribute), 29
<i>history_change_reason</i> (<i>gestion.models.HistoricalProduct</i> attribute), 25	<i>history_date</i> (<i>preferences.models.HistoricalCotisation</i> attribute), 51
<i>history_change_reason</i> (<i>gestion.models.HistoricalRefund</i> attribute), 27	<i>history_date</i> (<i>preferences.models.HistoricalGeneralPreferences</i> attribute), 53
<i>history_change_reason</i> (<i>gestion.models.HistoricalReload</i> attribute), 29	<i>history_date</i> (<i>preferences.models.HistoricalPaymentMethod</i> attribute), 55
<i>history_change_reason</i> (<i>preferences.models.HistoricalCotisation</i> attribute), 51	<i>history_date</i> (<i>users.models.HistoricalCotisationHistory</i> attribute), 41
<i>history_change_reason</i> (<i>preferences.models.HistoricalGeneralPreferences</i> attribute), 53	<i>history_date</i> (<i>users.models.HistoricalProfile</i> attribute), 42
<i>history_change_reason</i> (<i>preferences.models.HistoricalPaymentMethod</i> attribute), 55	<i>history_date</i> (<i>users.models.HistoricalSchool</i> attribute), 44
<i>history_change_reason</i> (<i>users.models.HistoricalCotisationHistory</i> attribute), 40	<i>history_date</i> (<i>users.models.HistoricalWhiteListHistory</i> attribute), 46
<i>history_change_reason</i> (<i>users.models.HistoricalProfile</i> attribute), 42	<i>history_id</i> (<i>gestion.models.HistoricalConsumption</i> attribute), 11
<i>history_change_reason</i> (<i>users.models.HistoricalSchool</i> attribute), 44	<i>history_id</i> (<i>gestion.models.HistoricalConsumptionHistory</i> attribute), 13
<i>history_change_reason</i> (<i>users.models.HistoricalWhiteListHistory</i> attribute), 46	<i>history_id</i> (<i>gestion.models.HistoricalKeg</i> attribute), 16
<i>history_date</i> (<i>gestion.models.HistoricalConsumption</i> attribute), 11	<i>history_id</i> (<i>gestion.models.HistoricalKegHistory</i> attribute), 17
<i>history_date</i> (<i>gestion.models.HistoricalConsumptionHistory</i> attribute), 13	<i>history_id</i> (<i>gestion.models.HistoricalMenu</i> attribute), 19
<i>history_date</i> (<i>gestion.models.HistoricalKeg</i> attribute), 15	<i>history_id</i> (<i>gestion.models.HistoricalMenuHistory</i> attribute), 21
<i>history_date</i> (<i>gestion.models.HistoricalKegHistory</i> attribute), 17	<i>history_id</i> (<i>gestion.models.HistoricalPinte</i> attribute), 23
<i>history_date</i> (<i>gestion.models.HistoricalMenu</i> attribute), 19	<i>history_id</i> (<i>gestion.models.HistoricalProduct</i> attribute), 25
<i>history_date</i> (<i>gestion.models.HistoricalMenuHistory</i> attribute), 21	<i>history_id</i> (<i>gestion.models.HistoricalRefund</i> attribute), 27
	<i>history_id</i> (<i>gestion.models.HistoricalReload</i> attribute), 29
	<i>history_id</i> (<i>preferences.models.HistoricalCotisation</i> attribute), 52
	<i>history_id</i> (<i>preferences.models.HistoricalGeneralPreferences</i> attribute), 53
	<i>history_id</i> (<i>preferences.models.HistoricalPaymentMethod</i> attribute), 55
	<i>history_id</i> (<i>users.models.HistoricalCotisationHistory</i> attribute), 41

attribute), 41
 history_id (*users.models.HistoricalProfile* attribute), 42
 history_id (*users.models.HistoricalSchool* attribute), 44
 history_id (*users.models.HistoricalWhiteListHistory* attribute), 46
 history_object (*gestion.models.HistoricalConsumption* attribute), 11
 history_object (*gestion.models.HistoricalConsumptionHistory* attribute), 13
 history_object (*gestion.models.HistoricalKeg* attribute), 16
 history_object (*gestion.models.HistoricalKegHistory* attribute), 17
 history_object (*gestion.models.HistoricalMenu* attribute), 19
 history_object (*gestion.models.HistoricalMenuHistory* attribute), 21
 history_object (*gestion.models.HistoricalPinte* attribute), 23
 history_object (*gestion.models.HistoricalProduct* attribute), 25
 history_object (*gestion.models.HistoricalRefund* attribute), 27
 history_object (*gestion.models.HistoricalReload* attribute), 29
 history_object (*preferences.models.HistoricalCotisation* attribute), 52
 history_object (*preferences.models.HistoricalGeneralPreferences* attribute), 53
 history_object (*preferences.models.HistoricalPaymentMethod* attribute), 55
 history_object (*users.models.HistoricalCotisationHistory* attribute), 41
 history_object (*users.models.HistoricalProfile* attribute), 43
 history_object (*users.models.HistoricalSchool* attribute), 44
 history_object (*users.models.HistoricalWhiteListHistory* attribute), 46
 history_type (*gestion.models.HistoricalConsumption* attribute), 11
 history_type (*gestion.models.HistoricalConsumptionHistory* attribute), 13
 history_type (*gestion.models.HistoricalKeg* attribute), 16
 history_type (*gestion.models.HistoricalKegHistory* attribute), 17
 history_type (*gestion.models.HistoricalMenu* attribute), 19
 history_type (*gestion.models.HistoricalMenuHistory* attribute), 21
 history_type (*gestion.models.HistoricalPinte* attribute), 23
 history_type (*gestion.models.HistoricalProduct* attribute), 25
 history_type (*gestion.models.HistoricalRefund* attribute), 27
 history_type (*gestion.models.HistoricalReload* attribute), 29
 history_type (*preferences.models.HistoricalCotisation* attribute), 52
 history_type (*preferences.models.HistoricalGeneralPreferences* attribute), 53
 history_type (*preferences.models.HistoricalPaymentMethod* attribute), 55
 history_type (*users.models.HistoricalCotisationHistory* attribute), 41
 history_type (*users.models.HistoricalProfile* attribute), 43
 history_type (*users.models.HistoricalSchool* attribute), 44
 history_type (*users.models.HistoricalWhiteListHistory* attribute), 46
 history_user (*gestion.models.HistoricalConsumption* attribute), 11
 history_user (*gestion.models.HistoricalConsumptionHistory* attribute), 13
 history_user (*gestion.models.HistoricalKeg* attribute), 16
 history_user (*gestion.models.HistoricalKegHistory* attribute), 17
 history_user (*gestion.models.HistoricalMenu* attribute), 19
 history_user (*gestion.models.HistoricalMenuHistory* attribute), 21
 history_user (*gestion.models.HistoricalPinte* attribute), 23
 history_user (*gestion.models.HistoricalProduct* attribute), 25
 history_user (*gestion.models.HistoricalRefund* at-

