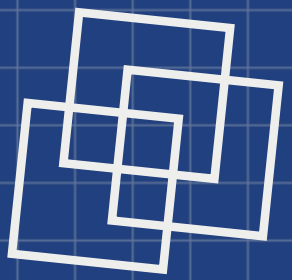
Developing Review Board

christian hammond
david trowbridge

2004 - ~600 employees



2004 - ~600 employees

2004 - ~600 employees

2007 - ~5000 employees

Navbar

Outgoing

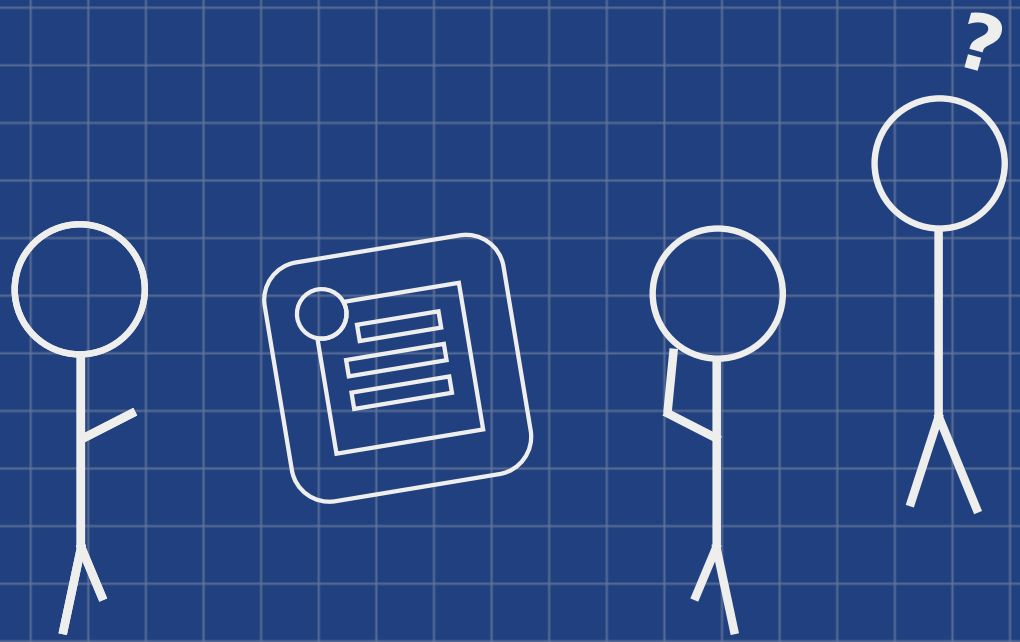
Incoming

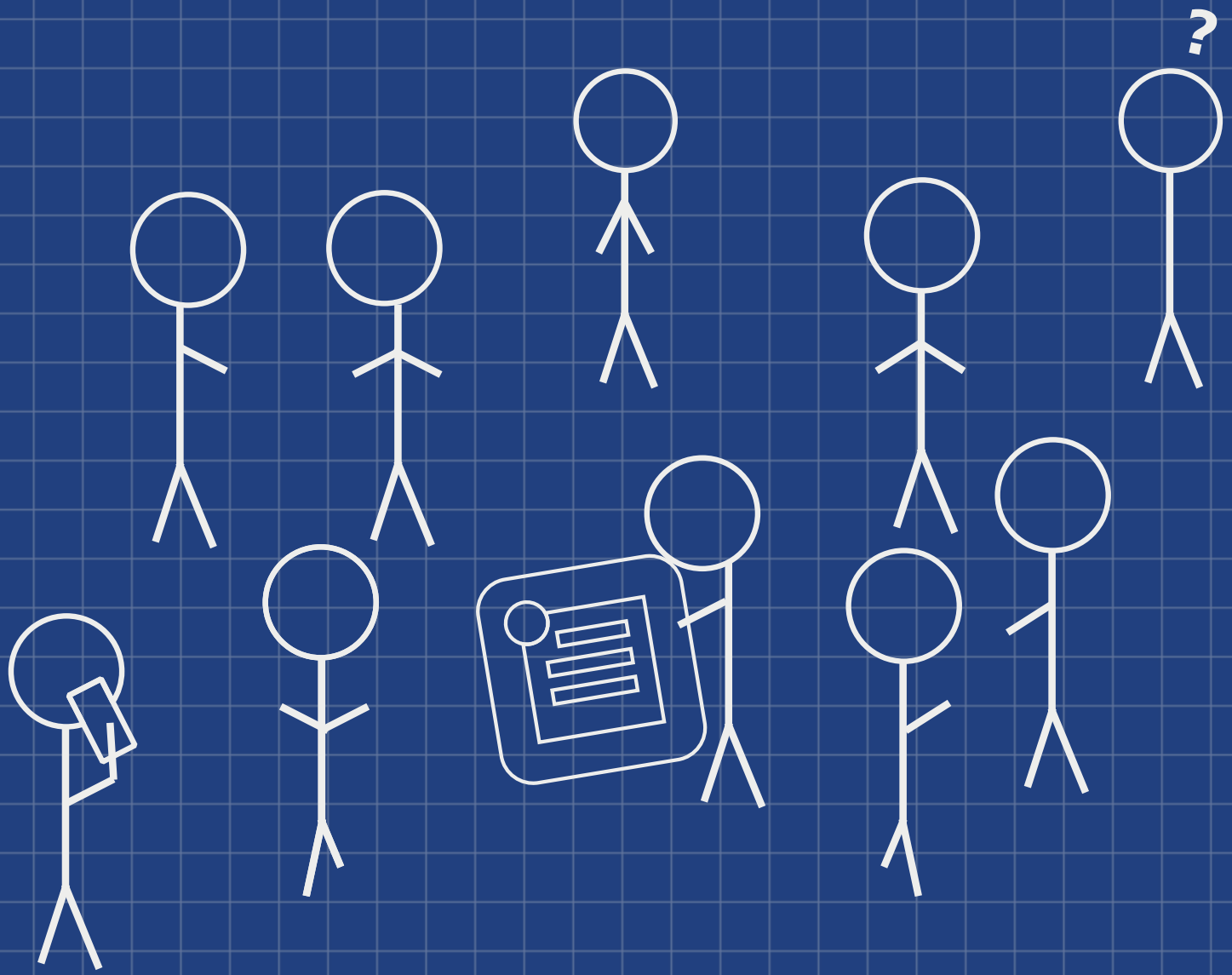
Groups

Dashboard Title

