
django-cruds-adminlte Documentation

Release 0.0.3

Óscar M. Lage

Jul 02, 2020

Contents

1	django-cruds-adminlte	3
1.1	History and goal	3
1.2	Features	4
1.3	Online Resources	4
1.4	Screenshots	5
2	Installation	11
2.1	Installation	11
3	Templates and templatetags	15
3.1	Templates and templatetags	15
4	Components	17
4.1	Components	17
4.2	CRUDView Usage	25
4.3	UserCRUDView Usage	31
4.4	InlineAjaxCRUD Usage	32
4.5	CRUDMixin Usage	32
4.6	Demo / testing project	33
5	Screenshots	35
5.1	Screenshots	35
6	Thanks	43
6.1	Thanks	43
7	Changelog	45
7.1	Changelog	45
7.2	Indices and tables	53

Welcome to django-cruds-adminlte's documentation!

django-cruds-adminlte

`django-cruds-adminlte` is simple drop-in django app that creates CRUD (Create, read, update and delete) views for existing models and apps.

`django-cruds-adminlte` goal is to make prototyping faster.

- Note: This version of `django-cruds-adminlte` is based on [bmihelac's one](#).

1.1 History and goal

Developers spends a lot of time just doing cruds, Django built-in admin was pretty and really nice... years ago. Right now customers (and people in general) are more used to the web and they want to change whatever on their smartphones, upload images cropping the important part, they're used to `select2` (or similar) in selects with so many options, etc..

A friend of mine told me to try backpack for laravel (<https://backpackforlaravel.com/>), well in fact he showed me a demo. I was impressed with what he could do just configuring a bit the models and the forms:

- Responsive design, more or less I guess you could use `django-flat-responsive` for that
- Tabbed forms: really easy to place fields in tabs, imho much more useful for the end user if the form is complex and has many fields (I've found nothing similar for django's admin)
- Wrappable fields: You can define the wrapper of the label+input (col-6, col-12), so it's easy to place fields side-by-side or 3 in a row, etc... You can do the same with `django-crispy-forms` but I've seen no easy way to integrate it on django's admin. Note from [@spookylukey](#): There is a [really easy way](#) to put the fields side-by-side in the django's contrib admin.
- `Select2` for selects with fk, etc... I've tried `django-select2` + `django-easy-select2` with not too much luck (I'm sure it was my fault), didn't know `django-autocomplete-light` tbh.
- Lots of widgets depending on the type of field (44+ field types: date, time, datetime, toggle, video...).
- Lots of columns - the field representation in a listing table - (images, data with/without link, buttons, extra buttons...).

- Reordering - nested sortable - (something similar to `django-mptt`)...

After seeing all that stuff I felt a bit shocked, started to look for something similar for django (for the built-in admin or some other piece of code that gives me something closer). I've tried `django-material`, `django-jet`, `grappelli`, `django-adminlte2`, `djadmin`, `django-flat-responsive`... but in the end I felt that only a cocktail with some of them could do the job. I did a list of soft and features (similar to the above's one) and, in the end, I've started to think that if I had that need, why not to make it public and test if the community feels same "lacks" than me?. That's the story behind this project.

Crazy? yep, I felt myself really weird after read Jacob's post (<https://jacobian.org/writing/so-you-want-a-new-admin/>) but I needed to make the project public.

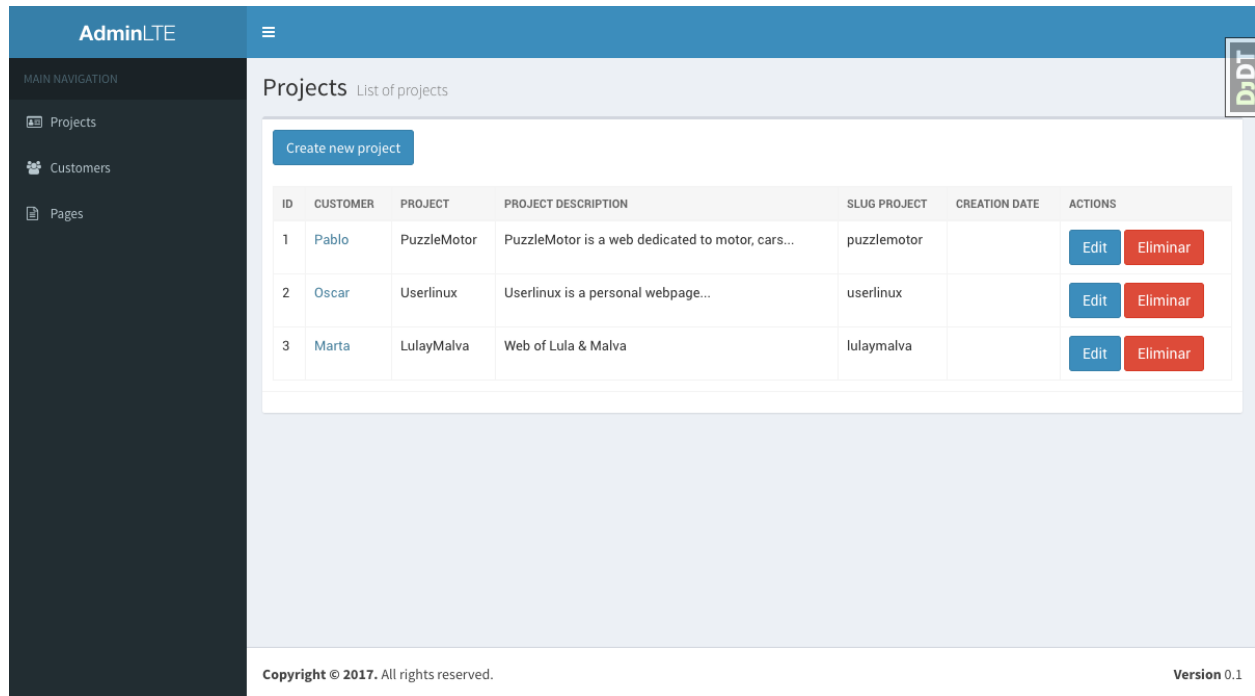
1.2 Features

- Responsive design: `django-adminlte2`
- Tabbed forms: `django-crispy-forms`
- Wrappable fields: `django-crispy-forms`
- Image cropping: `django-image-cropping` (custom widget)
- something for `select2` (custom widget)
- something for other file types (upload, multiple upload, date, time, color etc...) (custom widgets)
- Reordering: `django-mptt`
- Easy to understand/adapt: A cruds mixin with CBV was a good idea, I've found <https://github.com/bmihelac/django-cruds> and it rang the definitely bell here
- Easy to extend (anyone could contribute with new widgets or behaviors, inlines, search, filters...)