tribute), 27

history_user (*gestion.models.HistoricalReload attribute*), 29

history_user (*preferences.models.HistoricalCotisation attribute*), 52

history_user (*preferences.models.HistoricalGeneralPreferences attribute*), 53

history_user (*preferences.models.HistoricalPaymentMethod attribute*), 55

history_user (*users.models.HistoricalCotisationHistory attribute*), 41

history_user (*users.models.HistoricalProfile attribute*), 43

history_user (*users.models.HistoricalSchool attribute*), 44

history_user (*users.models.HistoricalWhiteListHistory attribute*), 46

history_user_id (*gestion.models.HistoricalConsumption attribute*), 11

history_user_id (*gestion.models.HistoricalConsumptionHistory attribute*), 14

history_user_id (*gestion.models.HistoricalKeg attribute*), 16

history_user_id (*gestion.models.HistoricalKegHistory attribute*), 18

history_user_id (*gestion.models.HistoricalMenu attribute*), 19

history_user_id (*gestion.models.HistoricalMenuHistory attribute*), 21

history_user_id (*gestion.models.HistoricalPinte attribute*), 23

history_user_id (*gestion.models.HistoricalProduct attribute*), 25

history_user_id (*gestion.models.HistoricalRefund attribute*), 27

history_user_id (*gestion.models.HistoricalReload attribute*), 29

history_user_id (*preferences.models.HistoricalCotisation attribute*), 52

history_user_id (*preferences.models.HistoricalGeneralPreferences attribute*), 54

