Building large app on top of Django's admin

Story about Ralph 3

Mateusz Kurek

mkurek

Slides: https://mkurek.com/att17.pdf

Allegro.tech in numbers

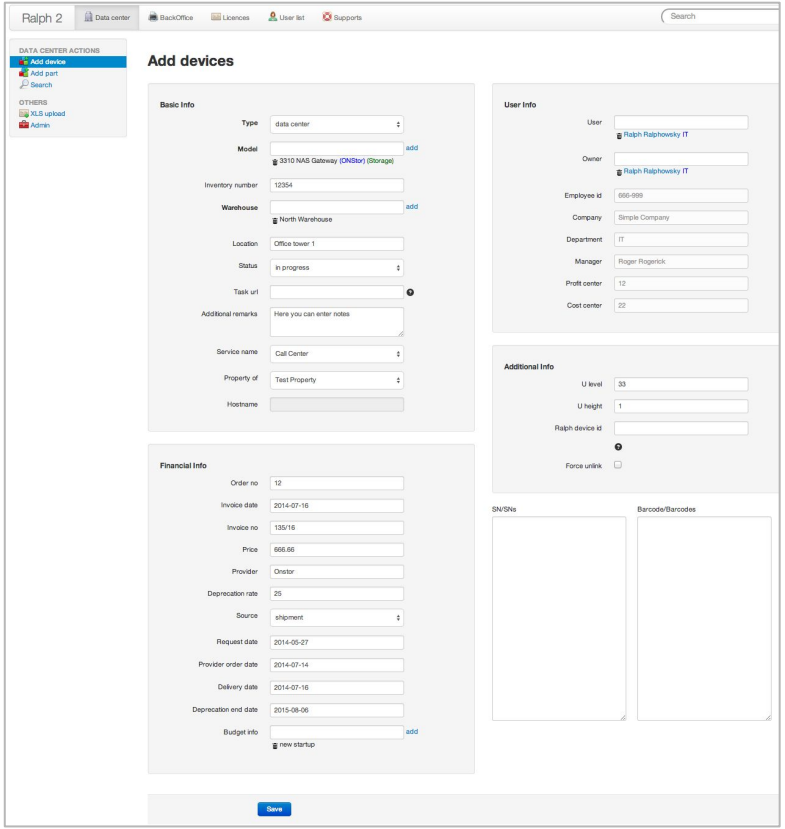
- 800+ microservices
- 40k+ RPS on data bus
- 4.5+ PB of data
- 5.5k+ data center assets in 2 Data Centers
- 10k+ hosts (physical, virtual, cloud)

What is Ralph?

- Assets Management System for Data Center and Back Office
- fully open-sourced → https://github.com/allegro/ralph
- developed in Allegro since 2011
- based on Django from the very beginning

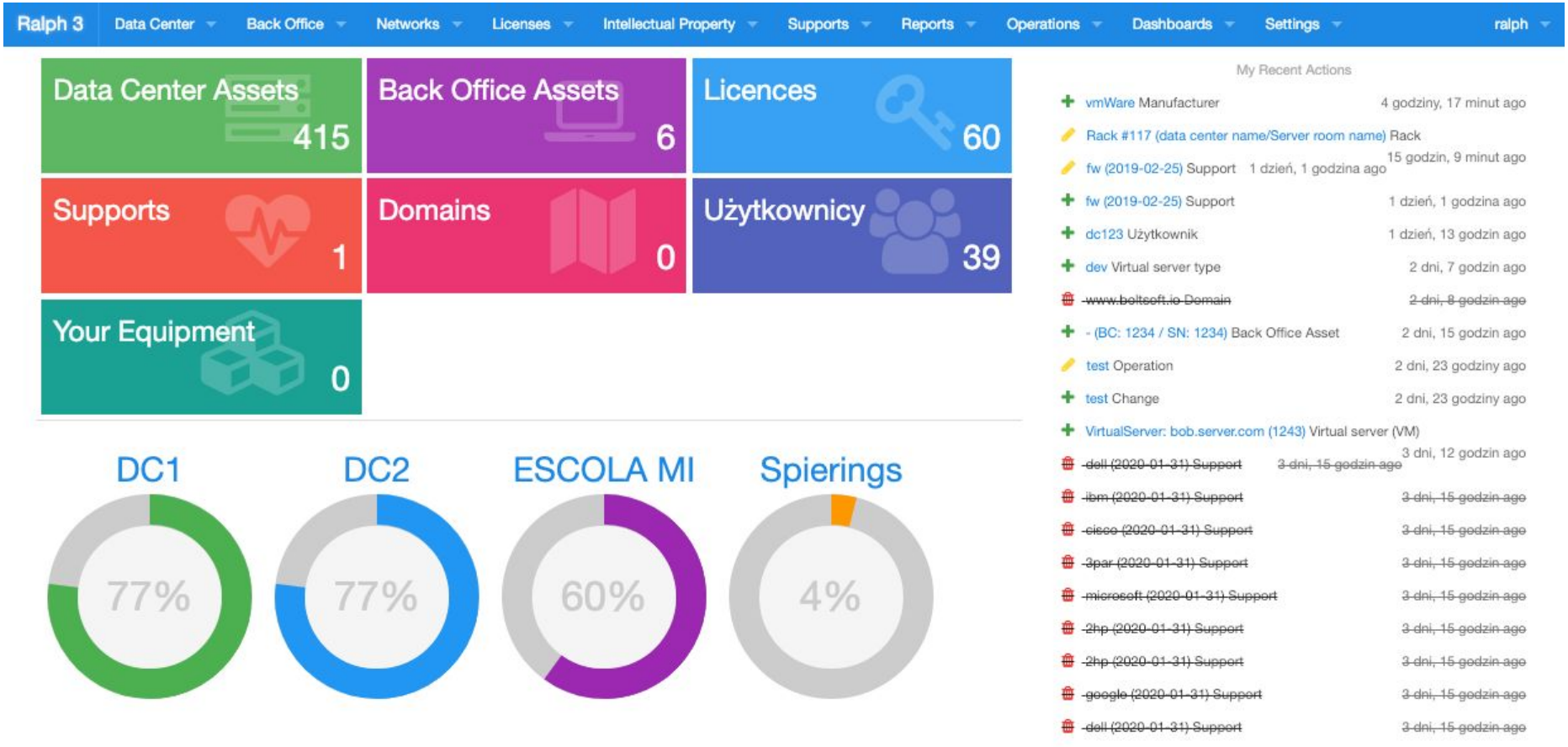
Cons of Ralph 2

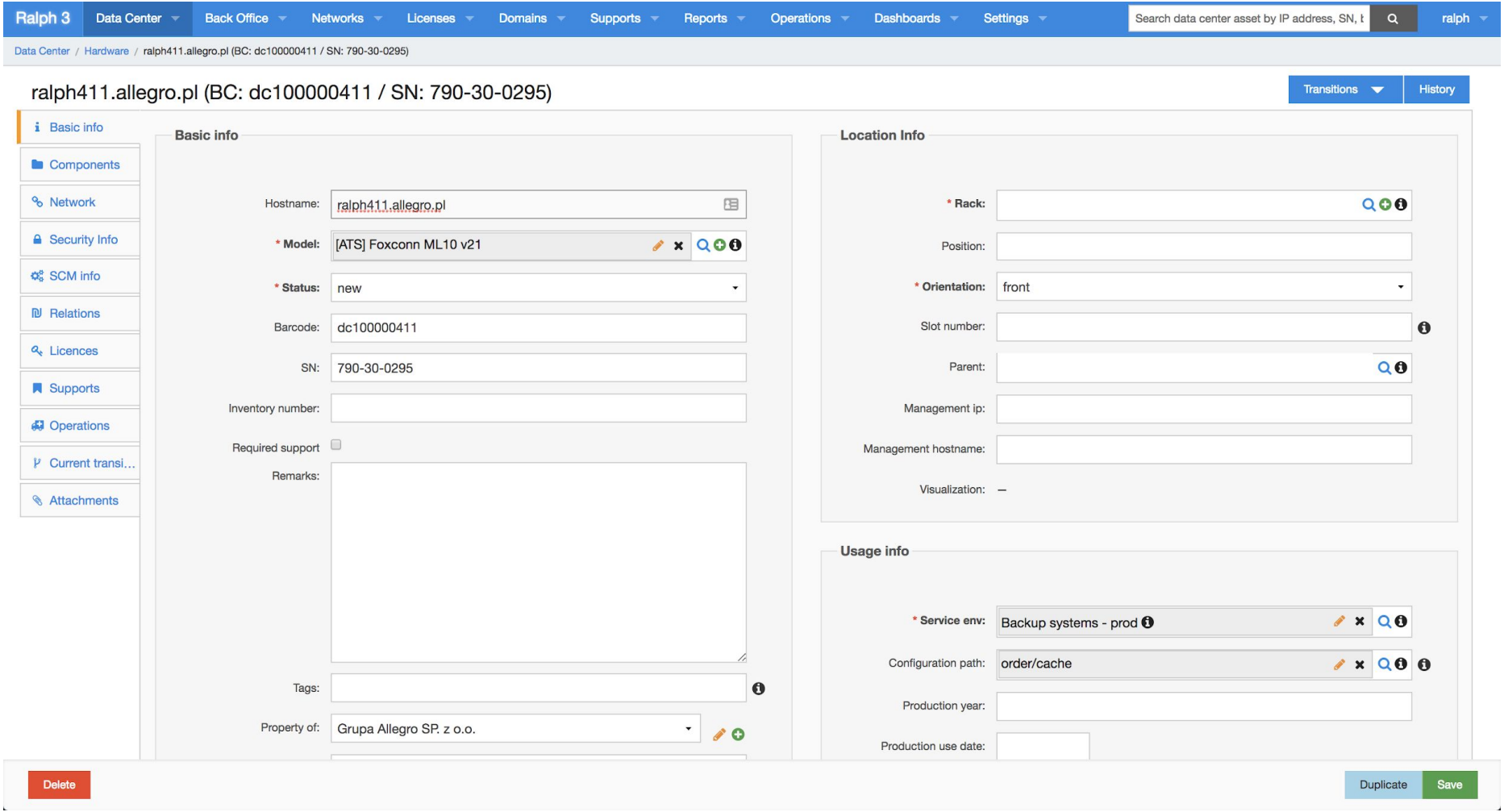
- poor API, not-so-well models
- information in 2-3 subsystems
- adding new features took too much time
- two inconsistent UIs - custom and admin
- "*data in admin is more reliable than in UI*"
- written in Python2



Let’s rewrite it!