1.3 Online Resources

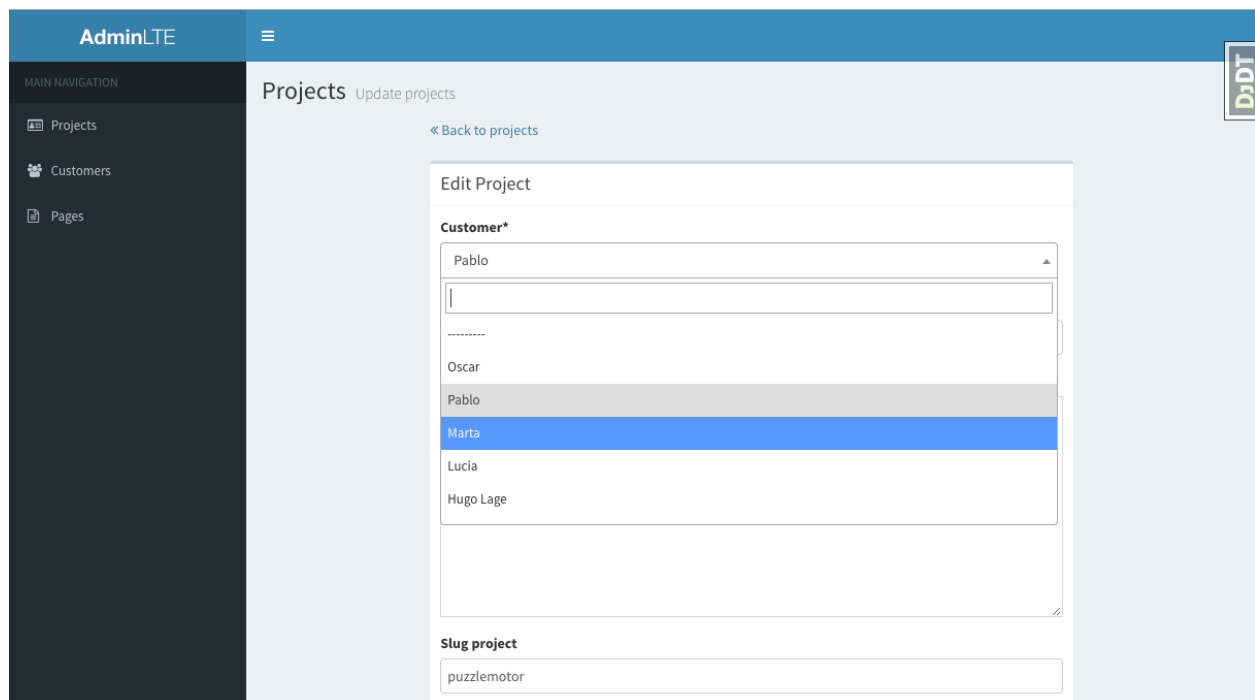
- [Code repository](#)
- [Documentation](#)
- [Pypi](#)
- [DjangoPackages](#)
- For reporting a bug use [GitHub Issues](#)

1.4 Screenshots



The screenshot shows the 'Projects' list view in the AdminLTE interface. The left sidebar contains the 'AdminLTE' logo and a 'MAIN NAVIGATION' menu with links to 'Projects', 'Customers', and 'Pages'. The main content area has a blue header with the 'Projects' title and a subtitle 'List of projects'. Below the header is a 'Create new project' button. A table lists three projects with columns for ID, Customer, Project, Project Description, Slug Project, Creation Date, and Actions. The 'Actions' column contains 'Edit' and 'Eliminar' buttons for each project. The footer shows 'Copyright © 2017. All rights reserved.' and 'Version 0.1'.

ID	CUSTOMER	PROJECT	PROJECT DESCRIPTION	SLUG PROJECT	CREATION DATE	ACTIONS
1	Pablo	PuzzleMotor	PuzzleMotor is a web dedicated to motor, cars...	puzzlemotor		Edit Eliminar
2	Oscar	Userlinux	Userlinux is a personal webpage...	userlinux		Edit Eliminar
3	Marta	LulayMalva	Web of Lula & Malva	lulaymalva		Edit Eliminar



The screenshot shows the 'Edit Project' form in the AdminLTE interface. The left sidebar is the same as the previous screenshot. The main content area has a blue header with the 'Projects' title and a subtitle 'Update projects'. Below the header is a '« Back to projects' link. The form is titled 'Edit Project' and contains two sections: 'Customer*' and 'Slug project'. The 'Customer*' section has a dropdown menu with 'Pablo' selected, and a list of other customers: Oscar, Pablo, Marta, Lucia, and Hugo Lage. The 'Slug project' section has a text input field containing 'puzzlemotor'.

AdminLTE

MAIN NAVIGATION

Projects

Customers

Pages

Customers

Update customers

« Back to customers

Basic information

Other information

Customer*

Oscar

Address*

Lugo

Correo electrónico

email@example.org

Submit

Eliminar

Copyright © 2017. All rights reserved.

Version 0.1

AdminLTE

MAIN NAVIGATION

Projects

Customers

Pages

Customers

Update customers

« Back to customers

Basic information

Other information

Image

Actualmente: media/customers/slider01.png

Limpiar

Modificar:

Seleccionar archivo

Ningún archivo seleccionado

CIF/NIF

33322145

Cropping

Te gustaría ser campeón/a en el deporte y en la vida?



Slug for invoicing

oscar

Submit

Eliminar

Copyright © 2017. All rights reserved.

Version 0.1

6

Chapter 1. django-cruds-adminlte

[<< Back to invoices](#)

Edit Invoice

Invoice

Email

Customer*

Customer object

☒ Registered
Registered yet?

☐ Sent
Invoice sent?

☒ Paid
Invoice paid?

Creation date*

April 30, 2017, midnight

Invoice Number*

1

Subtotal

200

Calculated field

Subtotal IVA

20

Calculated field

Subtotal Retentions

10

Calculated field

Total

210

Calculated field

Submit

Delete

Lines

ID	INVOICE	REFERENCE	CONCEPT	QUANTITY	UNIT	UNIT PRICE	AMMOUNT	ACTIONS
#1	1->2017-04-30 00:00:00+00:00	1	Systems	2	days	30	60	Edit Delete

Create

[<< Back to customers](#)

Edit Customer

Basic information

Other information

Customer*

Address*

Text

Formato ▾

B ***I***
| ↶ ↷
 | ✂
 | 📄
 | 📋
 | ↕
 | ⌵ ⌶
 | ⌵ ⌶
 | ⋮
 | 🖼
 | 📅
 | 🔗

🔗 Fuente HTML

☒ Status

Submit

Eliminar

The screenshot displays the AdminLTE dashboard. The left sidebar contains a 'MAIN NAVIGATION' section with links for Authors, Addresses, Customers, and Invoices (which is highlighted). Below this is an 'AUTH' section with links for Users, Groups, and Permissions. The main content area is titled 'Invoices' with a subtitle 'List of invoices'. It features a 'Create new invoice' button and a table with the following data:

CUSTOMER	REGISTERED	SENT	PAID	CREATION DATE
Customer object	☒	☐	☒	April 30, 2017, midnight
Customer object	☒	☐	☐	May 1, 2017, midnight

AdminLTE

Pages

Update pages

« Back to pages

Edit Page

Title*

Historia

Page text

Historia page

Parent

Submit

Eliminar

2.1 Installation

2.1.1 Quickstart

Install `django-cruds-adminlte` (already in Pypi):

```
pip install django-cruds-adminlte
```

Then use it in a project, add `cruds_adminlte` to `INSTALLED_APPS`. Note that you will have to install `crispy_forms` and `image_cropping` if before the app if you want to use them:

```
pip install django-crispy-forms
pip install easy-thumbnails
pip install django-image-cropping
pip install djangoajax
```

Next step is to add the urls to your `project.urls` as was said above:

```
# django-cruds-adminlte
from cruds_adminlte.urls import crud_for_app
urlpatterns += crud_for_app('testapp')
```

And you can start modeling your app, migrate it and directly browse to the urls described above, that's all.