history_user_id (*preferences.models.HistoricalPaymentMethod attribute*), 55

history_user_id (*users.models.HistoricalCotisationHistory attribute*), 41

history_user_id (*users.models.HistoricalProfile attribute*), 43

history_user_id (*users.models.HistoricalSchool attribute*), 44

history_user_id (*users.models.HistoricalWhiteListHistory attribute*), 46

home () (*in module coopeV3.views*), 7

home_text (*preferences.models.GeneralPreferences attribute*), 50

home_text (*preferences.models.HistoricalGeneralPreferences attribute*), 54

homepage () (*in module coopeV3.views*), 7

I

icon (*preferences.models.HistoricalPaymentMethod attribute*), 56

icon (*preferences.models.PaymentMethod attribute*), 57

id (*gestion.models.Consumption attribute*), 9

id (*gestion.models.ConsumptionHistory attribute*), 10

id (*gestion.models.HistoricalConsumption attribute*), 11

id (*gestion.models.HistoricalConsumptionHistory attribute*), 14

id (*gestion.models.HistoricalKeg attribute*), 16

id (*gestion.models.HistoricalKegHistory attribute*), 18

id (*gestion.models.HistoricalMenu attribute*), 19

id (*gestion.models.HistoricalMenuHistory attribute*), 21

id (*gestion.models.HistoricalPinte attribute*), 23

id (*gestion.models.HistoricalProduct attribute*), 25

id (*gestion.models.HistoricalRefund attribute*), 27

id (*gestion.models.HistoricalReload attribute*), 29

id (*gestion.models.Keg attribute*), 30

id (*gestion.models.KegHistory attribute*), 31

id (*gestion.models.Menu attribute*), 32

id (*gestion.models.MenuHistory attribute*), 33

id (*gestion.models.Pinte attribute*), 34

id (*gestion.models.Product attribute*), 36

id (*gestion.models.Refund attribute*), 37

id (*gestion.models.Reload attribute*), 38

id (*preferences.models.Cotisation attribute*), 50

id (*preferences.models.GeneralPreferences attribute*), 50

id (*preferences.models.HistoricalCotisation attribute*), 52

id (*preferences.models.HistoricalGeneralPreferences attribute*), 54

id (*preferences.models.HistoricalPaymentMethod attribute*), 56

id (*preferences.models.PaymentMethod attribute*), 57

id (*users.models.CotisationHistory attribute*), 39

id (*users.models.HistoricalCotisationHistory attribute*), 41

id (*users.models.HistoricalProfile attribute*), 43

id (*users.models.HistoricalSchool attribute*), 44

- id (*users.models.HistoricalWhiteListHistory* attribute), 46
- id (*users.models.Profile* attribute), 47
- id (*users.models.School* attribute), 48
- id (*users.models.WhiteListHistory* attribute), 49
- inactive() (in module *preferences.views*), 7
- index() (in module *users.views*), 6
- instance (*gestion.models.HistoricalConsumption* attribute), 12
- instance (*gestion.models.HistoricalConsumptionHistory* attribute), 14
- instance (*gestion.models.HistoricalKeg* attribute), 16
- instance (*gestion.models.HistoricalKegHistory* attribute), 18
- instance (*gestion.models.HistoricalMenu* attribute), 19
- instance (*gestion.models.HistoricalMenuHistory* attribute), 21
- instance (*gestion.models.HistoricalPinte* attribute), 23
- instance (*gestion.models.HistoricalProduct* attribute), 25
- instance (*gestion.models.HistoricalRefund* attribute), 27
- instance (*gestion.models.HistoricalReload* attribute), 29
- instance (*preferences.models.HistoricalCotisation* attribute), 52
- instance (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- instance (*preferences.models.HistoricalPaymentMethod* attribute), 56
- instance (*users.models.HistoricalCotisationHistory* attribute), 41
- instance (*users.models.HistoricalProfile* attribute), 43
- instance (*users.models.HistoricalSchool* attribute), 45
- instance (*users.models.HistoricalWhiteListHistory* attribute), 46
- is_active (*gestion.models.HistoricalKeg* attribute), 16
- is_active (*gestion.models.HistoricalMenu* attribute), 20
- is_active (*gestion.models.HistoricalProduct* attribute), 25
- is_active (*gestion.models.Keg* attribute), 30
- is_active (*gestion.models.Menu* attribute), 32
- is_active (*gestion.models.Product* attribute), 36
- is_active (*preferences.models.GeneralPreferences* attribute), 50
- is_active (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- is_active (*preferences.models.HistoricalPaymentMethod* attribute), 56
- is_active (*preferences.models.PaymentMethod* attribute), 57
- is_adherent (*users.models.Profile* attribute), 47
- is_usable_in_cotisation (*preferences.models.HistoricalPaymentMethod* attribute), 56
- is_usable_in_cotisation (*preferences.models.PaymentMethod* attribute), 57
- is_usable_in_reload (*preferences.models.HistoricalPaymentMethod* attribute), 56
- is_usable_in_reload (*preferences.models.PaymentMethod* attribute), 56
- instance_type (*gestion.models.HistoricalConsumption* attribute), 12
- instance_type (*gestion.models.HistoricalConsumptionHistory* attribute), 14
- instance_type (*gestion.models.HistoricalKeg* attribute), 16
- instance_type (*gestion.models.HistoricalKegHistory* attribute), 18
- instance_type (*gestion.models.HistoricalMenu* attribute), 19
- instance_type (*gestion.models.HistoricalMenuHistory* attribute), 21
- instance_type (*gestion.models.HistoricalPinte* attribute), 23
- instance_type (*gestion.models.HistoricalProduct* attribute), 25
- instance_type (*gestion.models.HistoricalRefund* attribute), 27
- instance_type (*gestion.models.HistoricalReload* attribute), 29
- instance_type (*preferences.models.HistoricalCotisation* attribute), 52
- instance_type (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- instance_type (*preferences.models.HistoricalPaymentMethod* attribute), 56
- instance_type (*users.models.HistoricalCotisationHistory* attribute), 41
- instance_type (*users.models.HistoricalProfile* attribute), 43
- instance_type (*users.models.HistoricalSchool* attribute), 45
- instance_type (*users.models.HistoricalWhiteListHistory* attribute), 46

- 57
 isCurrentKegHistory (*gestion.models.HistoricalKegHistory* attribute), 18
 isCurrentKegHistory (*gestion.models.KegHistory* attribute), 31
 isDemi () (*in module gestion.models*), 38
 isGalopin () (*in module gestion.models*), 38
 isPinte () (*in module gestion.models*), 38
- ## K
- Keg (*class in gestion.models*), 30
 keg (*gestion.models.HistoricalKegHistory* attribute), 18
 keg (*gestion.models.KegHistory* attribute), 31
 Keg.DoesNotExist, 30
 Keg.MultipleObjectsReturned, 30
 keg_id (*gestion.models.HistoricalKegHistory* attribute), 18
 keg_id (*gestion.models.KegHistory* attribute), 31
 KegActiveAutocomplete (*class in gestion.views*), 1
 KegAdmin (*class in gestion.admin*), 59
 KegForm (*class in gestion.forms*), 63
 KegForm.Meta (*class in gestion.forms*), 63
 kegH () (*in module gestion.views*), 2
 KegHistory (*class in gestion.models*), 31
 KegHistory.DoesNotExist, 31
 KegHistory.MultipleObjectsReturned, 31
 keghistory_set (*gestion.models.Keg* attribute), 30
 KegHistoryAdmin (*class in gestion.admin*), 59
 KegPositiveAutocomplete (*class in gestion.views*), 1
 kegsList () (*in module gestion.views*), 2
- ## L
- last_update_date (*gestion.models.HistoricalPinte* attribute), 23
 last_update_date (*gestion.models.Pinte* attribute), 34
 list_display (*gestion.admin.ConsumptionAdmin* attribute), 59
 list_display (*gestion.admin.ConsumptionHistoryAdmin* attribute), 59
 list_display (*gestion.admin.KegAdmin* attribute), 59
 list_display (*gestion.admin.KegHistoryAdmin* attribute), 60
 list_display (*gestion.admin.MenuAdmin* attribute), 60
 list_display (*gestion.admin.MenuHistoryAdmin* attribute), 60
 list_display (*gestion.admin.ProductAdmin* attribute), 60
 list_display (*gestion.admin.RefundAdmin* attribute), 60
 list_display (*gestion.admin.ReloadAdmin* attribute), 60
 list_display (*preferences.admin.CotisationAdmin* attribute), 61
 list_display (*preferences.admin.GeneralPreferencesAdmin* attribute), 62
 list_display (*preferences.admin.PaymentMethodAdmin* attribute), 62
 list_display (*users.admin.CotisationHistoryAdmin* attribute), 61
 list_display (*users.admin.ProfileAdmin* attribute), 61
 list_display (*users.admin.WhiteListHistoryAdmin* attribute), 61
 list_filter (*gestion.admin.ConsumptionHistoryAdmin* attribute), 59
 list_filter (*gestion.admin.KegAdmin* attribute), 59
 list_filter (*gestion.admin.KegHistoryAdmin* attribute), 60
 list_filter (*gestion.admin.MenuAdmin* attribute), 60
 list_filter (*gestion.admin.ProductAdmin* attribute), 60
 list_filter (*gestion.admin.ReloadAdmin* attribute), 60
 list_filter (*preferences.admin.PaymentMethodAdmin* attribute), 62
 list_filter (*users.admin.CotisationHistoryAdmin* attribute), 61
 list_filter (*users.admin.ProfileAdmin* attribute), 61
 LoginForm (*class in users.forms*), 68
 loginView () (*in module users.views*), 6
 logout_time () (*in module coopeV3.templatetags.vip*), 73
 logoutView () (*in module users.views*), 6
 lookups () (*users.admin.BalanceFilter* method), 61
 lost_pintes_allowed (*preferences.models.GeneralPreferences* attribute), 51
 lost_pintes_allowed (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- ## M
- manage () (*in module gestion.views*), 2
 media (*gestion.admin.ConsumptionAdmin* attribute), 59
 media (*gestion.admin.ConsumptionHistoryAdmin* attribute), 59
 media (*gestion.admin.KegAdmin* attribute), 59