- may 2015 - decision about rewriting Ralph from scratch
- use Django’s admin as the only one UI


Ralph 3
Data Center
Back Office
Networks
Licenses
Intellectual Property
Supports
Reports
Operations
Dashboards
Settings
ralph
Data Center Assets
415
Back Office Assets
6
Licences
60
Supports
1
Domains
0
Użytkownicy
39
Your Equipment
0
DC1
77%
DC2
77%
ESCOLA MI
60%
Spierings
4%
My Recent Actions
vmWare Manufacturer 4 godziny, 17 minut ago
Rack #117 (data center name/Server room name) Rack 15 godzin, 9 minut ago
fw (2019-02-25) Support 1 dzień, 1 godzina ago
fw (2019-02-25) Support 1 dzień, 1 godzina ago
dc123 Użytkownik 1 dzień, 13 godzin ago
dev Virtual server type 2 dni, 7 godzin ago
www.boltsoft.io Domain 2 dni, 8 godzin ago
- (BC: 1234 / SN: 1234) Back Office Asset 2 dni, 15 godzin ago
test Operation 2 dni, 23 godziny ago
test Change 2 dni, 23 godziny ago
VirtualServer: bob.server.com (1243) Virtual server (VM) 3 dni, 12 godzin ago
dell (2020-01-31) Support 3 dni, 15 godzin ago
ibm (2020-01-31) Support 3 dni, 15 godzin ago
cisco (2020-01-31) Support 3 dni, 15 godzin ago
3par (2020-01-31) Support 3 dni, 15 godzin ago
microsoft (2020-01-31) Support 3 dni, 15 godzin ago
2hp (2020-01-31) Support 3 dni, 15 godzin ago
2hp (2020-01-31) Support 3 dni, 15 godzin ago
google (2020-01-31) Support 3 dni, 15 godzin ago
dell (2020-01-31) Support 3 dni, 15 godzin ago

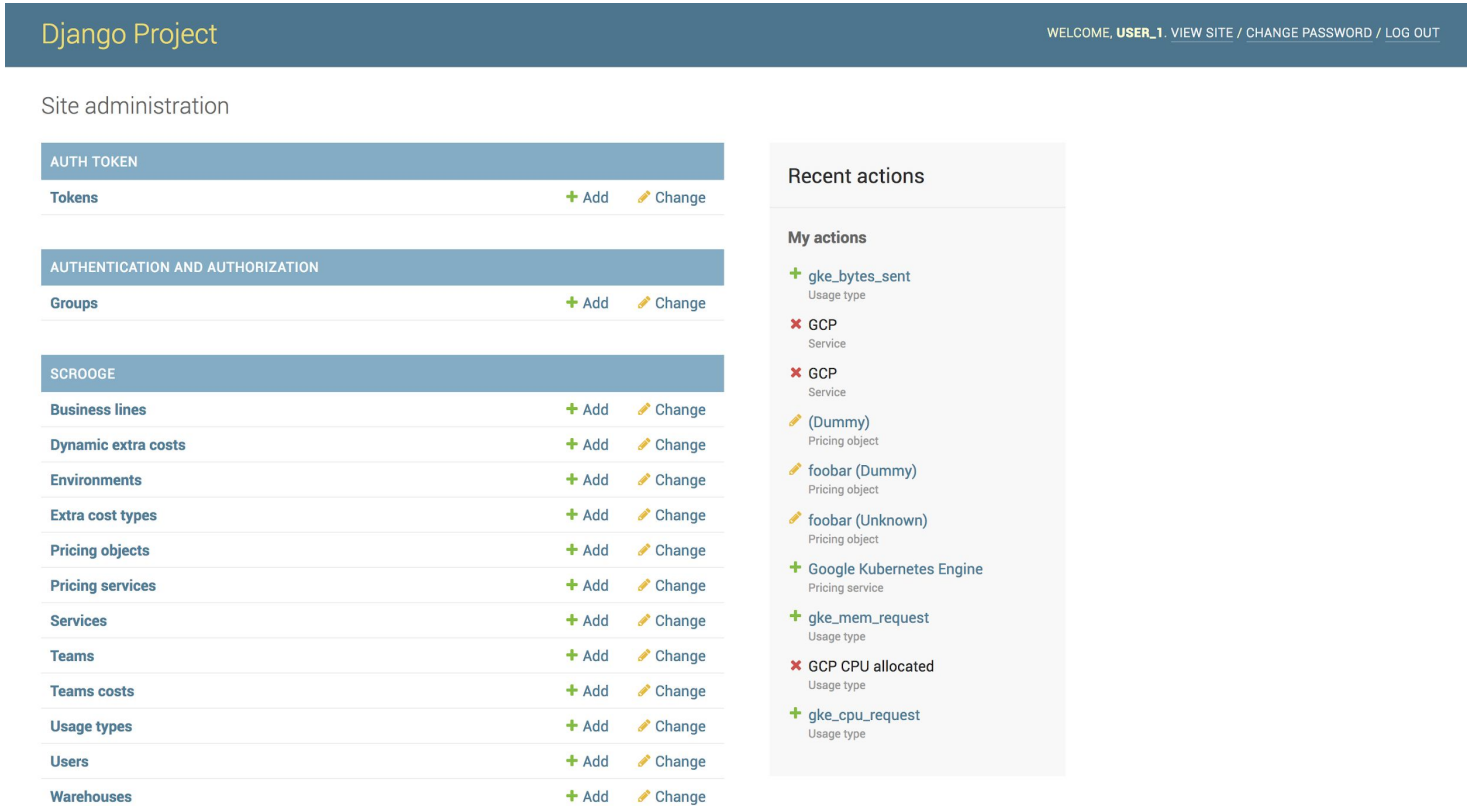

Ralph 3
Data Center
Back Office
Networks
Licenses
Domains
Supports
Reports
Operations
Dashboards
Settings
Search data center asset by IP address, SN, t
ralph
Data Center / Hardware / ralph411.allegro.pl (BC: dc100000411 / SN: 790-30-0295)
ralph411.allegro.pl (BC: dc100000411 / SN: 790-30-0295)
Transitions
History
Basic info
Components
Network
Security Info
SCM info
Relations
Licences
Supports
Operations
Current transi...
Attachments
Basic info
Hostname: ralph411.allegro.pl
* Model: [ATS] Foxconn ML10 v21
* Status: new
Barcode: dc100000411
SN: 790-30-0295
Inventory number:
Required support
Remarks:
Tags:
Property of: Grupa Allegro SP. z o.o.
Location Info
* Rack:
Position:
* Orientation: front
Slot number:
Parent:
Management ip:
Management hostname:
Visualization: –
Usage info
* Service env: Backup systems - prod
Configuration path: order/cache
Production year:
Production use date:
Delete
Duplicate
Save

The reasons behind choosing Django admin as our foundation

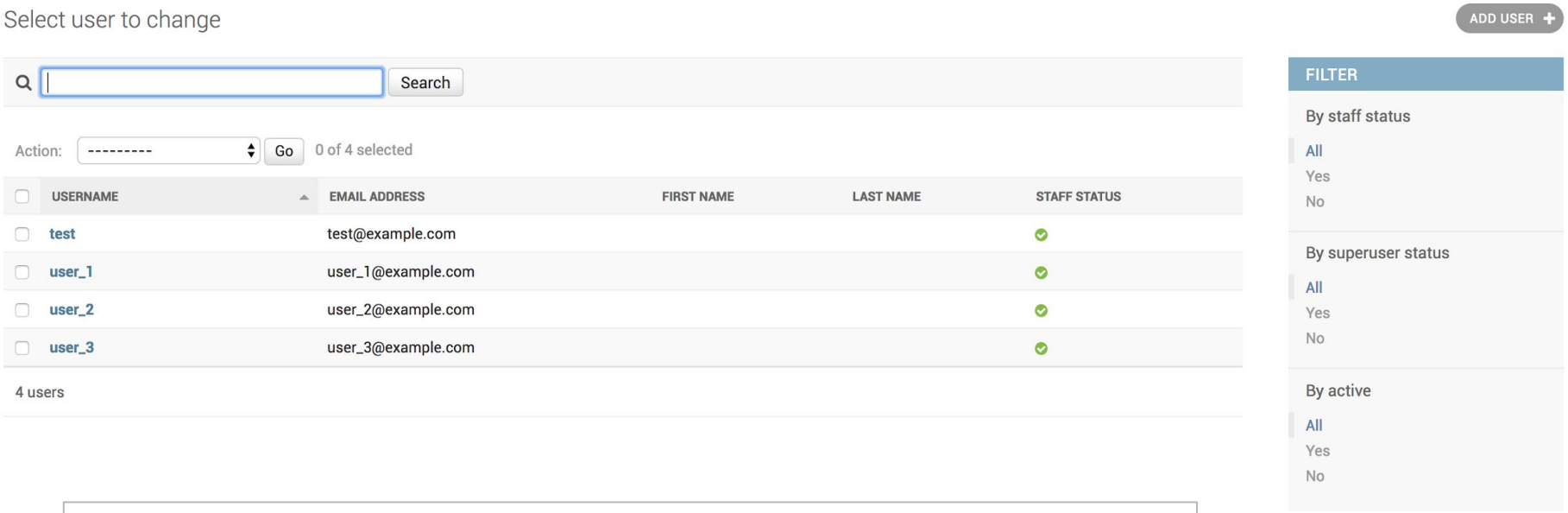
- one and only one UI
- easiness of adding new domain models in declarative way
- lot of external packages support

Built-in admin features

List of models

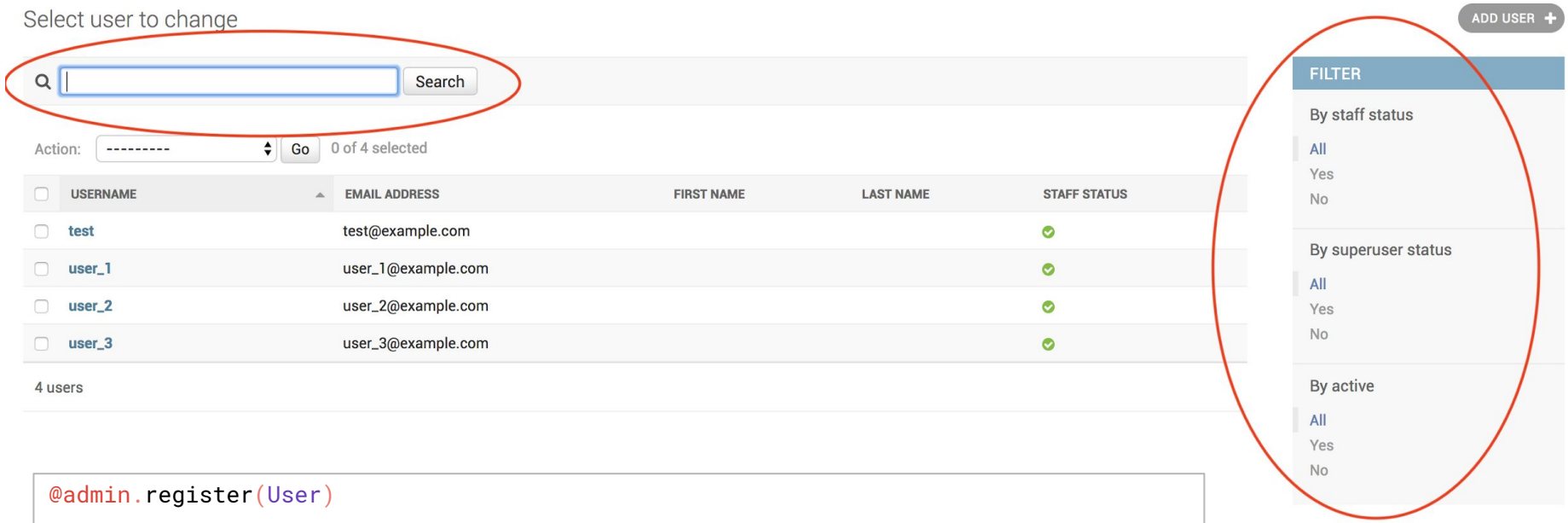


List of model entities




```python
@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('username', 'email', 'first_name', 'last_name', 'is_staff')
```