2.1.2 Requirements

The `django-cruds-adminlte` works thanks to:

- Python 3.7+
- Django >=2.2
- django-crispy-forms

- django-image-cropping and easy-thumbnails (optional if you want to crop)
- djangoajax (for the inlines stuff)

If you want full support then install dependencies make sure to install these packages prior to installation in your environment:

```
pip install django-crispy-forms
pip install django-select2
pip install django-image-cropping
pip install easy-thumbnails
pip install djangoajax
```

2.1.3 Getting the code

For the latest stable version of django-cruds-adminlte use **pip**:

```
$ pip install django-cruds-adminlte
```

You could also retrieve the last sources from <https://github.com/oscarmlage/django-cruds-adminlte>. Clone the repository using **git** and run the installation script:

```
$ git clone git://github.com/oscarmlage/django-cruds-adminlte.git
$ cd django-cruds-adminlte
$ python setup.py install
```

or more easily via **pip**:

```
$ pip install -e git://github.com/oscarmlage/django-cruds-adminlte.git
```

2.1.4 Applications

Assuming that you have an already existing Django project, register `cruds_adminlte` in the `INSTALLED_APPS` section of your project's settings:

```
INSTALLED_APPS = (
    ...
    'crispy_forms',
    'django_select2',
    'easy_thumbnails',
    'image_cropping',
    'django_ajax',
    'cruds_adminlte'
)
```

2.1.5 Configuration

Configure template pack and jquery for `image_cropping`. Note: Template also import jquery so it's not necessary import custom `IMAGE_CROPPING_JQUERY_URL`:

```
CRISPY_TEMPLATE_PACK = 'bootstrap3'
IMAGE_CROPPING_JQUERY_URL = None
```

Configure internal IPs:

```
INTERNAL_IPS = ('127.0.0.1',)
```

Configure easy_thumbnails:

```
from easy_thumbnails.conf import Settings as thumbnail_settings
THUMBNAI_PROCESSORS = (
    'image_cropping.thumbnail_processors.crop_corners',
) + thumbnail_settings.THUMBNAI_PROCESSORS
```

Configure the default time and datetime:

```
TIME_FORMAT= 'h:i A'
DATETIME_FORMAT='m/d/Y H:i:s'
DATE_FORMAT="m/d/Y"

TIME_INPUT_FORMATS = ['%I:%M %p']
```

Warning: Datetime and time depends on *USE_TZ* attribute, so changes there impact in all django timezone management

2.1.6 URLs for the CRUD

To add CRUD for whole app, add this to `urls.py`:

```
# django-cruds-adminlte
from cruds_adminlte.urls import crud_for_app
urlpatterns += crud_for_app('testapp')
```

This will create following urls and appropriate views (assuming there is a application named `testapp` with model `Author`):

URL	name
/testapp/author/list/	testapp_author_list
/testapp/author/new/	testapp_author_create
/testapp/author/(?P<pk>d+)	testapp_author_detail
/testapp/author/(?P<pk>d+)/update/	testapp_author_update
/testapp/author/(?P<pk>d+)/delete/	testapp_author_delete

It is also possible to add CRUD for one model:

```
from django.apps import apps
from cruds_adminlte.urls import crud_for_model
urlpatterns += crud_for_model(apps.get_model('testapp', 'Author'))
```

2.1.7 cruds_for_app

Parameters you can set in `cruds_for_app` method call:

- `login_required` (boolean): Check if the login is required, need to activate 'login' and 'logout' urls, for example:

```
url(r'^accounts/login/$', auth_views.login, name='login'),
url(r'^logout/$', auth_views.logout, name='logout'),
```

- `check_perms` (boolean): Check if the user has the proper permissions.
- `cruds_url` (string): Put all the generated cruds in a common url, instead of 'app_one/model/list' and 'app_two/model/list' we can set it to 'myadmin' and then the urls will be 'myadmin/app_one/model/list' and 'myadmin/app_two/model/list'.
- `modelforms`: Load custom forms for the cruds of the model.

Different samples:

```
urlpatterns += crud_for_app('app_one', login_required=True,
                           check_perms=True, cruds_url='myadmin')
urlpatterns += crud_for_app('app_two', login_required=False,
                           check_perms=True, cruds_url='myadmin')

from testapp.forms import CustomerForm, InvoiceForm
custom_forms = {
    'add_customer': CustomerForm,
    'update_customer': CustomerForm,
    'add_invoice': InvoiceForm,
    'update_invoice': InvoiceForm,
}
urlpatterns += crud_for_app('app_three', login_required=True,
                           check_perms=True, cruds_url='myadmin',
                           modelforms=custom_forms)
```

Templates and templatetags

3.1 Templates and templatetags

3.1.1 Templates

django-cruds-adminlte views will append CRUD template name to a list of default candidate template names for given action.

CRUD Templates are:

```
cruds/create.html
cruds/delete.html
cruds/detail.html
cruds/list.html
cruds/update.html
```

Templates are based in [AdminLTE2](#) and [django-adminlte2](#) (but this last is not required because the templates are included in this project). They're ready to run with:

- [django-crispy-forms](#)
- [select2](#)
- [django-cropping-image](#)

You will probably want to override some templates, check the `TEMPLATES` config in your settings file and ensure you have your custom templates dir in `DIRS`:

```
TEMPLATES = [
    {
        'BACKEND': ...,
        'DIRS': [normpath(join(dirname(dirname(abspath(__file__))),
                                'demo', 'templates'))],
        ...
    }
```

(continues on next page)

(continued from previous page)

```
}  
]
```

If you want to override the sidebar you can do it creating a file called `templates/adminlte/lib/_main_sidebar.html` inside your project and you can put there the contents you want.

3.1.2 Templatetags

`crud_fields` templatetag displays fields for an object:

```
{% load crud_tags %}  
  
<table class="table">  
  <tbody>  
    {% crud_fields object "name, description" %}  
  </tbody>  
</table>
```

Use `cruds_adminlte.util.crud_url` shortcut function to quickly get url for instance for given action:

```
crud_url(author, 'update')
```

Is same as:

```
reverse('testapp_author_update', kwargs={'pk': author.pk})
```

CHAPTER 4

Components

4.1 Components

4.1.1 Columns

If you take a look to the directory “templates/cruds/columns” you can see the different kinds of columns depending on the type of field:

```
autofield.html
booleanfield.html
charfield.html
datefield.html
datetimefield.html
filefield.html
textfield.html
timefield.html
```

You can override the column type in lists pages with the custom html you want for your project. Just recreate the structure (templates/cruds/columns/) in your project and write your own html.

Create new customer

ID	CUSTOMER	ADDRESS	CIF/NIF	CORREO ELECTRÓNICO	SLUG FOR INVOICING	IMAGE	CROPPING	DATETIME	DATE	TIEMPO	COLOR	DESCRIPTION	STATUS
#1	Oscar	Lugo	11111111H	xxx@xx.xx	oscarmlage	WhatsApp_Image_2017-03-14_at_18.34.29.jpeg - crop	128,187,561,551				#d12a7a	This is how we do it, baby! Oh baby	✓
#2	Pablo	Coruña	33322112	xxx@xx.xx	pablo	slider02.png - crop	541,0,970,359				#FFFFFF		✓
#3	Marta	Roca	23rr32	xxx@xx.xx	marta	slider04.png - crop	218,0,562,288				#FFFFFF		✓

4.1.2 Forms

If you want to override a form with some other crispy features you can add to your testapp.urls the following:

```
from cruds_adminlte.urls import crud_for_model
urlpatterns += crud_for_model(Author, views=['create', 'update'],
                             add_form=AuthorForm, update_form=AuthorForm )
```

And define the AuthorForm with tabs or any other crispy feature in your app:

```
self.helper.layout = Layout(
    TabHolder(
        Tab(
            _('Basic information'),
            Field('name', wrapper_class="col-md-6"),
            Field('address', wrapper_class="col-md-6"),
            Field('email', wrapper_class="col-md-12"),
        ),
        Tab(
            _('Other information'),
            Field('image', wrapper_class="col-md-6"),
            Field('cropping', wrapper_class="col-md-6"),
            Field('cif', wrapper_class="col-md-6"),
            Field('slug', wrapper_class="col-md-6")
        )
    )
)
```

You will get something similar to this:

« Back to customers

Edit Customer

Basic information Other information

Customer* **Address***

Oscar Lugo

Correo electrónico

oscar@enlugo.com

Submit Eliminar

Crispy tabbed form sample

forms.py:

```
class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'image', 'cropping']
        widgets = {
            'image': ImageCropWidget,
        }

    def __init__(self, *args, **kwargs):
        super(CustomerForm, self).__init__(*args, **kwargs)
        self.helper = FormHelper(self)

        self.helper.layout = Layout(
            TabHolder(
                Tab(
                    _('Basic information'),
                    Field('name', wrapper_class="col-md-6"),
                    Field('address', wrapper_class="col-md-6"),
                    Field('email', wrapper_class="col-md-12"),
                ),
                Tab(
                    _('Other information'),
                    Field('image', wrapper_class="col-md-6"),
                    Field('cropping', wrapper_class="col-md-6"),
                    Field('cif', wrapper_class="col-md-6"),
                    Field('slug', wrapper_class="col-md-6")
                )
            )
        )

        self.helper.layout.append(
            FormActions(
                Submit('submit', _('Submit'), css_class='btn btn-primary'),
                HTML("""{% load i18n %}<a class="btn btn-danger"
                    href="{% url_delete %}">{% trans 'Delete' %}</a>"""),
            )
        )
```

4.1.3 Fields

Cropping widget

models.py:

```
from image_cropping import ImageCropField, ImageRatioField
class Customer(models.Model):
    name = models.CharField(_('Customer'), max_length=200)
    image = ImageCropField(upload_to='media/customers', blank=True)
    cropping = ImageRatioField('image', '430x360')
```

forms.py:

```
class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'image', 'cropping']
        widgets = {
            'image': ImageCropWidget,
        }
```

Select2 widget

By default all the select are automatically converted in select2.

DatePicker widget

forms.py:

```
from cruds_adminlte import DatePickerWidget

class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'date']
        widgets = {
            'date': DatePickerWidget(attrs={'format': 'mm/dd/yyyy',
                                           'icon': 'fa-calendar'}),
        }
```

[« Back to customers](#)

Edit Customer

Basic information

Other information

Color

#2031c7 

Datetime

 03/29/2017 05:25:54

Time

 07:15 PM

Date



« MARCH 2017 »

SU	MO	TU	WE	TH	FR	SA
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Submit

Eliminar

TimePicker widget

forms.py:

```
from cruds_adminlte import TimePickerWidget

class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'time']
        widgets = {
            'time': TimePickerWidget(attrs={'icon': 'fa-clock-o'}),
        }
```

[« Back to customers](#)

Edit Customer

[Basic information](#)
[Other information](#)

Color



Datetime



Time



^

07

:

15

^

PM

v

:

v

Date



[Submit](#)
[Eliminar](#)

DateTimePicker widget

forms.py:

```
from cruds_adminlte import DateTimePickerWidget

class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'datetime']
        widgets = {
            'datetime': DateTimePickerWidget(attrs={'format': 'mm/dd/yyyy HH:ii:ss',
                                                    'icon': 'fa-calendar'}),
        }
```

« Back to customers

Edit Customer

Basic information
Other information

Color
 

Datetime

16 MARCH 2017

0:00	1:00	2:00	3:00
4:00	5:00	6:00	7:00
8:00	9:00	10:00	11:00
12:00	13:00	14:00	15:00
16:00	17:00	18:00	19:00
20:00	21:00	22:00	23:00

Time

Date

Submit
Eliminar

ColorPicker widget

forms.py:

```
from cruds_adminlte import ColorPickerWidget

class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'color']
        widgets = {
            'color': ColorPickerWidget,
        }
```

« Back to customers

Edit Customer

Basic information
Other information

Color

Time

Datetime

Date

CKEditor widget

forms.py:

```
from cruds_adminlte import CKEditorWidget

class CustomerForm(forms.ModelForm):

    class Meta:
        model = Customer
        fields = ['name', 'text']
        widgets = {
            'text': CKEditorWidget(attrs={'lang': 'es'}),
        }
```

4.2.1 Using CRUDView

How to use:

In your views file create a class inherit for CRUDView

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView
class Myclass(CRUDView):
    model = Customer
```

In urls.py

```
myview = Myclass()
urlpatterns = [
    url('path', include(myview.get_urls())) # also support
                                           # namespace
]
```

If you want to filter views add views_available list

```
class Myclass(CRUDView):
    model = Customer
    views_available=['create', 'list', 'delete', 'update', 'detail']
```

4.2.2 Permissions

The default behavior is check_login = True and check_perms=True but you can turn off with

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView

class Myclass(CRUDView):
    model = Customer
    check_login = False
    check_perms = False
```

You also can defined extra perms in two ways as django perm string or like a function

```
def myperm_system(user, view):
    # user is django user
    # view is one of this 'list', 'add', 'update', 'detail'
    return True or False

class Myclass(CRUDView):
    model = Customer
    perms = { 'create': ['applabel.mycustom_perm'],
              'list': [],
              'delete': [myperm_system],
              'update': [],
              'detail': []
            }
```

If check_perms = True we will add default django model perms (<applabel>.[add|change|delete|view]_<model>) ej. mytestapp.add_mymodel

Warning: applabel.view_model are not part of django perms, so needs to be create in models metadata ej.

```
class Autor(models.Model):
    name = models.CharField(max_length=200)
    class Meta:
        ordering = ('pk',)
        permissions = (
            ("view_author", "Can see available Authors"),
        )
```

apppabel.view_model is used by default for list perm, so if it's not created then list view raise 503 permission denied (with screen in browser)

4.2.3 Searching

As django admin does, **search_fields** are available, and you can filter using double underscore (__) to search across the objects.

split_space_search split search text in parts using the string provided, this can be usefull to have better results but have impact in search performance, if split_space_search is True then ' ' is used

```
class Myclass(CRUDView):
    model = Customer
    search_fields = ['description__icontains']
    split_space_search = ' ' # default False
```

Note: 'icontains' is not set by default as django admin does, so you need to set if not equal search is wanted

Invoices List of invoices

[Create new invoice](#)

CUSTOMER	REGISTERED	SENT	PAID	CREATION DATE	INVOICE NUMBER	DESCRIPTION HEADER	DESCRIPTION FOOTER	SUBTOTAL	SUBTOTAL IVA	SUBTOTAL RETENTIONS	TOTAL	ACTIONS
Customer object	☒	☐	☒	April 27, 2017, midnight	1	Header	Footer	200	20	10	210	Show
Customer object	☒	☒	☒	April 12, 2017, midnight	3	Esto es un ejemplo Esto va en negrita	esto es un ejemplo y va subrayado <i>hola mundo</i>					Show

4.2.4 Filter content

Warning: Code preserve filter it's a complex task, and filter content with high grade of liberty is hard to do, so this is a experimental version.

Use **list_filter** as list of model attributes or FormFilter objects like:

```
class Myclass (CRUDView):
    model = Invoice
    list_filter = ['invoice_number', 'sent', 'paid']
```

Filter method is based on forms and filter query set, so we use different approach compared with django admin

FormFilter is a special class used for filter content based on form.

```
from cruds_adminlte.filter import FormFilter
class LineForm(forms.Form):
    line = forms.ModelMultipleChoiceField(queryset=Line.objects.all())

class LineFilter(FormFilter):
    form = LineForm

class Myclass (CRUDView):
    model = Invoice
    list_filter = ['sent', 'paid', LineFilter]
```

Magic..., not, just and good example of how to do a multiple value search based end a reverse foreignkey.