- media (*gestion.admin.KegHistoryAdmin* attribute), 60
- media (*gestion.admin.MenuAdmin* attribute), 60
- media (*gestion.admin.MenuHistoryAdmin* attribute), 60
- media (*gestion.admin.ProductAdmin* attribute), 60
- media (*gestion.admin.RefundAdmin* attribute), 60
- media (*gestion.admin.ReloadAdmin* attribute), 60
- media (*gestion.forms.GenerateReleveForm* attribute), 63
- media (*gestion.forms.GestionForm* attribute), 63
- media (*gestion.forms.KegForm* attribute), 64
- media (*gestion.forms.MenuForm* attribute), 64
- media (*gestion.forms.PinteForm* attribute), 64
- media (*gestion.forms.ProductForm* attribute), 64
- media (*gestion.forms.RefundForm* attribute), 65
- media (*gestion.forms.ReloadForm* attribute), 65
- media (*gestion.forms.SearchMenuForm* attribute), 65
- media (*gestion.forms.SearchProductForm* attribute), 65
- media (*gestion.forms.SelectActiveKegForm* attribute), 66
- media (*gestion.forms.SelectPositiveKegForm* attribute), 66
- media (*preferences.admin.CotisationAdmin* attribute), 61
- media (*preferences.admin.GeneralPreferencesAdmin* attribute), 62
- media (*preferences.admin.PaymentMethodAdmin* attribute), 62
- media (*preferences.forms.CotisationForm* attribute), 70
- media (*preferences.forms.GeneralPreferencesForm* attribute), 70
- media (*preferences.forms.PaymentMethodForm* attribute), 71
- media (*users.admin.CotisationHistoryAdmin* attribute), 61
- media (*users.admin.ProfileAdmin* attribute), 61
- media (*users.admin.WhiteListHistoryAdmin* attribute), 61
- media (*users.forms.addCotisationHistoryForm* attribute), 69
- media (*users.forms.addWhiteListHistoryForm* attribute), 70
- media (*users.forms.CreateGroupForm* attribute), 66
- media (*users.forms.CreateUserForm* attribute), 67
- media (*users.forms.EditGroupForm* attribute), 67
- media (*users.forms.EditPasswordForm* attribute), 67
- media (*users.forms.ExportForm* attribute), 67
- media (*users.forms.GroupsEditForm* attribute), 68
- media (*users.forms.LoginForm* attribute), 68
- media (*users.forms.SchoolForm* attribute), 68
- media (*users.forms.SelectNonAdminUserForm* attribute), 68
- media (*users.forms.SelectNonSuperUserForm* attribute), 69
- media (*users.forms.SelectUserForm* attribute), 69
- Menu (class in *gestion.models*), 32
- menu (*gestion.models.HistoricalMenuHistory* attribute), 21
- menu (*gestion.models.MenuHistory* attribute), 33
- menu (*preferences.models.GeneralPreferences* attribute), 51
- menu (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- menu () (in module *coopeV3.templatetags.vip*), 73
- Menu.DoesNotExist, 32
- Menu.MultipleObjectsReturned, 32
- menu_id (*gestion.models.HistoricalMenuHistory* attribute), 22
- menu_id (*gestion.models.MenuHistory* attribute), 33
- menu_set (*gestion.models.Product* attribute), 36
- MenuAdmin (class in *gestion.admin*), 60
- MenuForm (class in *gestion.forms*), 64
- MenuForm.Meta (class in *gestion.forms*), 64
- MenuHistory (class in *gestion.models*), 32
- MenuHistory.DoesNotExist, 32
- MenuHistory.MultipleObjectsReturned, 32
- menuhistory_set (*gestion.models.Menu* attribute), 32
- menuhistory_set (*preferences.models.PaymentMethod* attribute), 57
- MenuHistoryAdmin (class in *gestion.admin*), 60
- menus_list () (in module *gestion.views*), 3
- MenusAutocomplete (class in *gestion.views*), 1
- model (*gestion.forms.KegForm.Meta* attribute), 63
- model (*gestion.forms.MenuForm.Meta* attribute), 64
- model (*gestion.forms.ProductForm.Meta* attribute), 64
- model (*gestion.forms.RefundForm.Meta* attribute), 65
- model (*gestion.forms.ReloadForm.Meta* attribute), 65
- model (*preferences.forms.CotisationForm.Meta* attribute), 70
- model (*preferences.forms.GeneralPreferencesForm.Meta* attribute), 70
- model (*preferences.forms.PaymentMethodForm.Meta* attribute), 71
- model (*users.forms.addCotisationHistoryForm.Meta* attribute), 69
- model (*users.forms.addWhiteListHistoryForm.Meta* attribute), 69
- model (*users.forms.CreateGroupForm.Meta* attribute), 66
- model (*users.forms.CreateUserForm.Meta* attribute), 67
- model (*users.forms.EditGroupForm.Meta* attribute), 67
- model (*users.forms.GroupsEditForm.Meta* attribute), 68
- model (*users.forms.SchoolForm.Meta* attribute), 68