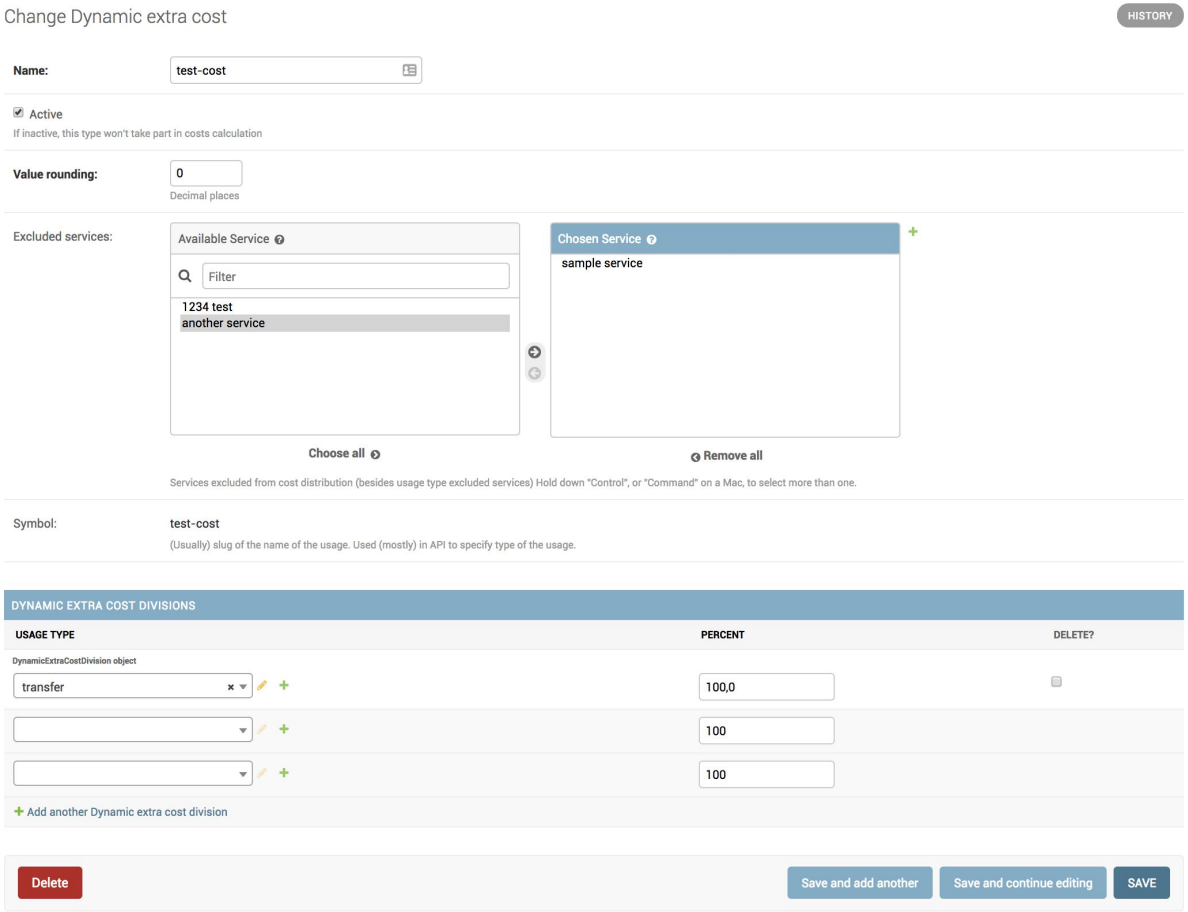
Simple filters and search




```python
@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('username', 'email', 'first_name', 'last_name', 'is_staff')
    list_filter = ('is_staff', 'is_superuser', 'is_active')
    search_fields = ('username', 'first_name', 'last_name', 'email')
```

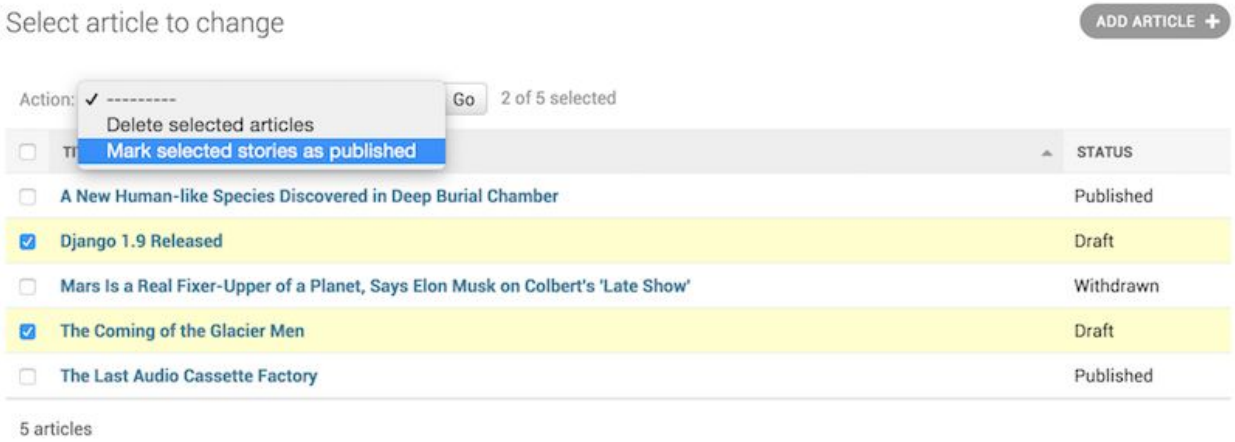
Model form

```
@register(models.DynamicExtraCostType)
class DynamicExtraCostTypeAdmin(
    admin.ModelAdmin
):
    readonly_fields = ('symbol',)
```



But did you know that admin supports also

Actions



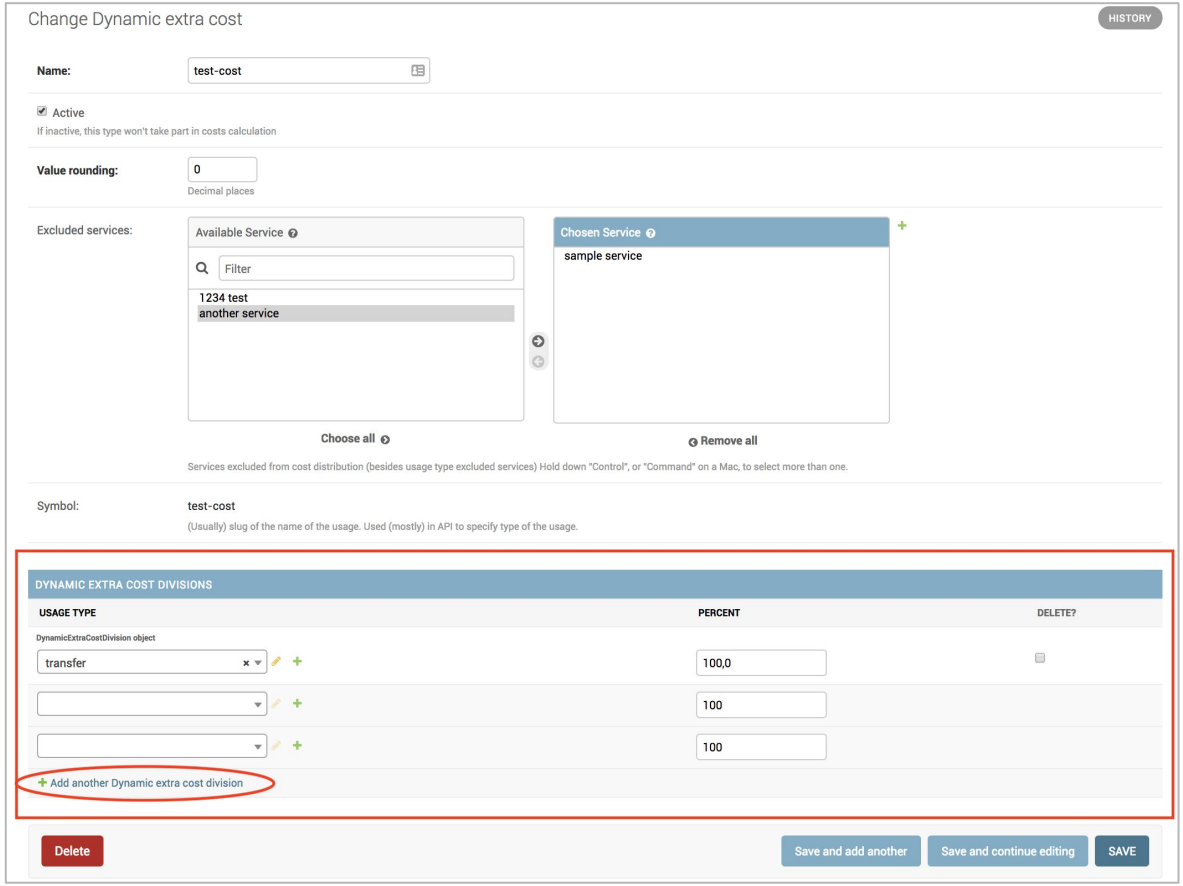

```
def make_published(modeladmin, request, queryset):
    queryset.update(status='p')
make_published.short_description = "Mark selected stories as published"

class ArticleAdmin(admin.ModelAdmin):
    actions = [make_published]
```

Inlines

```
class DivisionInline(
    admin.TabularInline
):
    model = DynamicExtraCostDivision


@register(
    models.DynamicExtraCostType
)
class DynamicExtraCostTypeAdmin(
    admin.ModelAdmin
):
    inlines = [DivisionInline]
```