Review request entries

1 per line





Datacomp

Hedgehog Lab

NetLogic Microsystems

rPath

Tripwire

VMware

Wesabe

Yahoo! Web Search

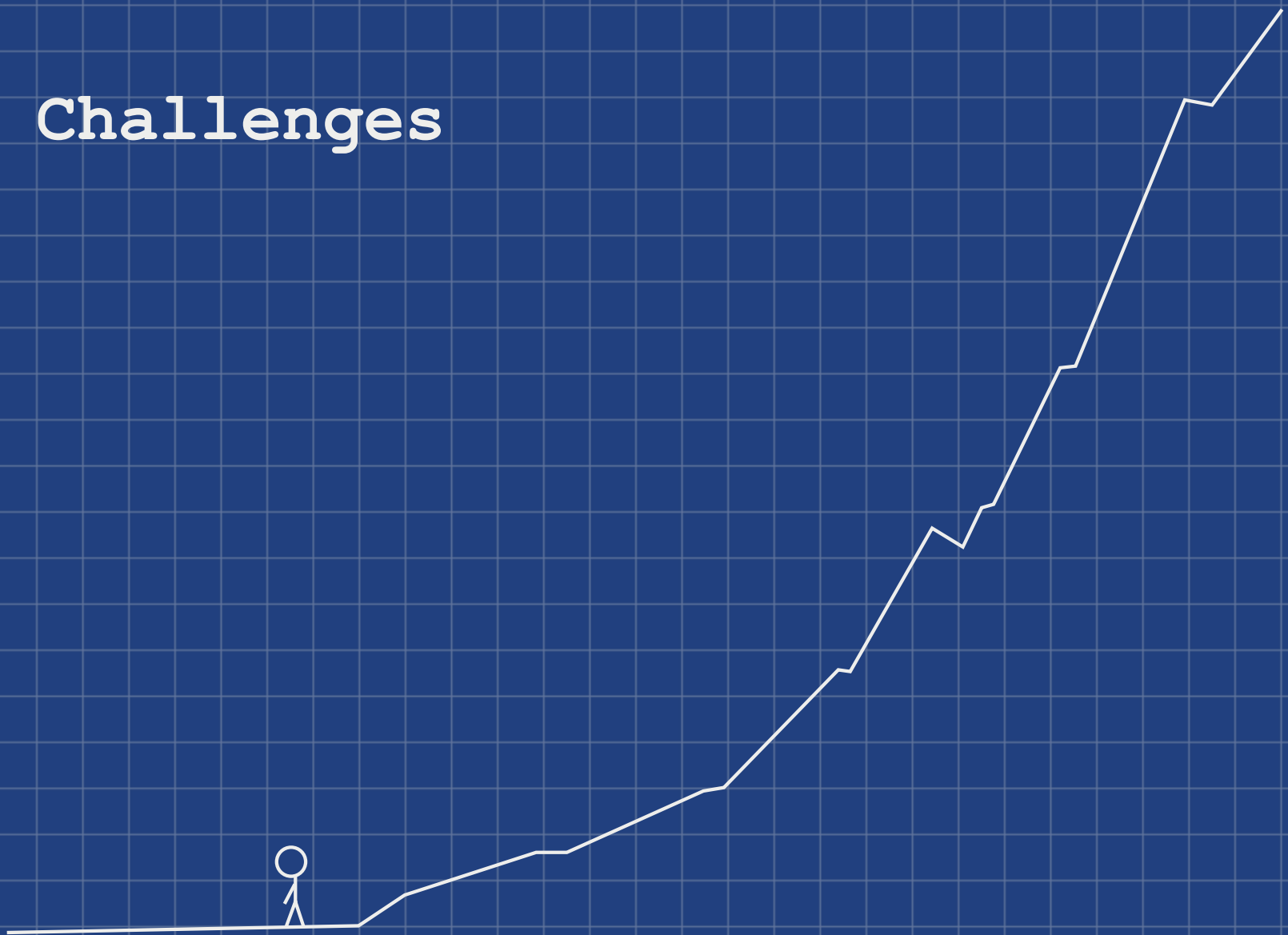
... and over 30 more

2 core developers

74 contributors

295 mailing list members

Challenges




Designing large distributable web applications is hard. Let's go shopping!