N

- name (*django_tex.models.TeXTemplateFile* attribute), 76
- name (*gestion.models.HistoricalKeg* attribute), 16

- name (*gestion.models.HistoricalMenu* attribute), 20
- name (*gestion.models.HistoricalProduct* attribute), 25
- name (*gestion.models.Keg* attribute), 31
- name (*gestion.models.Menu* attribute), 32
- name (*gestion.models.Product* attribute), 36
- name (*preferences.models.HistoricalPaymentMethod* attribute), 56
- name (*preferences.models.PaymentMethod* attribute), 57
- name (*users.models.HistoricalSchool* attribute), 45
- name (*users.models.School* attribute), 48
- nb_pintes (*users.models.Profile* attribute), 47
- needQuantityButton (*gestion.models.HistoricalProduct* attribute), 25
- needQuantityButton (*gestion.models.Product* attribute), 36
- next_record (*gestion.models.HistoricalConsumption* attribute), 12
- next_record (*gestion.models.HistoricalConsumptionHistory* attribute), 14
- next_record (*gestion.models.HistoricalKeg* attribute), 16
- next_record (*gestion.models.HistoricalKegHistory* attribute), 18
- next_record (*gestion.models.HistoricalMenu* attribute), 20
- next_record (*gestion.models.HistoricalMenuHistory* attribute), 22
- next_record (*gestion.models.HistoricalPinte* attribute), 24
- next_record (*gestion.models.HistoricalProduct* attribute), 25
- next_record (*gestion.models.HistoricalRefund* attribute), 27
- next_record (*gestion.models.HistoricalReload* attribute), 30
- next_record (*preferences.models.HistoricalCotisation* attribute), 52
- next_record (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- next_record (*preferences.models.HistoricalPaymentMethod* attribute), 56
- next_record (*users.models.HistoricalCotisationHistory* attribute), 41
- next_record (*users.models.HistoricalProfile* attribute), 43
- next_record (*users.models.HistoricalSchool* attribute), 45
- next_record (*users.models.HistoricalWhiteListHistory* attribute), 46
- NonAdminUserAutocomplete (class in *users.views*), 4
- NonSuperUserAutocomplete (class in *users.views*), 4
- ## O
- objects (*gestion.models.Consumption* attribute), 9
- objects (*gestion.models.ConsumptionHistory* attribute), 10
- objects (*gestion.models.HistoricalConsumption* attribute), 12
- objects (*gestion.models.HistoricalConsumptionHistory* attribute), 14
- objects (*gestion.models.HistoricalKeg* attribute), 16
- objects (*gestion.models.HistoricalKegHistory* attribute), 18
- objects (*gestion.models.HistoricalMenu* attribute), 20
- objects (*gestion.models.HistoricalMenuHistory* attribute), 22
- objects (*gestion.models.HistoricalPinte* attribute), 24
- objects (*gestion.models.HistoricalProduct* attribute), 25
- objects (*gestion.models.HistoricalRefund* attribute), 28
- objects (*gestion.models.HistoricalReload* attribute), 30
- objects (*gestion.models.Keg* attribute), 31
- objects (*gestion.models.KegHistory* attribute), 31
- objects (*gestion.models.Menu* attribute), 32
- objects (*gestion.models.MenuHistory* attribute), 33
- objects (*gestion.models.Pinte* attribute), 34
- objects (*gestion.models.Product* attribute), 36
- objects (*gestion.models.Refund* attribute), 37
- objects (*gestion.models.Reload* attribute), 38
- objects (*preferences.models.Cotisation* attribute), 50
- objects (*preferences.models.GeneralPreferences* attribute), 51
- objects (*preferences.models.HistoricalCotisation* attribute), 52
- objects (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- objects (*preferences.models.HistoricalPaymentMethod* attribute), 56
- objects (*preferences.models.PaymentMethod* attribute), 57
- objects (*users.models.CotisationHistory* attribute), 39
- objects (*users.models.HistoricalCotisationHistory* attribute), 41
- objects (*users.models.HistoricalProfile* attribute), 43
- objects (*users.models.HistoricalSchool* attribute), 45
- objects (*users.models.HistoricalWhiteListHistory* attribute), 46
- objects (*users.models.Profile* attribute), 47
- objects (*users.models.School* attribute), 48
- objects (*users.models.WhiteListHistory* attribute), 49