FormFilter has this public method:

- **render():** return a form or your own html, has an instance of form in self.form_instance, and also has self.request.
- **get_filter(queryset):** filter your content here
- **get_params(exclude):** clean the get parameters

4.2.5 Pagination

Pagination is supported for list view using **paginate_by** and **paginate_template**, the default pagination value is:

- **paginate_by** = 10
- **paginate_template** = 'cruds/pagination/prev_next.html'
- **paginate_position** = 'Bottom'

For example paginate customers using enumeration paginate

```
class Myclass (CRUDView):
    model = Customer
    paginate_by = 5
    paginate_template = 'cruds/pagination/enumeration.html'
    paginate_position = 'Both'
```

The **paginate_position** options are *Bottom*, *Both*, *Up*

4.2.6 Overwrite forms

You can also overwrite add and update forms

```
class Myclass (CRUDView):
    model = Customer
    add_form = MyFormClass
    update_form = MyFormClass
```

4.2.7 Overwrite templates

And of course overwrite base template name

```
class Myclass(CRUDView):
    model = Customer
    template_name_base = "mybase"
```

Remember basename is generated like app_label/modelname if template_name_base is set as None add 'cruds' by default so template loader search this structure

```
basename + '/create.html'
basename + '/detail.html'
basename + '/update.html'
basename + '/list.html'
basename + '/delete.html'
```

Note: Also import <applabel>/<model>/<basename>/<view type>.html

4.2.8 Using namespace

There is no way to create 2 CRUDView to the same model, because urls could be crash, so namespace come to help with this, *namespace* are part of django urls system and allows to have same urls with diferent context, so you can use this to add different behavior to a model, also different urls.

In views

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView
class Myclass(CRUDView):
    model = Customer
    namespace = "mynamespace"
```

In urls.py

```
myview = Myclass()
urlpatterns = [
    url('path', include(myview.get_urls(),
                        namespace="mynamespace"))
]
```

Namespace in views and urls needs to match, or url match problem are raise.

4.2.9 Related fields

A common scenario is that you have a model with a foreignkey to other model that is the main of your view so you want to pass the main model as parameter to a crud views to filter and create using it as main reference, and always save the foreignkey with the main model object.

For example In models

```
class Author(models.Model):
    name=models.CharField(max_length=150)
class Book(models.Model):
    author = models.ForeignKey(Author):
    name=models.CharField(max_length=150)
```

In views

```
from cruds_adminlte.crud import CRUDView
class Myclass(CRUDView):
    model = Book
    related_fields = ['autor']
```

So with this you now have management of author's book.

Warning: we provide all internal references but you need to create the first author to book list/create/update/detail/delete reference.

4.2.10 Decorators

CRUDViews use a generic Django views and provide some utilities to manage decorator. As django documentation say you can use decorator in urls when you call `as_view` method in generic views like.

In `urls.py`

```
urlpatterns = [
    url('list', login_required(ListView.as_view()) )
]
```

CRUDViews take advantage of this and create this methods

- `decorator_create(self, viewclass)`
- `decorator_detail(self, viewclass)`
- `decorator_list(self, viewclass)`
- `decorator_update(self, viewclass)`
- `decorator_delete(self, viewclass)`

So you can overwrite it and put your own decorator. Be warried about `login_required` decorator, because when `check_login` is set we used this method to insert `login_required` decorator.

How to overwrite:

In views

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView
class Myclass(CRUDView):
    model = Customer
    def decorator_list(self, viewclass):
        viewclass = super(Myclass, self).decorator_list(viewclass) # help with
                                                                    # login_required
    return mydecorator(viewclass)
```

4.2.11 Overwrite views

Overwrite views are easy because we are using django generic views, but you need to have some worry.

If you don't need to overwrite this functions

- `get_template_names`
- `get_context_data`
- `dispatch`
- `paginate_by` attr in list view

then you can overwrite and return your own class

- `get_create_view_class`
- `get_update_view_class`
- `get_detail_view_class`
- `get_list_view_class`
- `get_delete_view_class`

but if you need to overwrite some of the above functions you need to overwrite

- `get_create_view`
- `get_update_view`
- `get_detail_view`
- `get_list_view`
- `get_delete_view`

Like

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView
class Myclass(CRUDView):
    model = Customer
    def get_list_view(self):
        ListViewClass = super(Myclass, self).get_list_view()
        class MyListView(ListViewClass):
            def get_context_data(self):
                context = super(MyListView, self).get_context_data()
                return context
        return MyListView
```

Warning: It's really important that you use `super(MyListView, self).get_context_data()` instead of `ListView.get_context_data()` because we insert some extra context there.

4.3 UserCRUDView Usage

A usefull utility class is provided named as UserCRUDView, and works link CRUDView but include user management, but require than base model has user attribute.

In Create and Update view save the model adding current user as user attribute. In List View filter objects using current user.

In models

```
from django.contrib.auth.models import User
from django.db import models
class Customer(models.Model):
    user = models.ForeignKey(User)
    ...
```

In views

```
from testapp.models import Customer
from cruds_adminlte.crud import CRUDView
class Myclass(UserCRUDView):
    model = Customer
```

4.4 InlineAjaxCRUD Usage

Inlines works like django admin inlines but with some differences, firsts use django-ajax for provide a crud view, and second not inlines in create view (sorry for now we need model created to have pk reference).

Basically works like CRUDView and support all cases described above. Require this extra parameters

1. *base_model* model used to refence the inline
2. *inline_field* field used to update object, needs to be the same class that *base_model*
3. *title* title of the inline (used to show separation between model fields and inline fields).

```
class Address_AjaxCRUD(InlineAjaxCRUD):
    model = Addresses
    base_model = Autor
    inline_field = 'autor'
    fields = ['address', 'city']
    title = _("Addresses")

class AutorCRUD(CRUDView):
    model = Autor
    inlines = [Address_AjaxCRUD]
```

4.5 CRUDMixin Usage

CRUDMixin is a mixin-like class that all the views inherit from. It provides a convenient way of customizing your views, requiring of no additional changes. You can access that class when calling the functions “crud_for_app” or “crud_for_models”, passing the reference to your custom CRUDMixin object as a new parameter to any of these functions.

The following example uses the class “MyMixin” to customize the object called “context_data” for all the views. This way, all the templates will have a new object called “cars” available.

```
class MyMixin(CRUDMixin):
    def get_context_data(self, *args, **kwargs):
```

(continues on next page)

(continued from previous page)

```

context = super(Mixin, self).get_context_data(*args, **kwargs)
context['cars'] = MyModel.objects.all()
return context

urlpatterns += crud_for_app('myapp', login_required=True, mixin=MyMixin)

```

Warning: The class “MyMixin” needs to inherit from “CRUDMixin”; otherwise an exception is raised.

4.6 Demo / testing project

A full demonstration project is included with django-cruds-adminlte’s which you can use as a companion “cookbook”. The project is used intensively for the testsuite and/or to generate screenshots for the documentation.

You might find the demo project in the `tests/test_project` folder of the source tree.

- *Prepare the environment*
- *Run the demo*
- *Test user*

4.6.1 Prepare the environment

The demo project can be run using the same python environment used for development as it’s dependencies are the same.

If you have not setup such environment you can either:

- Refer to the contributing documentation.
- Setup the environment as detailed here.

The main difference is that the detailed instructions on the contributing documentation are meant for proper development, testing and patch submission. The instructions here are meant as a quick solution to run the demo.

```

# From the root directory of the project
# (the one with a setup.py file).
$ python -m venv .venv
$ source .venv/bin/activate
(venv) $ pip install -r requirements-test.txt

```

Do not forget to enable the virtual environment when using the demo

```

$ source .venv/bin/activate
(venv) $

```

4.6.2 Run the demo

The demo is, pretty much, a standard django project so you can run it like you would run any other project:

```
(venv)$ ./tests/demo_project/manage.py migrate
(venv)$ ./tests/demo_project/manage.py createdemodata
(venv)$ ./tests/demo_project/manage.py runserver --settings=demo_project.settings 0.0.
→0.0:4444
Starting development server at http://0.0.0.0:4444/
Quit the server with CONTROL-C.
```

4.6.3 Test user

If you want to make the most of the demo, you should really create a test user to access it.