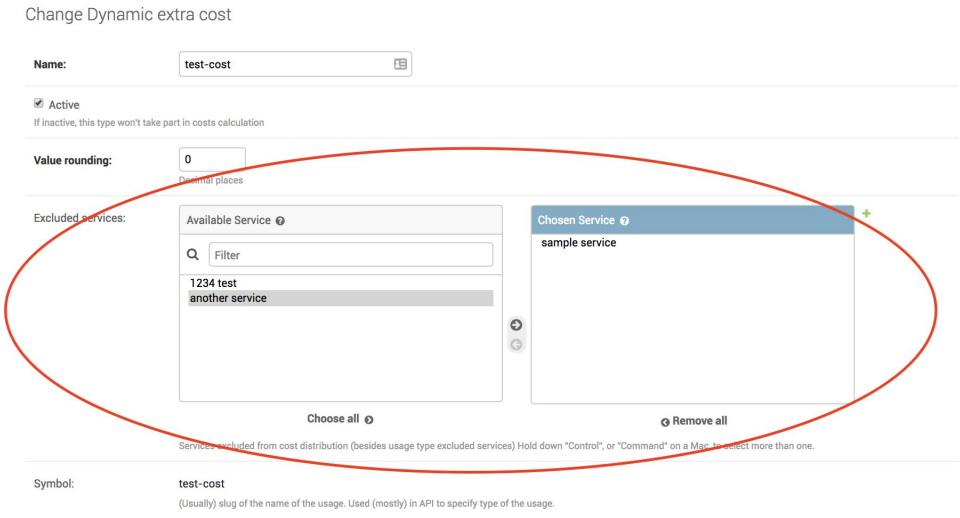


Custom form or custom widgets

```python
from django import forms
from django.contrib import admin
from django.contrib.admin.widgets import (
    FilteredSelectMultiple
)

class DynamicExtraTypeForm(forms.ModelForm):
    class Meta:
        model = models.DynamicExtraCostType
        fields = '__all__'
        widgets = {
            'excluded_services':FilteredSelectMultiple(
                'Service', False
            )
        }

@register(models.DynamicExtraCostType)
class DynamicExtraCostTypeAdmin(admin.ModelAdmin):
    form = DynamicExtraTypeForm
    # ...
```



Callable field on listing

Select pricing service to change

Search

Action: --------- Go 0 of 2 selected

NAME	SYMBOL	ACTIVE	PLUGIN TYPE	SERVICES
Cache	cache	✓	universal	redis, memcache
Databases	databases	✓	fixed price	mysql, mongo, oracle

2 pricing services

```
@register(models.PricingService)
class PricingServiceAdmin(admin.ModelAdmin):
    list_display = ('name', 'symbol', 'active', 'plugin_type', 'get_services')

    def get_services(self, pricing_service):
        return ', '.join(map(str, pricing_service.services.all()))
    get_services.short_description = _('Services')
```

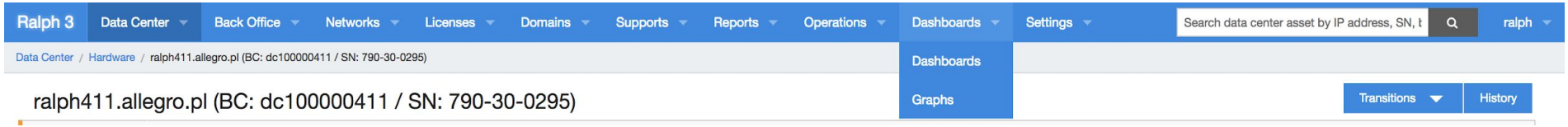
And many others

- ordering
- items per page
- select related
- prefetch related

Ralph 3

How did we start?

- chosen front-end framework → foundation
- modified admin default listing view → django-sitetree



- overridden a lot of Django's admin templates
- created our extension of `ModelAdmin` → `RalphAdmin`

External libraries

- `django-sitetree` for menu and breadcrumbs
- `django-reversion` for history of objects
- `django-rest-framework` for REST API
- `django-import-export` for CSV/Excel import/export of objects

Extra views

Views not related to any particular model

- create template file
- create view inheriting from Django's `TemplateView`
- add it to urls and menu

Server rooms

baba
- room1

DC.MMT.01
- UC1
- UC2

DC2
- Server Room B

dc2.1
- dc2.1_a

France
- server room 123
- Server Room A

Predio "C"
- 3st. Andar

TLV LX
- TLV Room 1A

Tohid
- Mobinone

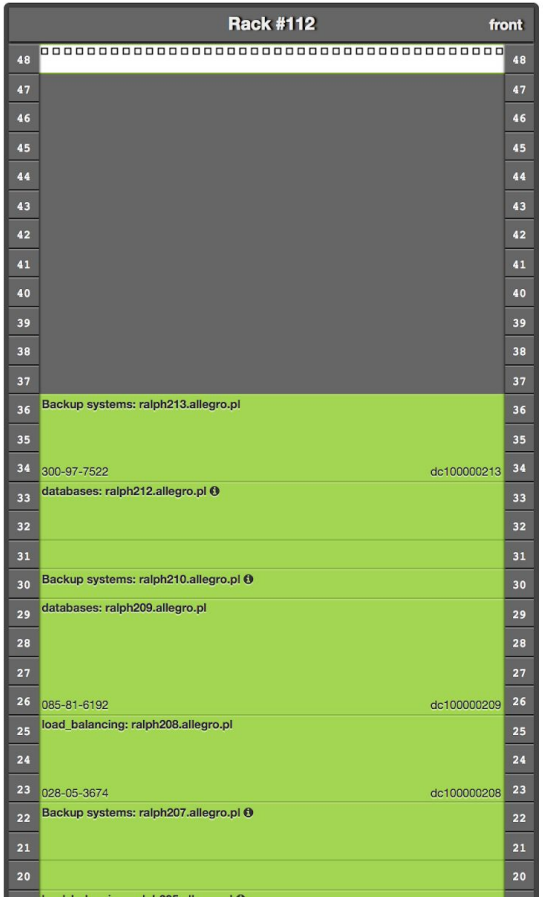
Racks in Server Room B

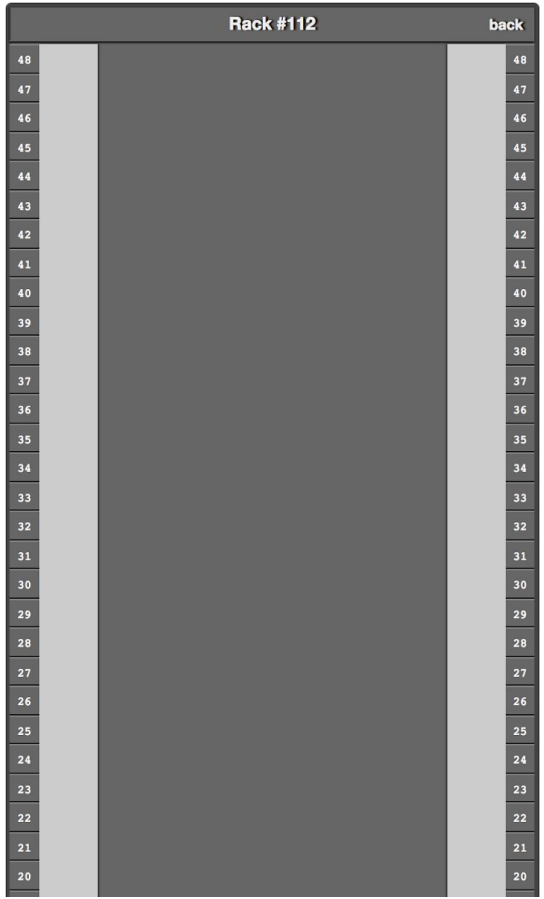
Filter rack by name

Rack #110
Rack #111
Rack #112
Rack #113

Service Catalog	-
Hostname	-
Model	-
Barcode	-
SN	-
Position	-
Height	-
Management	-
Remarks	-

-





Extra views for particular model

- two ways: form based on admin or customized view (with template)
- inherit from `RalphDetailView` or `RalphDetailViewAdmin`
    - automatically handle tab, icon, url etc

Test-licence-2 (BC: 100011 / SN: -)

- Basic info
- Compon…
- Network
- Security I…
- SCM info
- Relations
- Licences
- Supports
- Operations
- Current tr…
- Attachme…

Base object licences

Licence	Quantity	Delete?
11 (8 free) x Magic - 2018-03-27 (1)	1	
111 (110 free) x Photoshop - None (12345)	1	
	1	

Add another Base object licence

s12345.mydc.net (BC: 12345 / SN: 543-211)

- Basic info
- Components
- Network
- Security Info
- SCM info
- Relations
- Licences
- Supports
- Operations
- DNS
- Current tra…
- Attachments

Security scan

last scan date	2018-08-15
scan status	ok
next scan date	2018-08-22
details	-

Vulnerabilities

Name	Risk	Patch deadline
Ubuntu 14.04 LTS / 16.04 LTS / 17.10 : ubuntu-release-upgrader vulnerability (USN-3623-1)	high	2018-05-10
Ubuntu 14.04 LTS / 16.04 LTS : libxcursor vulnerability (USN-3729-1)	high	2018-09-06
Ubuntu 14.04 LTS / 16.04 LTS : gnupg vulnerability (USN-3733-1)	high	2018-09-06
Ubuntu 16.04 LTS : openjdk-8 vulnerability (USN-3734-1)	medium	2018-10-09

```python
class DCAssetLicence(RalphDetailViewAdmin):
    icon = 'key'
    name = 'dc_asset_licences'
    label = _('Licences')
    url_name = 'data_center_asset_licences'

    class DCAssetLicenceInline(RalphTabularInline):
        model = BaseObjectLicence
        raw_id_fields = ('licence',)
        extra = 1

    inlines = [DCAssetLicenceInline]
```

```python
@register(DataCenterAsset)
class DataCenterAssetAdmin(RalphAdmin):
    # ...
    change_views = [DCAssetLicence, DCAssetSecurityInfo]
    # ...
```

```python
class DCAssetSecurityInfo(RalphDetailView):
    icon = 'lock'
    label = 'Security Info'
    name = 'security_info'
    url_name = 'datacenter_asset_security_info'
    template_name = 'security/security_info.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        try:
            scan = self.object.securityscan
        except ObjectDoesNotExist:
            scan = None
        context['security_scan'] = scan
        return context
```

See more: https://github.com/allegro/ralph/tree/ng/src/ralph/admin

Polymorphic models

3 styles of model inheritance in Django

- Abstract base classes
- Proxy models
- Multi-table inheritance

Abstract base classes

- use the parent class to hold information that you don’t want to have to type out for each child model
- database table will NOT be created for abstract model