- openDirectKeg() (in module *gestion.views*), 3
- openingDate (*gestion.models.HistoricalKegHistory* attribute), 18
- openingDate (*gestion.models.KegHistory* attribute), 31
- openKeg() (in module *gestion.views*), 3
- order() (in module *gestion.views*), 3
- ordering (*gestion.admin.ConsumptionAdmin* attribute), 59
- ordering (*gestion.admin.ConsumptionHistoryAdmin* attribute), 59
- ordering (*gestion.admin.KegAdmin* attribute), 59
- ordering (*gestion.admin.KegHistoryAdmin* attribute), 60
- ordering (*gestion.admin.MenuAdmin* attribute), 60
- ordering (*gestion.admin.MenuHistoryAdmin* attribute), 60
- ordering (*gestion.admin.ProductAdmin* attribute), 60
- ordering (*gestion.admin.RefundAdmin* attribute), 60
- ordering (*gestion.admin.ReloadAdmin* attribute), 60
- ordering (*preferences.admin.CotisationAdmin* attribute), 62
- ordering (*preferences.admin.PaymentMethodAdmin* attribute), 62
- ordering (*users.admin.CotisationHistoryAdmin* attribute), 61
- ordering (*users.admin.ProfileAdmin* attribute), 61
- ordering (*users.admin.WhiteListHistoryAdmin* attribute), 61
- ## P
- P_PRESSION (*gestion.models.Product* attribute), 34
- PANINI (*gestion.models.Product* attribute), 34
- parameter_name (*users.admin.BalanceFilter* attribute), 61
- paymentDate (*users.models.CotisationHistory* attribute), 39
- paymentDate (*users.models.HistoricalCotisationHistory* attribute), 41
- paymentDate (*users.models.HistoricalWhiteListHistory* attribute), 46
- paymentDate (*users.models.WhiteListHistory* attribute), 49
- PaymentMethod (class in *preferences.models*), 56
- paymentMethod (*gestion.models.ConsumptionHistory* attribute), 10
- paymentMethod (*gestion.models.HistoricalConsumptionHistory* attribute), 14
- paymentMethod (*gestion.models.HistoricalMenuHistory* attribute), 22
- PaymentMethod (*gestion.models.HistoricalReload* attribute), 28
- paymentMethod (*gestion.models.MenuHistory* attribute), 33
- PaymentMethod (*gestion.models.Reload* attribute), 37
- paymentMethod (*users.models.CotisationHistory* attribute), 39
- paymentMethod (*users.models.HistoricalCotisationHistory* attribute), 41
- PaymentMethodAdmin (class in *preferences.admin*), 62
- PaymentMethodForm (class in *preferences.forms*), 70
- PaymentMethodForm.Meta (class in *preferences.forms*), 70
- paymentMethodsIndex() (in module *preferences.views*), 7
- PDFResponse (class in *django_tex.views*), 76
- Pinte (class in *gestion.models*), 33
- pinte (*gestion.models.HistoricalKeg* attribute), 16
- pinte (*gestion.models.Keg* attribute), 31
- Pinte.DoesNotExist, 33
- Pinte.MultipleObjectsReturned, 34
- pinte_id (*gestion.models.HistoricalKeg* attribute), 16
- pinte_id (*gestion.models.Keg* attribute), 31
- PinteForm (class in *gestion.forms*), 64
- pintes_list() (in module *gestion.views*), 3
- pintes_user_list() (in module *gestion.views*), 3
- positiveBalance() (*users.models.Profile* method), 47
- preferences.admin (module), 61
- preferences.forms (module), 70
- preferences.models (module), 49
- preferences.views (module), 6

- president (*preferences.models.GeneralPreferences* attribute), 51
 president (*preferences.models.HistoricalGeneralPreferences* attribute), 54
 president () (in module *coopeV3.templatetags.vip*), 74
 prettify_message () (in module *django_tex.exceptions*), 75
 prev_record (*gestion.models.HistoricalConsumption* attribute), 12
 prev_record (*gestion.models.HistoricalConsumptionHistory* attribute), 14
 prev_record (*gestion.models.HistoricalKeg* attribute), 17
 prev_record (*gestion.models.HistoricalKegHistory* attribute), 18
 prev_record (*gestion.models.HistoricalMenu* attribute), 20
 prev_record (*gestion.models.HistoricalMenuHistory* attribute), 22
 prev_record (*gestion.models.HistoricalPinte* attribute), 24
 prev_record (*gestion.models.HistoricalProduct* attribute), 26
 prev_record (*gestion.models.HistoricalRefund* attribute), 28
 prev_record (*gestion.models.HistoricalReload* attribute), 30
 prev_record (*preferences.models.HistoricalCotisation* attribute), 52
 prev_record (*preferences.models.HistoricalGeneralPreferences* attribute), 54
 prev_record (*preferences.models.HistoricalPaymentMethod* attribute), 56
 prev_record (*users.models.HistoricalCotisationHistory* attribute), 42
 prev_record (*users.models.HistoricalProfile* attribute), 43
 prev_record (*users.models.HistoricalSchool* attribute), 45
 prev_record (*users.models.HistoricalWhiteListHistory* attribute), 46
 previous_owner (*gestion.models.HistoricalPinte* attribute), 24
 previous_owner (*gestion.models.Pinte* attribute), 34
 previous_owner_id (*gestion.models.HistoricalPinte* attribute), 24
 previous_owner_id (*gestion.models.Pinte* attribute), 34
 Product (class in *gestion.models*), 34
 product (*gestion.models.Consumption* attribute), 9
 product (*gestion.models.ConsumptionHistory* attribute), 10
 product (*gestion.models.HistoricalConsumption* attribute), 12
 product (*gestion.models.HistoricalConsumptionHistory* attribute), 14
 Product.DoesNotExist, 34
 Product.MultipleObjectsReturned, 34
 product_id (*gestion.models.Consumption* attribute), 9
 product_id (*gestion.models.ConsumptionHistory* attribute), 10
 product_id (*gestion.models.HistoricalConsumption* attribute), 12
 product_id (*gestion.models.HistoricalConsumptionHistory* attribute), 14
 ProductAdmin (class in *gestion.admin*), 60
 ProductForm (class in *gestion.forms*), 64
 ProductForm.Meta (class in *gestion.forms*), 64
 productProfile () (in module *gestion.views*), 3
 ProductsAutocomplete (class in *gestion.views*), 1
 productsIndex () (in module *gestion.views*), 3
 productsList () (in module *gestion.views*), 3
 Profile (class in *users.models*), 47
 profile () (in module *users.views*), 6
 Profile.DoesNotExist, 47
 Profile.MultipleObjectsReturned, 47
 profile_set (*users.models.School* attribute), 48
 ProfileAdmin (class in *users.admin*), 61
- ## Q
- quantity (*gestion.models.Consumption* attribute), 9
 quantity (*gestion.models.ConsumptionHistory* attribute), 10
 quantity (*gestion.models.HistoricalConsumption* attribute), 12
 quantity (*gestion.models.HistoricalConsumptionHistory* attribute), 14
 quantity (*gestion.models.HistoricalMenuHistory* attribute), 22
 quantity (*gestion.models.MenuHistory* attribute), 33
 quantitySold (*gestion.models.HistoricalKegHistory* attribute), 18
 quantitySold (*gestion.models.KegHistory* attribute), 31
 QUERY_TYPE_CHOICES (*users.forms.ExportForm* attribute), 67
 queryset () (*users.admin.BalanceFilter* method), 61
- ## R
- random_filter () (in module *users.templatetags.users_extra*), 74
 rank (*users.models.Profile* attribute), 47
 ranking (*gestion.models.Product* attribute), 36
 ranking () (in module *gestion.views*), 3