```
(venv)$ ./tests/demo_project/manage.py createsuperuser
Username (leave blank to use 'admin'): admin
Email address: admin@example.com
Password:
Password (again):
Superuser created successfully.
```

CHAPTER 5

Screenshots

5.1 Screenshots

The screenshot displays the 'Projects' management interface in AdminLTE. It features a sidebar with navigation links for 'Projects', 'Customers', and 'Pages'. The main content area shows a table of projects with columns for ID, Customer, Project, Project Description, Slug Project, Creation Date, and Actions. There are three projects listed: 'PuzzleMotor' by 'Pablo', 'Userlinux' by 'Oscar', and 'LulayMalva' by 'Marta'. Each project has 'Edit' and 'Eliminar' buttons. A 'Create new project' button is also present. The footer includes the copyright notice 'Copyright © 2017. All rights reserved.' and the version 'Version 0.1'.

ID	CUSTOMER	PROJECT	PROJECT DESCRIPTION	SLUG PROJECT	CREATION DATE	ACTIONS
1	Pablo	PuzzleMotor	PuzzleMotor is a web dedicated to motor, cars...	puzzlemotor		Edit Eliminar
2	Oscar	Userlinux	Userlinux is a personal webpage...	userlinux		Edit Eliminar
3	Marta	LulayMalva	Web of Lula & Malva	lulaymalva		Edit Eliminar

AdminLTE

MAIN NAVIGATION

Projects
Customers
Pages

Projects
Update projects

« Back to projects

Edit Project

Customer*

Pablo

Oscar

Pablo

Marta

Lucia

Hugo Lage

Slug project

puzzlemotor

AdminLTE

MAIN NAVIGATION

Projects
Customers
Pages

Customers
Update customers

« Back to customers

Edit Customer

Basic information
Other information

Customer*

Oscar

Address*

Lugo

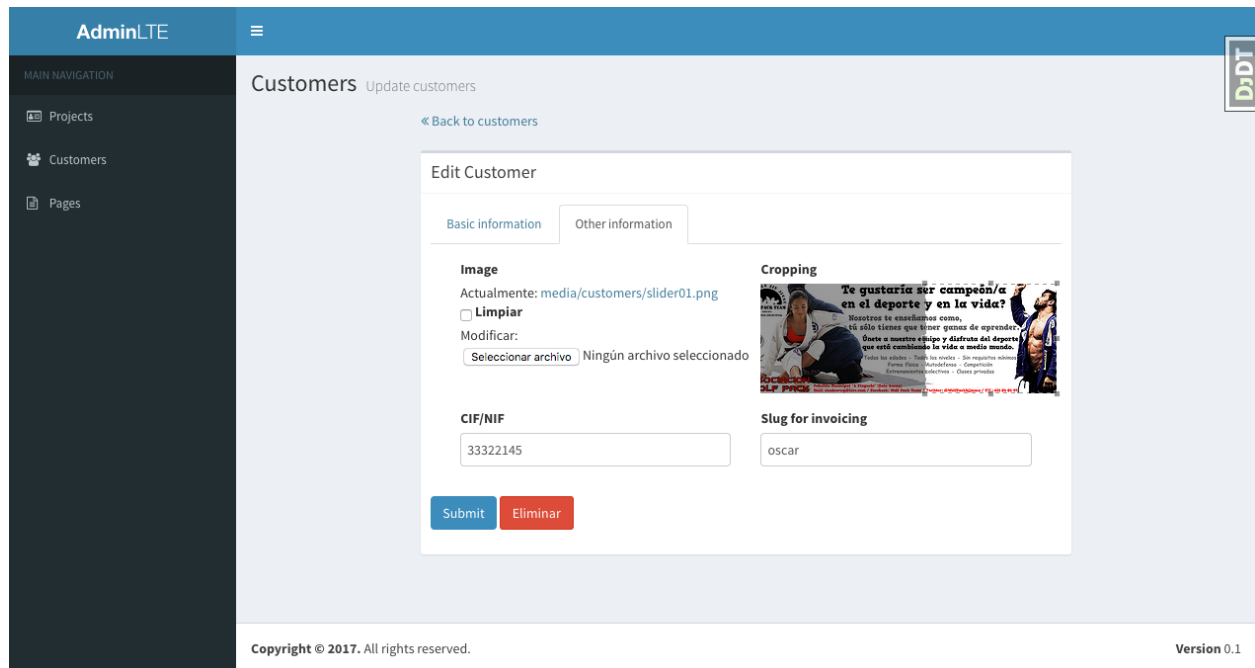
Correo electrónico

email@example.org

Submit
Eliminar

Copyright © 2017. All rights reserved.

Version 0.1



[<< Back to invoices](#)

Edit Invoice

Invoice

Email

Customer*

Customer object

☒ Registered
Registered yet?

☐ Sent
Invoice sent?

☒ Paid
Invoice paid?

Creation date*

April 30, 2017, midnight

Invoice Number*

1

Subtotal

200

Calculated field

Subtotal IVA

20

Calculated field

Subtotal Retentions

10

Calculated field

Total

210

Calculated field

Submit

Delete

Lines

ID	INVOICE	REFERENCE	CONCEPT	QUANTITY	UNIT	UNIT PRICE	AMMOUNT	ACTIONS
#1	1->2017-04-30 00:00:00+00:00	1	Systems	2	days	30	60	Edit Delete

Create

Invoices

List of invoices

Create new invoice

Search for...

CUSTOMER	REGISTERED	SENT	PAID	CREATION DATE	INVOICE NUMBER	DESCRIPTION HEADER	DESCRIPTION FOOTER	SUBTOTAL	SUBTOTAL IVA	SUBTOTAL RETENTIONS	TOTAL	ACTIONS
Customer object	☒	☐	☒	April 27, 2017, midnight	1	Header	Footer	200	20	10	210	Show
Customer object	☒	☒	☒	April 12, 2017, midnight	3	Esto es un ejemplo	esto es un ejemplo y va subrayado					Show
						Esto va en negrita	hola mundo					

AdminLTE

MAIN NAVIGATION

Authors

Addresses

Customers

Invoices

AUTH

Users

Groups

Permissions

Invoices

List of invoices

Create new invoice

CUSTOMER	REGISTERED	SENT	PAID	CREATION DATE
Customer object	☒	☐	☒	April 30, 2017, midnight
Customer object	☒	☐	☐	May 1, 2017, midnight

AdminLTE

Pages

Update pages

« Back to pages

Edit Page

Title*

Historia

Page text

Historia page

Parent

Submit

Eliminar

6.1 Thanks

django-cruds-adminlte cannot be a great application without great contributors who make this project greatest each day.

- Óscar M. Lage <info@oscarmlage.com>
- Luis Zárate <luisza14@gmail.com>
- Miguel Arguedas <migue.arguedas.mejias@gmail.com>
- Marc Bourqui
- Bojan Mihelac <bmihelac@mihelac.org>
- Alejandro Gamboa Barahona <<https://github.com/alegambo>>
- miguelmendez17 <<https://github.com/miguelmendez17>>
- Ricardo Tubio-Pardavila <<https://github.com/rtubio>>
- Marc Fargas (telenieko) <telenieko@telenieko.com>

And also a special thank to [GitHub.com](https://github.com), and ReadTheDocs.org for their services of great quality.

7.1 Changelog

7.1.1 0.0.17

- “082ac1b” Revert “Add getparams for search parameters”
- “39d10d0” Merge pull request #109 from BnFSoftware/master
- “4ea8ea3” Add getparams for search parameters
- “2c128b3” Merge pull request #108 from telenieko/tests-behold-circleci
- “a31627c” Merge pull request #107 from telenieko/tests-behold
- “323cebe” Merge branch ‘tests-behold’ into tests-behold-circleci
- “3b83177” Bring python3.5 back to tox.ini
- “6a547a4” Merge branch ‘tests-behold’ into tests-behold-circleci
- “442c892” Update CONTRIBUTING.rst to explain how to run tests.
- “6186cca” Merge branch ‘master’ into tests-behold-circleci
- “160f96d” Merge branch ‘tests-behold’ into tests-behold-circleci
- “768b1d8” More on testing (tox fixes)
- “f033e5f” circleci says that should be directory, not a file path
- “3e9126d” Updates to tox and circleci.
- “7be1239” Revert “idea files of pycharm (shared ones)”
- “bf5d8a5” idea files of pycharm (shared ones)
- “f48b980” Merge branch ‘tests-behold-circleci’
- “08a0482” Merge remote-tracking branch ‘oscarmlage/master’

- “07e0dc0” continue with circleci
- “030c756” Change circleci store_artifacts
- “149fb75” Merge branch ‘add-circleci-support’ into tests-behold-circleci
- “e78aff8” Merge remote-tracking branch ‘oscarmlage/master’ into tests-behold
- “79f84f7” Refactor of test infrastructure.
- “703be9d” Added Marc as collaborator
- “ff2f6aa” Merge remote-tracking branch ‘oscarmlage/master’
- “15b1664” Merge pull request #104 from telenieko/changes-to-tox
- “bcc913e” Merge branch ‘python3’ into changes-to-tox
- “b70e04d” Merge pull request #106 from telenieko/versioning
- “6664f0c” Use git commit for versioning builds
- “b30765b” Remove blank line from VERSION file when reading
- “a8a2bff” Merge branch ‘add-circleci-support’
- “579ee74” Merge branch ‘changes-to-tox’ into add-circleci-support
- “03ade52” Closes #103, Make python 3.7+ and django 2.2+ the minimal reqs.
- “575ad84” Add .circleci
- “baac21c” ci fixes
- “0a02332” Remove py36 from tox.
- “d39ac4c” Remove py36 from tox.
- “54162fd” ci fixes
- “90965ea” Rename circleci workflow
- “7d3e4fe” circleci: install tox and coverage
- “78296d7” Merge tox + circleci into master.
- “c8860d7” coverage working on tox
- “4011c85” Add .eggs/ to .gitignore
- “fc68e3b” Refs #103, change pypi classifiers regarding python versions.
- “5f90844” Properly set html_static_path in sphinx
- “5fe3e0b” Fix syntax error in docs/usage.rst
- “222f513” Fixes to tox.ini