Abstract base classes

```python
class CommonInfo(models.Model):
    name = models.CharField(max_length=100)
    age = models.PositiveIntegerField()

    class Meta:
        abstract = True


class Student(CommonInfo):
    home_group = models.CharField(max_length=5)


class Employee(CommonInfo):
    company = models.CharField(max_length=50)
```

```
> .tables
abc_inheritance_employee
abc_inheritance_student
```

```
> PRAGMA table_info([abc_inheritance_student]);
cid     name        type        notnull   dflt_value  pk
------  ----------  ----------  --------  ----------  -----
0       id          integer     1                     1
1       name        varchar(10  1                     0
2       age         integer un  1                     0
3       home_group  varchar(5)  1                     0
```

```
> PRAGMA table_info([abc_inheritance_employee]);
cid     name        type        notnull   dflt_value  pk
------  ----------  ----------  --------  ----------  -----
0       id          integer     1                     1
1       name        varchar(10  1                     0
2       age         integer un  1                     0
3       company     varchar(50  1                     0
```

Proxy models

- use it to change the Python behavior of a model (ex. default manager)
- operates on the same database table as its parent model class
- `QuerySets` still return the model that was requested

```python
class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    age = models.PositiveIntegerField()


class OrderedPerson(Person):
    class Meta:
        proxy = True
        ordering = ["last_name"]


class UnderagePersonManager(models.Manager):
    def get_queryset(self):
        return super().get_queryset().filter(age__lte=18)


class UnderagePerson(Person):
    objects = UnderagePersonManager()
    class Meta:
        proxy = True
```

Multi-table inheritance

- each model in the hierarchy is a model all by itself
- each model corresponds to its own database table
- each model can be queried and created individually
- link between child and each of its parents model using `OneToOneField`

Multi-table inheritance

```python
class Person(models.Model):
    name = models.CharField(max_length=100)
    age = models.PositiveIntegerField()


class Student(Person):
    home_group = models.CharField(
        max_length=5
    )


class Employee(Person):
    company = models.CharField(
        max_length=50
    )
```

```
> .tables
mmi_inheritance_person
mmi_inheritance_empolyee
mmi_inheritance_student
```

```
> PRAGMA table_info([mmi_inheritance_person]);
cid     name        type        notnull   dflt_value  pk
------  ----------  ----------  --------  ----------  -----
0       id          integer     1                     1
1       name        varchar(10  1                     0
2       age         integer un  1                     0
```

```
> PRAGMA table_info([mmi_inheritance_student]);
cid     name                type        notnull   dflt_value  pk
------  ------------------  ----------  --------  ----------  -----
0       person_ptr_id       integer     1                     1
1       home_group          varchar(5)  1                     0
```

```
> PRAGMA table_info([mmi_inheritance_employee]);
cid     name                type        notnull   dflt_value  pk
------  ------------------  ----------  --------  ----------  -----
0       person_ptr_id       integer     1                     1
1       company             varchar(50  1                     0
```

Multi-table inheritance

```
>>> from mmi_inheritance.models import Person, Student, Employee

>>> Student.objects.create(name='John Doe', age=22, home_group='G1')
<Student: John Doe>
>>> Employee.objects.create(name='Uncle Bob', age=44, company='Allegro')
<Employee: Uncle Bob>

>>> Person.objects.all()
<QuerySet [<Person: John Doe>, <Person: Uncle Bob>]>
>>> Student.objects.all()
<QuerySet [<Student: John Doe>]>

>>> Person.objects.filter(name='Uncle Bob')
<QuerySet [<Person: Uncle Bob>]>
>>> Employee.objects.filter(name='Uncle Bob')
<QuerySet [<Employee: Uncle Bob>]>
```

Polymorphic models - goal

```
>>> Person.polymorphic_objects.all()
<QuerySet [<Student: John Doe>, <Employee: Uncle Bob>]>
```

Ralph’s Polymorphic models - why?

- Common namespace of identifiers (identify object by single value) → CMDB
- Enforcement of some common fields (like service)
- Easiness of querying for related objects (ex. hosts of all types)

Ralph's Polymorphic models

```
from ralph.lib.polymorphic import Polymorphic, PolymorphicBase

class BaseObject(Polymorphic, metaclass=PolymorphicBase):
    name = models.CharField(max_length=255)
    remarks = models.TextField(blank=True)
    service = models.ForeignKey('Service')

class DataCenterAsset(BaseObject):
    model = models.ForeignKey(AssetModel)
    rack = models.ForeignKey(Rack)
    # ...

class CloudHost(BaseObject):
    cloud_provider = models.ForeignKey(CloudProvider)
    hypervisor = models.ForeignKey(DataCenterAsset)
    # ...
```

Ralph's Polymorphic models - features

- `polymorphic_select_related`, `polymorphic_prefetch_related` - allow to specify `select_related` / `prefetch_related` for child models

```
MyBaseModel.polymorphic_objects.polymorphic_select_related(
    MyDescendantModel=['related_field1', 'related_field2'],
    MyDescendantModel2=['related_field3'],
)
```

- Easy access from base model to final class instance (through `last_descendant` property)
- `polymorphic_objects` is default manager

Ralph's Polymorphic models - how?

- using Django's `contenttypes` framework
- custom QuerySet manager with heavily rewritten `iterator`:
  - Call super's `iterator` to get list of base objects and store it's content types and primary keys
  - Group objects by content types
  - For each content type call query on its model filtering by primary keys (with polymorphic filters)
  - Restore original order of items
- optimized number of queries: `1 + num of content types`

See more: https://github.com/allegro/ralph/tree/ng/src/ralph/lib/polymorphic

Granular permissions

Permissions in Django

- Django Admin has add, change and delete permission for model
- Django lack of view permission (out-of-the-box) before Django 2.1
- lot of open-source packages:
  - django-guardian
  - django-rules

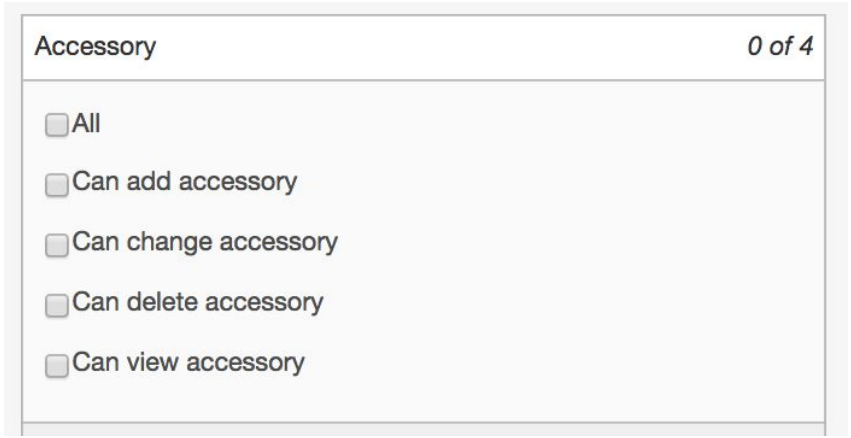
4 levels of permissions

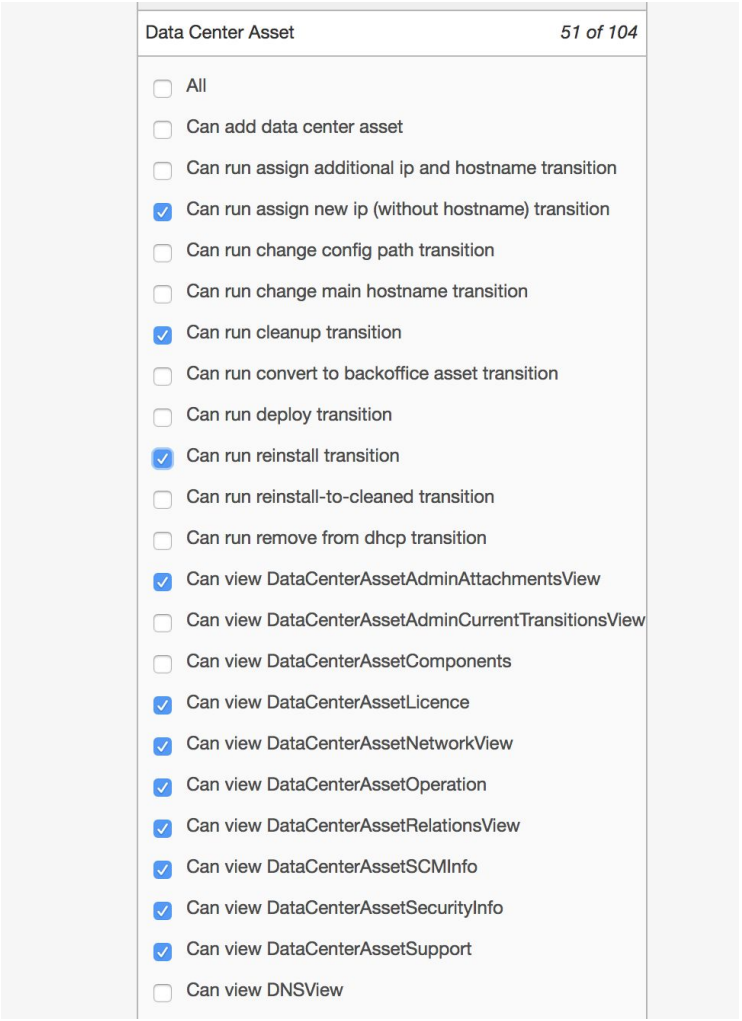
- permission for model
- permission for object
- permission per field
- permission for extra tabs and transitions

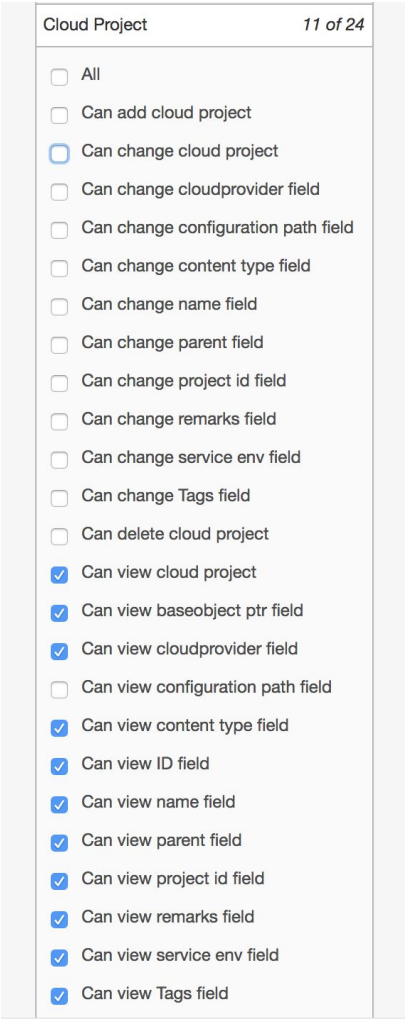
* Name: some_group

Permissions:

Accessory	*1 of 4*
Asset	*25 of 54*
Asset Holder	*1 of 4*
Attachment	*1 of 4*
Attachment Item	*1 of 4*
Back Office Asset	*1 of 95*
Base Object	*9 of 30*
Base Object Cluster	*0 of 4*
Base Object Licence	*2 of 4*
Base Objects Support	*2 of 4*
Budget Info	*1 of 4*
Business Segment	*1 of 4*
Category	*1 of 4*
Change	*0 of 4*
Cloud Flavor	*13 of 24*
Cloud Host	*18 of 33*
Cloud Project	*12 of 24*
Cloud Provider	*1 of 4*

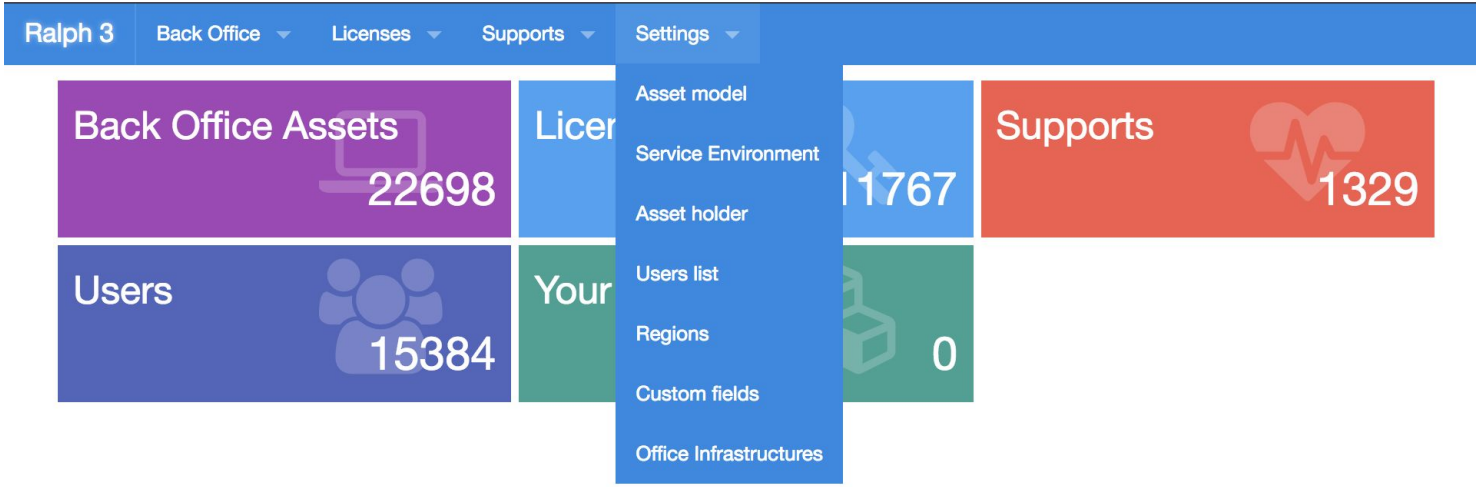


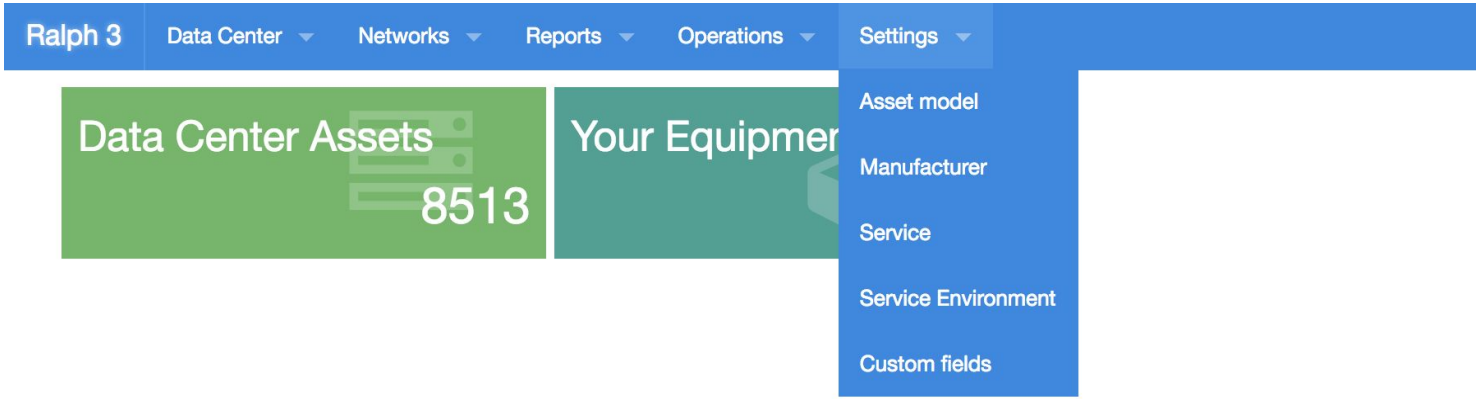

Data Center Asset
51 of 104
All
Can add data center asset
Can run assign additional ip and hostname transition
Can run assign new ip (without hostname) transition
Can run change config path transition
Can run change main hostname transition
Can run cleanup transition
Can run convert to backoffice asset transition
Can run deploy transition
Can run reinstall transition
Can run reinstall-to-cleaned transition
Can run remove from dhcp transition
Can view DataCenterAssetAdminAttachmentsView
Can view DataCenterAssetAdminCurrentTransitionsView
Can view DataCenterAssetComponents
Can view DataCenterAssetLicence
Can view DataCenterAssetNetworkView
Can view DataCenterAssetOperation
Can view DataCenterAssetRelationsView
Can view DataCenterAssetSCMInfo
Can view DataCenterAssetSecurityInfo
Can view DataCenterAssetSupport
Can view DNSView


Cloud Project
11 of 24
All
Can add cloud project
Can change cloud project
Can change cloudprovider field
Can change configuration path field
Can change content type field
Can change name field
Can change parent field
Can change project id field
Can change remarks field
Can change service env field
Can change Tags field
Can delete cloud project
Can view cloud project
Can view baseobject ptr field
Can view cloudprovider field
Can view configuration path field
Can view content type field
Can view ID field
Can view name field
Can view parent field
Can view project id field
Can view remarks field
Can view service env field
Can view Tags field

Permission for model

- scan all models and all its fields etc. and create permissions for them
- permission for adding/changing/deleting model is handled by Django
- render menu for each user basing on their permissions


Ralph 3
Back Office
Licenses
Supports
Settings
Asset model
Service Environment
Asset holder
Users list
Regions
Custom fields
Office Infrastructures
Back Office Assets
22698
Licer
1767
Supports
1329
Users
15384
Your
0


Ralph 3
Data Center
Networks
Reports
Operations
Settings
Asset model
Manufacturer
Service
Service Environment
Custom fields
Data Center Assets
8513
Your Equipmer

Permission for object

- override some of Admin methods:
  - `has_change_permission`
  - `has_delete_permission`
  - `get_queryset`
- Model has to
  - inherit from `PermissionsForObjectMixin`
  - add `Permissions` inner class with `has_access` property
- `RalphAdmin` provides both permission for object and per field

```python
@user_permission
def object_has_region(user):
    """
    Check if object's region is one of user regions.
    """
    return models.Q(region__in=user.regions_ids)


class BackOfficeAsset(PermissionsForObjectMixin):
    region = models.ForeignKey(Region, blank=False, null=False)

    class Permissions:
        has_access = object_has_region
```

Permissions per field

- the most tricky
- override Admin methods (to narrow down form to fields user has access):
  - `get_fieldsets`
  - `get_readonly_fields`
  - `get_form`
  - `get_list_display`
- model has to inherit from PermByFieldMixin abstract model

```
from ralph.lib.permissions import PermByFieldMixin

class LicenceType(PermByFieldMixin, models.Model):
    pass
```

1 x Sublime Text - None (94842)

History

- Basic info
- Assignments
- Assigned t…
- Attachments

Basic info

* **Licence type:** OTHER
Manufacturer:
* **Software:** Sublime Text
* **Inventory number:** 94842
SN / key: XXX
Valid thru:
License details:
* **Region:** PL
Remarks:

Financial info

Order no:
Invoice no:
Price:
Depreciation rate: 50,00
Invoice date:
* **Number of purchased items:** 1
Used: 1
Free: 0
Accounting id:
Budget info:
Provider:
Office infrastructure:
Property of: ---------

Delete

Save

1 x Sublime Text - None (94842)

History

- Basic info
- Assignments
- Assigned t…
- Attachments

Basic info

Licence type: OTHER

Manufacturer: (None)

Software: Sublime Text

Inventory number: 94842

SN / key: XXX

Valid thru: (None)

License details:

Region: PL

Remarks:

Service env: (None)

Financial info

Order no: -

Invoice no: -

Price: (None)

Invoice date: (None)