- Refund (class in *gestion.models*), 37
- refund() (in module *gestion.views*), 3
- Refund.DoesNotExist, 37
- Refund.MultipleObjectsReturned, 37
- RefundAdmin (class in *gestion.admin*), 60
- RefundForm (class in *gestion.forms*), 65
- RefundForm.Meta (class in *gestion.forms*), 65
- release() (in module *gestion.views*), 3
- release_pintes() (in module *gestion.views*), 3
- Reload (class in *gestion.models*), 37
- reload() (in module *gestion.views*), 3
- Reload.DoesNotExist, 37
- Reload.MultipleObjectsReturned, 37
- reload_set (*preferences.models.PaymentMethod* attribute), 57
- ReloadAdmin (class in *gestion.admin*), 60
- ReloadForm (class in *gestion.forms*), 65
- ReloadForm.Meta (class in *gestion.forms*), 65
- removeAdmin() (in module *users.views*), 6
- removeRight() (in module *users.views*), 6
- removeSuperuser() (in module *users.views*), 6
- removeUser() (in module *users.views*), 6
- render_template_with_context() (in module *django_tex.core*), 75
- render_to_pdf() (in module *django_tex.views*), 76
- resetPassword() (in module *users.views*), 6
- revert_url() (*gestion.models.HistoricalConsumption* method), 12
- revert_url() (*gestion.models.HistoricalConsumptionHistory* method), 14
- revert_url() (*gestion.models.HistoricalKeg* method), 17
- revert_url() (*gestion.models.HistoricalKegHistory* method), 18
- revert_url() (*gestion.models.HistoricalMenu* method), 20
- revert_url() (*gestion.models.HistoricalMenuHistory* method), 22
- revert_url() (*gestion.models.HistoricalPinte* method), 24
- revert_url() (*gestion.models.HistoricalProduct* method), 26
- revert_url() (*gestion.models.HistoricalRefund* method), 28
- revert_url() (*gestion.models.HistoricalReload* method), 30
- revert_url() (*preferences.models.HistoricalCotisation* method), 52
- revert_url() (*preferences.models.HistoricalGeneralPreferences* method), 54
- revert_url() (*preferences.models.HistoricalPaymentMethod* method), 56
- revert_url() (*users.models.HistoricalCotisationHistory* method), 42
- revert_url() (*users.models.HistoricalProfile* method), 43
- revert_url() (*users.models.HistoricalSchool* method), 45
- revert_url() (*users.models.HistoricalWhiteListHistory* method), 46
- rules (*preferences.models.GeneralPreferences* attribute), 51
- rules (*preferences.models.HistoricalGeneralPreferences* attribute), 54
- rules() (in module *coopeV3.templatetags.vip*), 74
- run_tex() (in module *django_tex.core*), 75
- ## S
- save_user_profile() (in module *users.models*), 49
- save_without_historical_record() (*gestion.models.Consumption* method), 9
- save_without_historical_record() (*gestion.models.ConsumptionHistory* method), 10
- save_without_historical_record() (*gestion.models.Keg* method), 31
- save_without_historical_record() (*gestion.models.KegHistory* method), 31
- save_without_historical_record() (*gestion.models.Menu* method), 32
- save_without_historical_record() (*gestion.models.MenuHistory* method), 33
- save_without_historical_record() (*gestion.models.Pinte* method), 34
- save_without_historical_record() (*gestion.models.Product* method), 36
- save_without_historical_record() (*gestion.models.Refund* method), 37
- save_without_historical_record() (*gestion.models.Reload* method), 38
- save_without_historical_record() (*preferences.models.Cotisation* method), 50
- save_without_historical_record() (*preferences.models.GeneralPreferences* method), 51
- save_without_historical_record() (*preferences.models.PaymentMethod* method), 58
- save_without_historical_record() (*users.models.CotisationHistory* method), 39
- save_without_historical_record() (*users.models.Profile* method), 47