- “d20e8a4” Improve requirements-test.txt
- “cd40ae8” Reorg dependencies.
- “adafaa0” Fix indentation error in docs/installation.rst
- “678a06c” Force call to sys.exit() on run_tests
- “6c61ec8” Fix test_views which would never pass previously
- “8ffd335” Fix (temporary) conflict with mixin and mymixin in NOCLASS that gives a metaclass TypeError that makes unusable the crud

- “e42ad2b” (origin/python3, python3) Demo testapp updates to fit python3 & django2.2
- “1916af6” Fix (temporary) conflict with mixin and mymixin in NOCLASS that gives a metaclass TypeError that makes unusable the crud
- “ffa7125” Python3 branch created.

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.16...master>

7.1.2 0.0.16

- “0aab3e2” Merge branch ‘master’ of <https://github.com/oscarmlage/django-cruds-adminlte>

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.15...master>

7.1.3 0.0.15

- “76a6913” (HEAD -> master, origin/master, origin/HEAD) Little improvement in the demo documentation
- “fd9fe2b” Merge pull request #97 from rtubio/master
- “3067e63” django-ajax fixed to point to the right min.js file
- “8f31118” Fixing security alerts from github
- “e3bf5a6” Added missed link in CHANGELOG
- “dcab42d” (tag: 0.0.14) New version 0.0.14

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.14...master>

7.1.4 0.0.14

- “fd0181e” (HEAD -> master, origin/master, origin/HEAD) Reviewed BSD-3 License
- “615d046” Added new contributors
- “379a885” Merge pull request #95 from miguelmendez17/Issue#89
- “5b8aeb9” Merge pull request #92 from alegambo/pasante
- “e95e425” Change in requirements (ajax) and solving Issue #89
- “9d4f88a” change the directory of the 403.html template from HttpRespense/403.html to cruds/403.html
- “8804b8c” Working on Changes of Issue 75
- “edf2e50” HTML cosmetics in detail/update
- “d00b5e8” FIX: You can add/define related_fields by model to not mess up things while saving models with FK fields
- “18fcc53” FIX: You can add/define list_fields by model to see only the fields you want in the list view, not all of them
- “705d1d6” Working on issue #75
- “d3c0c6a” PEP8 cosmetics
- “ae306bd” Merge branch ‘master’ of <https://github.com/oscarmlage/django-cruds-adminlte>
- “1939cd8” Merge pull request #85 from luisza/permcheck

- “2ef9cc3” fixed forbidden error
- “7836ba4” Merge pull request #83 from poipoi/master
- “b19c2b0” Re-order List View fields

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.13...master>

7.1.5 0.0.13

- “ac96520” (HEAD -> master, origin/master, origin/HEAD) PEP8 cosmetics
- “11b1444” Merge branch ‘master’ of <https://github.com/oscarmlage/django-cruds-adminlte>
- “58346ee” Merge pull request #81 from luisza/support2x
- “a7e9f4f” demo site fixed and prevent error on migrate with empty db

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.12...master>

7.1.6 0.0.12

- “28abf93” (HEAD -> master, origin/master, origin/HEAD) Fixes maxsplit issue + some pep8 cosmetic changes
- “b30b425” Merge pull request #78 from mbourqui/dev/crud/relatedmodelfields
- “04cd359” Merge pull request #77 from mbourqui/dev/crud/templateblocks
- “286f633” Support overriding blocks in CRUD views
- “da359d6” Support fields from related models in CRUD views
- “c07eafb” PEP8 Cosmetics (tests.py partially fixed)
- “6b5061b” PEP8 Cosmetics
- “9b438c7” Merge branch ‘master’ of <https://github.com/oscarmlage/django-cruds-adminlte>
- “ce6b0c4” Merge pull request #64 from luisza/perm_function
- “968f608” Merge pull request #67 from migue56/testing_django2_0
- “8c5f8f8” cleaning tests and comments
- “a26935d” adding comment on on test.py
- “52e402d” adding test to post without user log
- “b6ffea3” adding test to login/ out login views
- “42dfd86” testing delete url actions

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.11...master>

7.1.7 0.0.11

- “82cc5cf” (origin/master, origin/HEAD) Merge pull request #59 from luisza/patch-1
- “d50bbe0” Fixed broken register_tag behavior

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.10...master>

7.1.8 0.0.9

- “cce702a” Merge branch ‘master’ of <https://github.com/oscarmlage/django-cruds-adminlte>
- “621988e” Merge pull request #49 from migue56/django2_0_#48
- “b958d88” Checking lines
- “f473a6f” checking filter url when changes you from view to other view (checked)
- “b7fc88f” fixed filters paramters to grud.py
- “56f2efc” changes on filter.py to build pagination params
- “fd1f9d7” Changes filter.py get_cleaned_fields
- “b0730be” cleaning print() to filter.py
- “9c86a80” cleaning print() to filter.py
- “cf1e4f7” update filter.py
- “1b4d2e9” try/except ValueError over get_filter
- “9f4cd8f” Update demo to testing issues
- “6587ef5” Update demo to testing of issues django 2.0
- “ec175b1” Fixes #48: ‘ForeignKey’ object has no attribute ‘rel’
- “c5c3007” Merge pull request #47 from migue56/django2_0
- “ed61dfa” Update urls.py
- “d89d9d9” Update models.py
- “74c6050” Update utils.py
- “92dd062” Update crud_tags.py
- “5c59a38” Update adminlte_helpers.py
- “d28022a” Test and settings of demo django2.0/1.11/1.10
- “4f03c2c” Test and settings of demo django2.0/1.11/1.10
- “ce14461” Test and settings of demo django2.0
- “0c1b7bc” Test and settings of demo django2.0
- “f0ab4e8” Deleted debug-toolbar to demo django 2.0
- “f9ea2d6” Cleaning debug-toolbar to demo django 2.0
- “38482cb” Changed demo to django 2.0
- “5994072” Change requirements.txt ajax version
- “c09375f” Add intergerfild to templates colums
- “b35df35” Attibute ‘user.is_authenticated()’ don’t exist on django 2.0, change to ‘user.is_authenticated’
- “f760b44” Attibute ‘user.is_authenticated’ don’t exist on django 2.0
- “295d7d1” Changes of ‘installation.rst’ with django 2.0 librarys
- “3fdf9d3” fix ‘register.assignment_tag’ to django 2.0
- “1cd0548” Attribute ‘@register.assignment_tag’ don’t exist on django 2.0, changing that to @register.tag

- “ddf0f93” Import of file utils.py ‘from django.core.urlresolvers import reverse’ don’t exist on Django 2.0, that need to be changig to ‘from django.urls import reverse’
- “bb825b0” Import of file templatetags/crud_tags.py ‘from django.core.urlresolvers import reverse’ don’t exist on Django 2.0, that need to be changig to ‘from django.urls import reverse’.
- “04e8d17” Removing print
- “6a223af” Merge pull request #42 from justelex/patch-2