Number of purchased items: 1

Used: 1

Free: 0

Accounting id: -

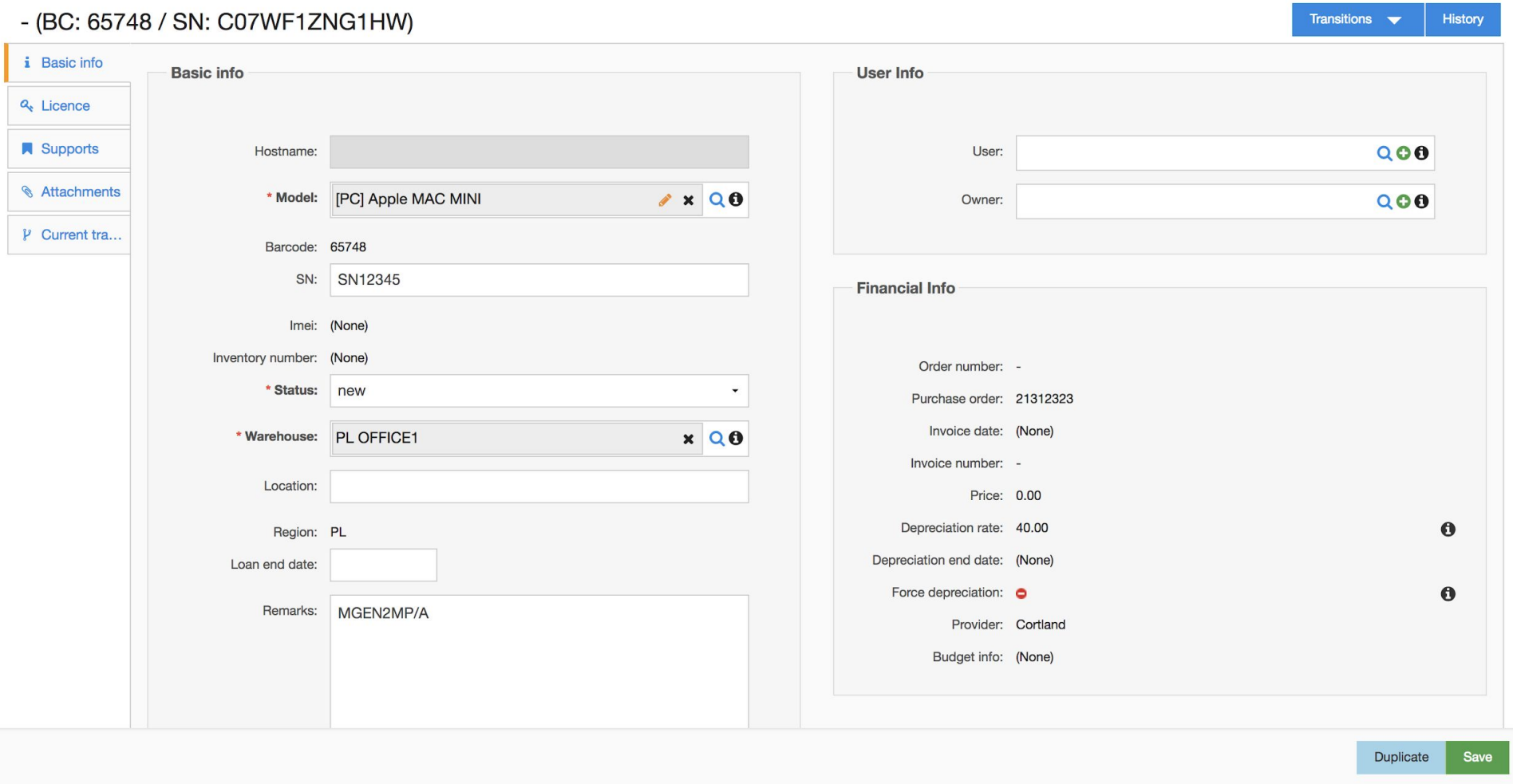
Budget info: (None)

Provider: -

Office infrastructure:

Property of: (None)

Save



See more: https://github.com/allegro/ralph/tree/ng/src/ralph/lib/permissions

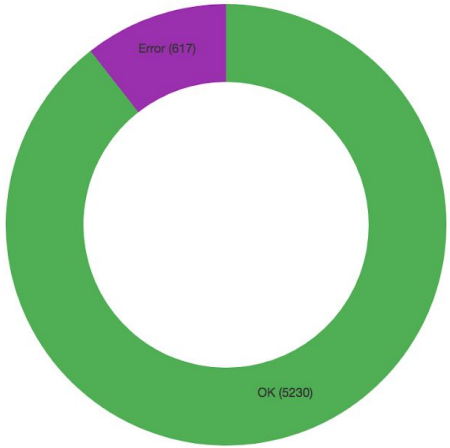
Charts and dashboards

Charts and dashboards

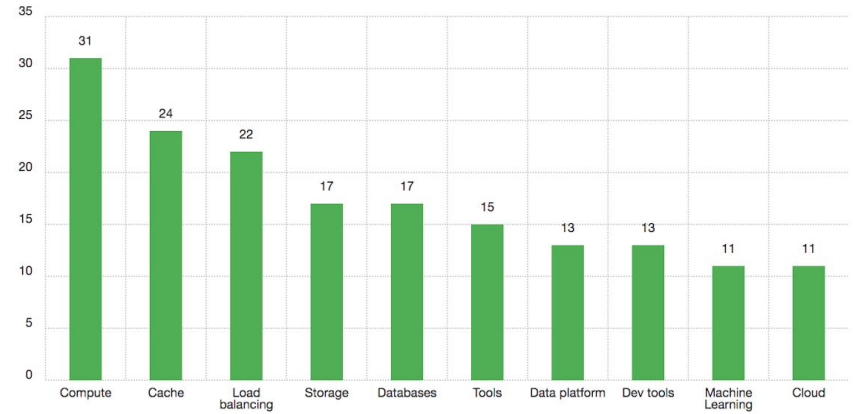
- allow to visualize data from Ralph easily
- using ORM-like declarative query
- using chartist to display charts
- charts could be connected together in dashboards

SCM status

DC Hosts SCM status



Failed DC Hosts SCM Status by service only prod



[assets] Workstations on stock with status "free"

History

* Name: [assets] Workstations on stock with status "free"

Description:

* Model: Back Office Asset

* Aggregate type: Count

* Chart type: Vertical Bar

Params:

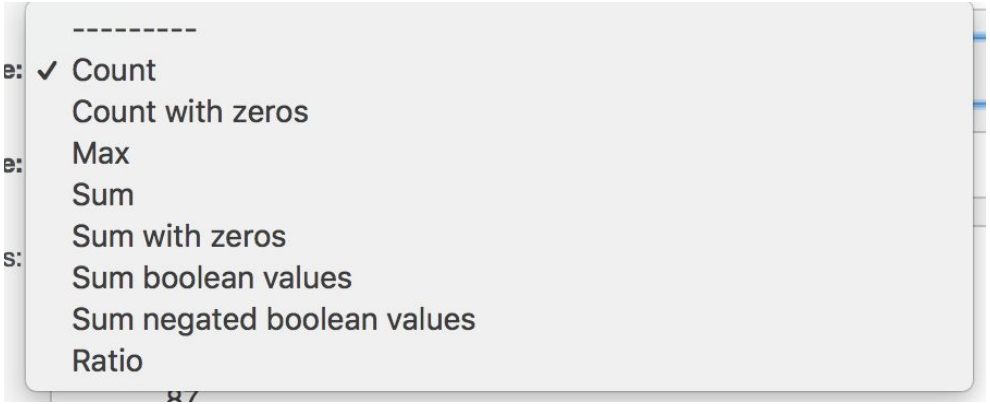
```
{
    "filters": {
        "model__category__in": [
            87,
            88
        ],
        "region_id": 6,
        "status": 10
    },
    "labels": "model__category__name",
    "series": "barcode"
}
```

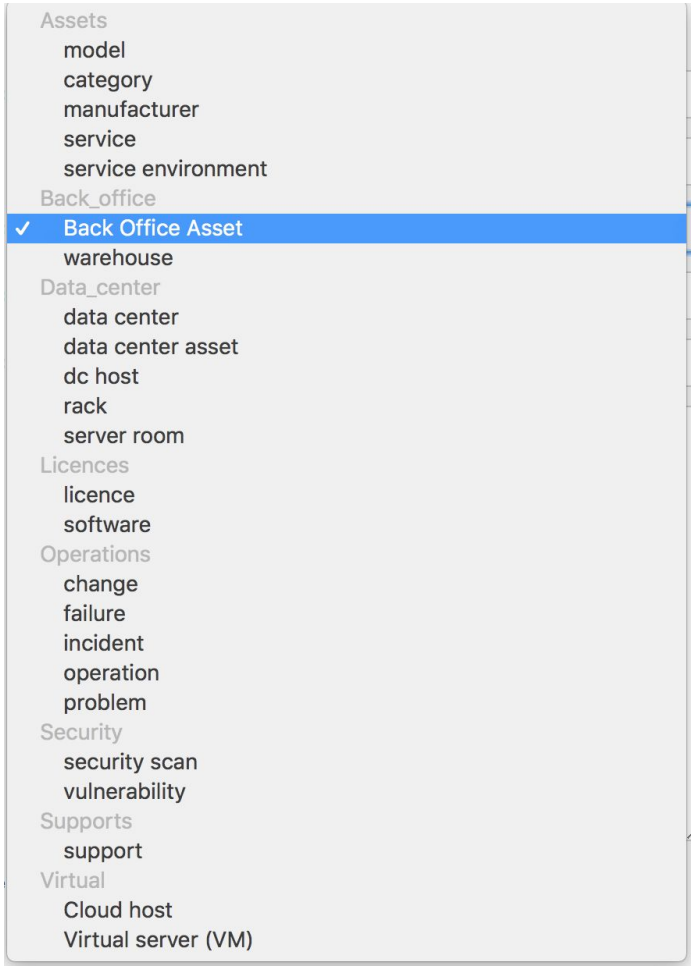
Active

Push to statsd

Preview

Graph:





See more: https://github.com/allegro/ralph/tree/ng/src/ralph/dashboards

Filtering

- built-in Django’s Admin filters are (too) simple
- we’ve created our own filters for a lot of types of fields (like text field, choice field, foreign key etc)
- each filter comes with its own template (input, select, autocomplete etc)
- but we wanted to keep using them as simple as possible

```
list_filter = [
    'barcode', 'status', 'imei', 'sn', 'model', 'purchase_order',
    'model__category', 'loan_end_date', 'niw', 'model__manufacturer',
]
```

Back Office Assets

Actions

Go 0 of 100 selected

Filters

Barcode

Status

All

Imei

SN

Model

Purchase order

Hostname

Required support

All

Region

Warehouse

	Status	Barcode	Purchase order	M
	free	bo100000227		[E
	new	bo100000226		[Di
	new	bo100000225	(None)	[Di
	new	bo100000224	(None)	[N
	new	bo100000222	(None)	[M
	new	bo100000221	(None)	[K
	new	bo100000220	(None)	[H
	new	bo100000219	(None)	[E se
	new	bo100000218	(None)	[E
	new	bo100000217	(None)	[Di
	new	bo100000216	(None)	[Di
	new	bo100000215	(None)	[E
	new	bo100000214	(None)	[Di
	new	bo100000213	(None)	[Di