- save_without_historical_record() (users.models.School method), 48
 save_without_historical_record() (users.models.WhiteListHistory method), 49
 School (class in users.models), 48
 school (users.models.HistoricalProfile attribute), 43
 school (users.models.Profile attribute), 48
 School.DoesNotExist, 48
 School.MultipleObjectsReturned, 48
 school_id (users.models.HistoricalProfile attribute), 43
 school_id (users.models.Profile attribute), 48
 SchoolForm (class in users.forms), 68
 SchoolForm.Meta (class in users.forms), 68
 schoolsIndex() (in module users.views), 6
 search_fields (gestion.admin.ConsumptionAdmin attribute), 59
 search_fields (gestion.admin.ConsumptionHistoryAdmin attribute), 59
 search_fields (gestion.admin.KegAdmin attribute), 59
 search_fields (gestion.admin.KegHistoryAdmin attribute), 60
 search_fields (gestion.admin.MenuAdmin attribute), 60
 search_fields (gestion.admin.MenuHistoryAdmin attribute), 60
 search_fields (gestion.admin.ProductAdmin attribute), 60
 search_fields (gestion.admin.RefundAdmin attribute), 60
 search_fields (gestion.admin.ReloadAdmin attribute), 61
 search_fields (preferences.admin.PaymentMethodAdmin attribute), 62
 search_fields (users.admin.CotisationHistoryAdmin attribute), 61
 search_fields (users.admin.ProfileAdmin attribute), 61
 search_fields (users.admin.WhiteListHistoryAdmin attribute), 61
 searchMenu() (in module gestion.views), 3
 SearchMenuForm (class in gestion.forms), 65
 searchProduct() (in module gestion.views), 3
 SearchProductForm (class in gestion.forms), 65
 searchUser() (in module users.views), 6
 secretary (preferences.models.GeneralPreferences attribute), 51
 secretary (preferences.models.HistoricalGeneralPreferences attribute), 54
 secretary() (in module coopeV3.templatetags.vip), 74
 SelectActiveKegForm (class in gestion.forms), 66
 SelectNonAdminUserForm (class in users.forms), 68
 SelectNonSuperUserForm (class in users.forms), 69
 SelectPositiveKegForm (class in gestion.forms), 66
 SelectUserForm (class in users.forms), 69
 self_or_has_perm() (in module coopeV3.acl), 73
 showingMultiplier (gestion.models.HistoricalProduct attribute), 26
 showingMultiplier (gestion.models.Product attribute), 36
 SOFT (gestion.models.Product attribute), 34
 statutes (preferences.models.GeneralPreferences attribute), 51
 statutes (preferences.models.HistoricalGeneralPreferences attribute), 54
 statutes() (in module coopeV3.templatetags.vip), 74
 stockBar (gestion.models.HistoricalProduct attribute), 26
 stockBar (gestion.models.Product attribute), 36
 stockHold (gestion.models.HistoricalKeg attribute), 17
 stockHold (gestion.models.HistoricalProduct attribute), 26
 stockHold (gestion.models.Keg attribute), 31
 stockHold (gestion.models.Product attribute), 36
 str_user() (in module users.models), 49
 superuser_required() (in module coopeV3.acl), 73
 superusersIndex() (in module users.views), 6
 switch_activate() (in module gestion.views), 3
 switch_activate_menu() (in module gestion.views), 3
 switch_activate_user() (in module users.views), 6
- ## T
- TeXEngine (class in django_tex.engine), 75
 TexError, 75
 TeXTemplateFile (class in django_tex.models), 76
 TeXTemplateFile.Meta (class in django_tex.models), 76
 title (django_tex.models.TeXTemplateFile attribute), 76
 title (users.admin.BalanceFilter attribute), 61
 tokenizer() (in module django_tex.exceptions), 76
 treasurer (preferences.models.GeneralPreferences attribute), 51
 treasurer (preferences.models.HistoricalGeneralPreferences attribute), 54

- treasurer() (in module *coopeV3.templatetags.vip*), 74
- TYPEINPUT_CHOICES_CATEGORIE (gestion.models.Product attribute), 34
- ## U
- use_pinte_monitoring (preferences.models.GeneralPreferences attribute), 51
- use_pinte_monitoring (preferences.models.HistoricalGeneralPreferences attribute), 54
- user (users.models.CotisationHistory attribute), 39
- user (users.models.HistoricalCotisationHistory attribute), 42
- user (users.models.HistoricalProfile attribute), 43
- user (users.models.HistoricalWhiteListHistory attribute), 47
- user (users.models.Profile attribute), 48
- user (users.models.WhiteListHistory attribute), 49
- user_id (users.models.CotisationHistory attribute), 39
- user_id (users.models.HistoricalCotisationHistory attribute), 42
- user_id (users.models.HistoricalProfile attribute), 44
- user_id (users.models.HistoricalWhiteListHistory attribute), 47
- user_id (users.models.Profile attribute), 48
- user_id (users.models.WhiteListHistory attribute), 49
- user_ranking() (gestion.models.Product method), 36
- users.admin (module), 61
- users.forms (module), 66
- users.models (module), 38
- users.templatetags.users_extra (module), 74
- users.views (module), 4
- usersIndex() (in module *users.views*), 6
- ## V
- validate_template_path() (in module *django_tex.models*), 76
- vice_president (preferences.models.GeneralPreferences attribute), 51
- vice_president (preferences.models.HistoricalGeneralPreferences attribute), 55
- vice_president() (in module *coopeV3.templatetags.vip*), 74
- volume (gestion.models.HistoricalProduct attribute), 26
- volume (gestion.models.Product attribute), 37
- WhiteListHistory.DoesNotExist, 48
- WhiteListHistory.MultipleObjectsReturned, 48
- WhiteListHistoryAdmin (class in *users.admin*), 61
- widgets (gestion.forms.KegForm.Meta attribute), 64
- widgets (gestion.forms.MenuForm.Meta attribute), 64
- widgets (gestion.forms.ProductForm.Meta attribute), 64
- widgets (gestion.forms.RefundForm.Meta attribute), 65
- widgets (gestion.forms.ReloadForm.Meta attribute), 65
- widgets (preferences.forms.GeneralPreferencesForm.Meta attribute), 70
- ## W
- WhiteListHistory (class in *users.models*), 48