- “f5ad3bc” Merge pull request #41 from justelex/patch-1
- “f443982” Python 2 -> Python 3
- “2f5cfd8” Fixed: Wrong packagename in documtation

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.9...0.0.10>

7.1.9 0.0.8

- “33e5166” (HEAD -> master, origin/master, origin/HEAD) PEP8 Cosmetics
- “906b023” Merge pull request #40 from luisza/filter
- “bbde3f1” Update documentation of components
- “33ffed7” Documentation and some fixes
- “907d72f” Filters working with pagination but not with search form
- “80dc29a” Filter form
- “8df2764” Merge master
- “3d0999d” Add filter collapse

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.8...0.0.9>

7.1.10 0.0.7

- cb61f46 (HEAD -> master, origin/master, origin/HEAD) PEP8 Cosmetics
- ac5d979 Merge pull request #39 from luisza/pagination
- 309d37e Pagination functionality with support of numeration and prev_next
- 160550b Partially fixes #23, pagination now is set (hardcoded) to 10 + added pagination links in list template
- 76d8245 Fixes #34, corrections to the README
- 80931a8 Fixed pypi url
- 33350d3 Adding online resources to the documentation

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.7...0.0.8>

7.1.11 0.0.6

- 2afad55 (HEAD -> master, origin/master, origin/HEAD) Fixes an error related with #32 + Updates the demo
- 5a63520 Added feature: common cruds url + fixes #32
- 57b4ea6 Adding new column types

- fa54e77 PEP8 Cosmetics
- 4b458f9 Fixed: adds the filter only if there is request.GET.get(related), if not it works as before
- 6af9252 Merge pull request #31 from luisza/relatedCrud
- 5932da4 Related fields for foreignkeys
- a5596f2 Adding relad field function
- 908c055 Fixes CHANGELOG links

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.6...master>

7.1.12 0.0.5

- a87c87c - (HEAD -> master, origin/master, origin/HEAD) Added some screenshot to the documentation (2)
- 96bdb58 - Added some screenshot to the documentation
- dfa74ce - PEP8 cosmetic + some minor fixes related to search feature
- 9ffc51 - Merge pull request #28 from luisza/search
- 6dd38ff - Search feature
- 4a64ca0 - Typo fixed, thanks to Matej Vadjal (@matejv) for noticing

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.5...0.0.6>

7.1.13 0.0.4

- df6cbcf - (HEAD -> master, origin/master, origin/HEAD) PEP8 cosmetics
- cdbb7d4 - Merge pull request #26 from luisza/perm_fixes
- b50d8a5 - fixed contentype not found on migrations
- 616a75c - fixed permission and allow inherit of other base template
- eed91a9 - Merge branch 'master' into docs
- 16e89e3 - Update CRUDViews documentation
- d788b6d - PEP8 cosmetics
- dbcee00 - Merge pull request #24 from luisza/demo_fixes
- 48b1d42 - Adding demo site and fix input time format
- 1a7f4a1 - Fix str not _meta in list view
- c99f2ba - Filter views and demo site
- c26132a - Added demo site
- c4a3055 - Adding tests structure
- 159981f - Inline forms running properly in modal. Errors are well formatted now. Fixes #22
- c4a830f - Fixing MANIFEST
- 6aa9676 - Merge master
- dd24b8f - Documentation minor issue fixed (2)
- 999dafb - Documentation minor issue fixed

- 7f803d0 - Documentation improved, new screenshot images for inlines
- 6a8acc1 - Adding cruds-columns screenshot to documentation
- 8b879af - Change documentation theme for RTD
- 5d50505 - Improve README.rst with screenshots (6)
- ea59ff4 - Improve README.rst with screenshots (5, github does not allow include in README.rst for security reasons)
- 0f60b87 - Improve README.rst with screenshots (4)
- 8d540e5 - Improve README.rst with screenshots (3)
- fe702d8 - Improve README.rst with screenshots (2)
- 6e6d7b1 - Improve README.rst with screenshots
- b26d90d - Documentation organized (fixes #13)
- b54ea2b - Documentation organized (fixes #13)

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.4...0.0.5>

7.1.14 0.0.3

- **e4e3f0b - (HEAD -> master, origin/master, origin/HEAD) Add bootstrap** modal windows to inline behavior
- 23b111f - Fixes some inline_crud errors
- 7a6ae01 - Some pep8 cosmetics
- e8bed1d - Merge pull request #12 from luisza/ajax_inline
- bc1992a - Merge pull request #11 from luisza/pip_package
- eeceeb2 - New Inline ajax functionality
- 542c6ff - Adding templatetags
- d81441d - remove IDE .settings/
- f367bdb - Adding templates and static to pip package
- e93741d - Merge remote-tracking branch 'upstream/master'
- 70fe2c1 - Merge pull request #9 from luisza/new_crud_system
- 72faa04 - Removing old code sample related to ckeditor, fixes #10
- b6a7a41 - Fix documentation
- baa5568 - Namespace full support
- fa57be6 - Adding forms
- e9fd050 - Requirements for project
- 007c7ac - New CRUD System without test or docs
- df2a868 - Fixed bad link to jquery-cookie (added to static)
- c220cd7 - Fixed bad link to jquery-cookie (added to static)
- af58e34 - Added some missing stuff due to bad .gitignore. Fixes #8
- 907b288 - Improve columns related code

- 3da9be8 - Fixed documentation
- a14e7ec - Added support for columns in list pages
- 636734b - Updated documentation with Pypi link. Fixes #6

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.3...0.0.4>

7.1.15 0.0.2

- 9ce4da4 - Minor changes in setup.py
- 6fd30e3 - Updated README.rst
- **e89bda0 - Massive refactor, changed name django-cruds to django-cruds-adminlte**, the app now is cruds_adminlte (instead of cruds). Deleted orphan references to django-compressor. Fixed setup.py references to HISTORY.rst. Fixes #5
- 53c2c49 - Updated TODO.rst
- abbe074 - Fixes #1 (problem with old db values)
- 56ca599 - Added CONTRIBUTING.rst, glad if someone joins to the project
- 1ac4467 - Fixed documentation typo
- 6faf248 - Added CKEditorWidget
- a936206 - Documentation updated
- bb21eeb - Added datetimepicker, Fixed timepicker
- **aa01a09 - Adding DatePickerWidget, TimePickerWidget and ColorPickerWidget** to custom forms
- 1314604 - Improved documentation a bit
- d4075fd - Improved documentation a bit
- d641a8f - Added some screenshots to the documentation
- b2cb152 - Adding basic documentation

<https://github.com/oscarmlage/django-cruds-adminlte/compare/0.0.2...0.0.3>

7.2 Indices and tables

If you can't find the information you're looking for, have a look at the index or try to find it using the search function:

- [genindex](#)
- [modindex](#)
- [search](#)