Data center assets

Actions

Go 0 of 100 selected

Filters

Hostname

Service env

Configuration class path

Configuration class module

MAC Address

IP

Vulnerability patch deadline

Start YYYY-MM

End YYYY-MM-

Security scan vulnerabilities

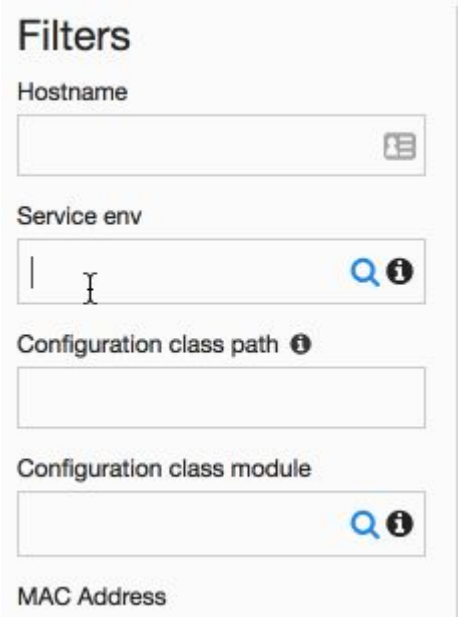
Security scan is patched

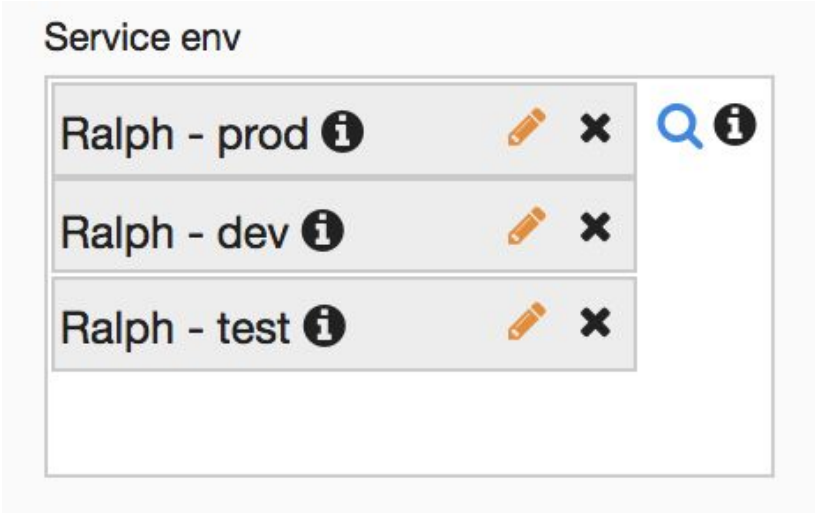
All

Invoice number

	Hostname	Status	Barcode
	ralph421.allegro.pl	in use	(None)
	ralph421.allegro.pl	new	dc100000421
	ralph420.allegro.pl	new	dc100000420
	ralph419.allegro.pl	in use	dc100000419
	ralph418.allegro.pl	new	dc100000418
	ralph417.allegro.pl	new	dc100000417
	ralph416.allegro.pl	new	dc100000416
	ralph415.allegro.pl	new	dc100000415
	ralph414.allegro.pl	new	dc100000414
	ralph413.allegro.pl	new	dc100000413
	ralph412.allegro.pl	new	dc100000412
	ralph411.allegro.pl	new	dc100000411
	ralph410.allegro.pl	new	dc100000410
		new	dc100000409


Invoice number
Use ; or | separators to search for multiple values


Filters
Hostname
Service env
Configuration class path
Configuration class module
MAC Address

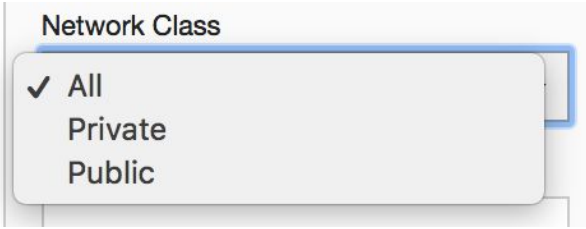

Service env
Ralph - prod
Ralph - dev
Ralph - test

Using dedicated filters for field

```
class NetworkClassFilter(ChoicesListFilter):
    _choices_list = [('private', _('Private')), ('public', _('Public'))]

    def queryset(self, request, queryset):
        if not self.value():
            return queryset
        if self.value().lower() == 'private':
            queryset = queryset.filter(PRIVATE_NETWORK_FILTER)
        elif self.value().lower() == 'public':
            queryset = queryset.exclude(PRIVATE_NETWORK_FILTER)
        return queryset


class IPAddressAdmin(RalphAdmin):
    list_filter = [..., ('address', NetworkClassFilter), ...]
```

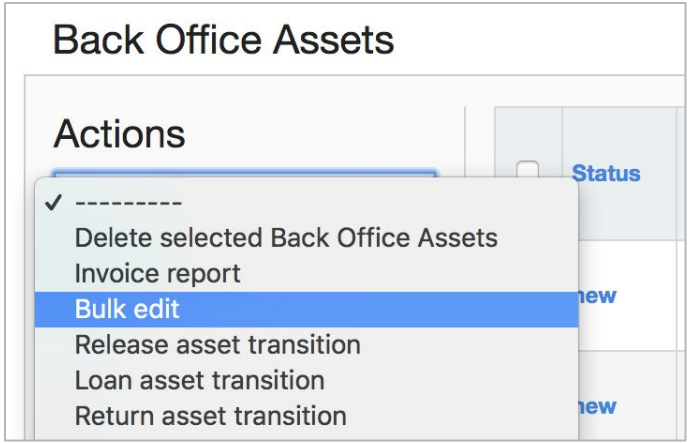


See more: https://github.com/allegro/ralph/blob/ng/src/ralph/admin/filters.py

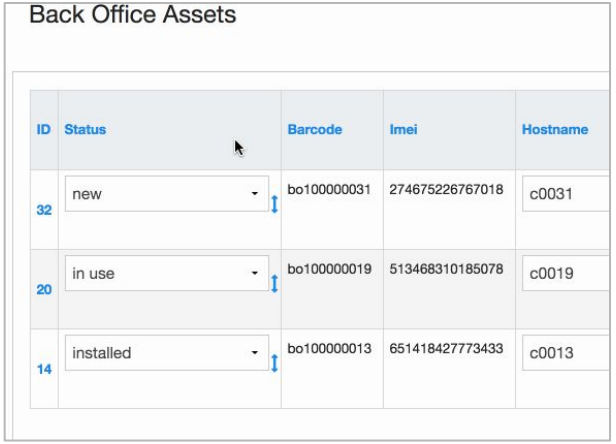
Bulk edit

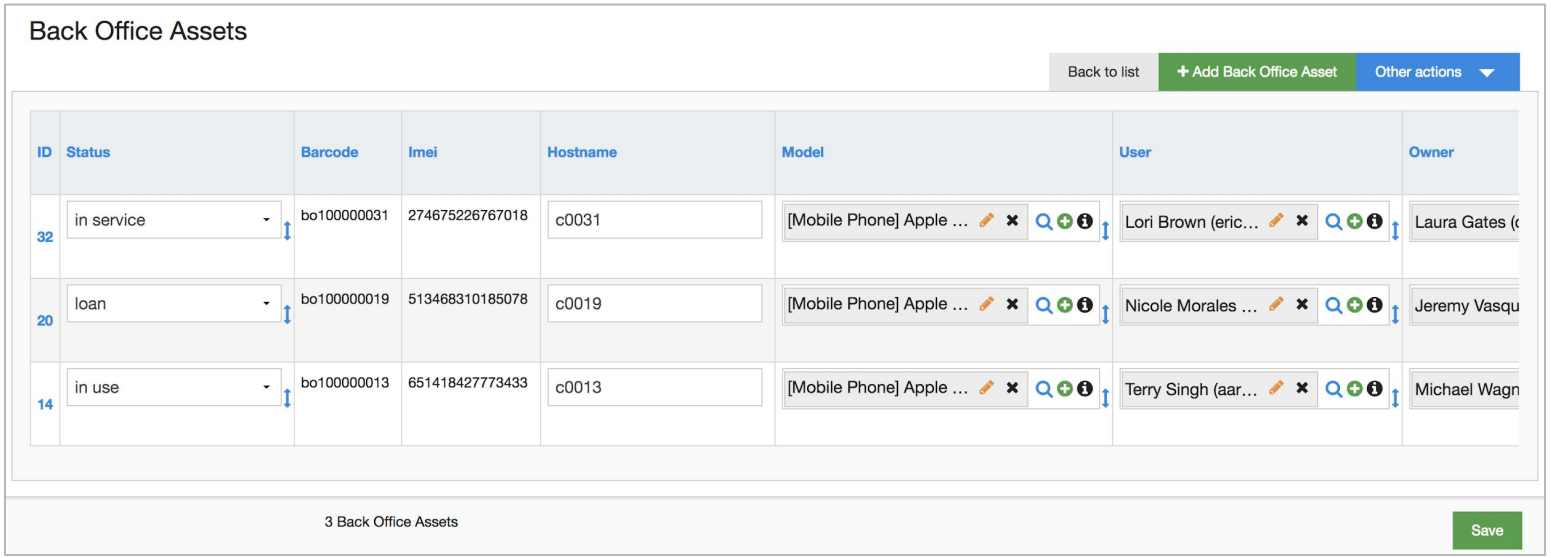
Bulk edit

- allow to edit multiple items at once
- respect permissions (per field)
- custom Django’s Admin action: `bulk_edit_action`


Back Office Assets
Actions

Delete selected Back Office Assets
Invoice report
Bulk edit
Release asset transition
Loan asset transition
Return asset transition
Status


Back Office Assets
ID
Status
Barcode
Imei
Hostname
32
new
bo100000031
274675226767018
c0031
20
in use
bo100000019
513468310185078
c0019
14
installed
bo100000013
651418427773433
c0013


Back Office Assets
Back to list
+ Add Back Office Asset
Other actions
ID
Status
Barcode
Imei
Hostname
Model
User
Owner
32
in service
bo100000031
274675226767018
c0031
[Mobile Phone] Apple ...
Lori Brown (eric...
Laura Gates (
20
loan
bo100000019
513468310185078
c0019
[Mobile Phone] Apple ...
Nicole Morales ...
Jeremy Vasqu
14
in use
bo100000013
651418427773433
c0013
[Mobile Phone] Apple ...
Terry Singh (aar...
Michael Wagn
3 Back Office Assets
Save

Bulk edit - how?

```
@register(BackOfficeAsset)
class BackOfficeAssetAdmin(BulkEditChangeListMixin, RalphAdmin):
    # ...
    bulk_edit_list = [
        'licences', 'status', 'barcode', 'imei', 'hostname', 'model',
        'purchase_order', 'user', 'owner', 'warehouse', 'sn', 'region',
        'property_of', 'remarks',
    ]
    bulk_edit_no_fillable = ['barcode', 'sn', 'imei', 'hostname']
    # ...
```

Bulk edit - how?

- using Django's admin `list_editable` property underneath
- override `get_queryset` to narrow it down only to selected objects
- override `get_list_display` to display editable fields (with respect to permissions)

See more: https://github.com/allegro/ralph/blob/ng/src/ralph/admin/mixins.py#L532

Conclusions

Conclusions

- Django’s admin is very powerful, but sometimes things getting complicated
- Lot of work with metaclasses and/or content types to make something generic
- It’s not so easy to customize/extend Django’s admin!
- Do we regret it? NO!

Q&A

https://github.com/allegro/ralph

Mateusz Kurek

mkurek

Slides: https://mkurek.com/att17.pdf