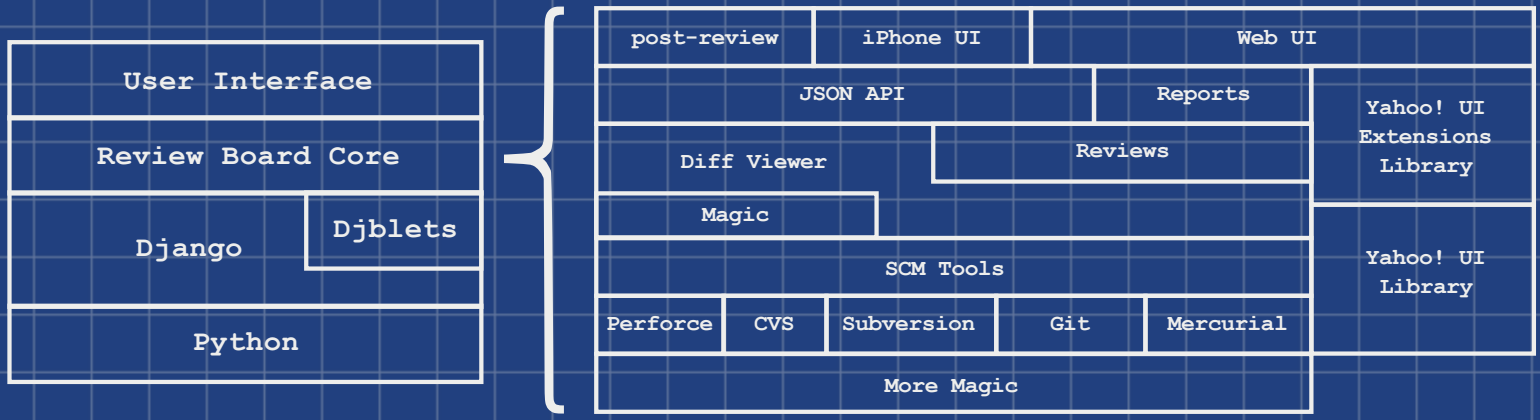
Installation Sucks!

- Writing server config files is error-prone
- Unfriendly errors if users forget to syncdb
- No good standard for database migrations
- Subdirectory installs should be effortless
- Many Django sites have site-specific and admin-specific media paths

Settings aren't easy enough

- No web UI for settings
- Changing any setting means downtime
- Users have to:
  1. SSH to the server
  2. Edit settings.py
  3. Restart the server
  4. Hope they didn't break anything

Large apps are complex



Fortunately, Django helps keep us organized.

4 groups of people
to make happy. :)

1 IT Guys want...

NIS, LDAP, etc.

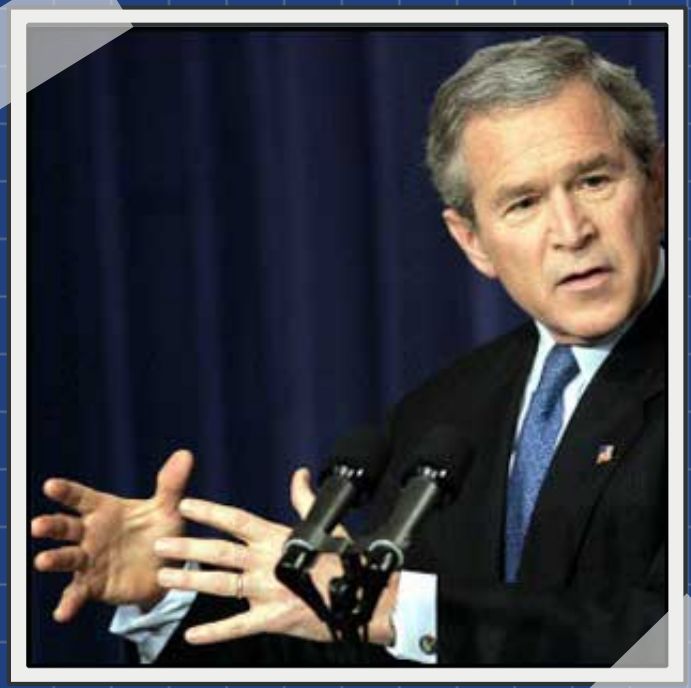
Very custom installs

Quickly modify settings

Keep things updated without crying

Full control over all e-mail support

2 Policy Deciders want...



To control who can access the site

To decide who can perform certain operations

To keep tabs on who has done what

3 Users want...

Customized views of everything

Instant access to everything

No interruptions while they're working

4 Other Developers want...

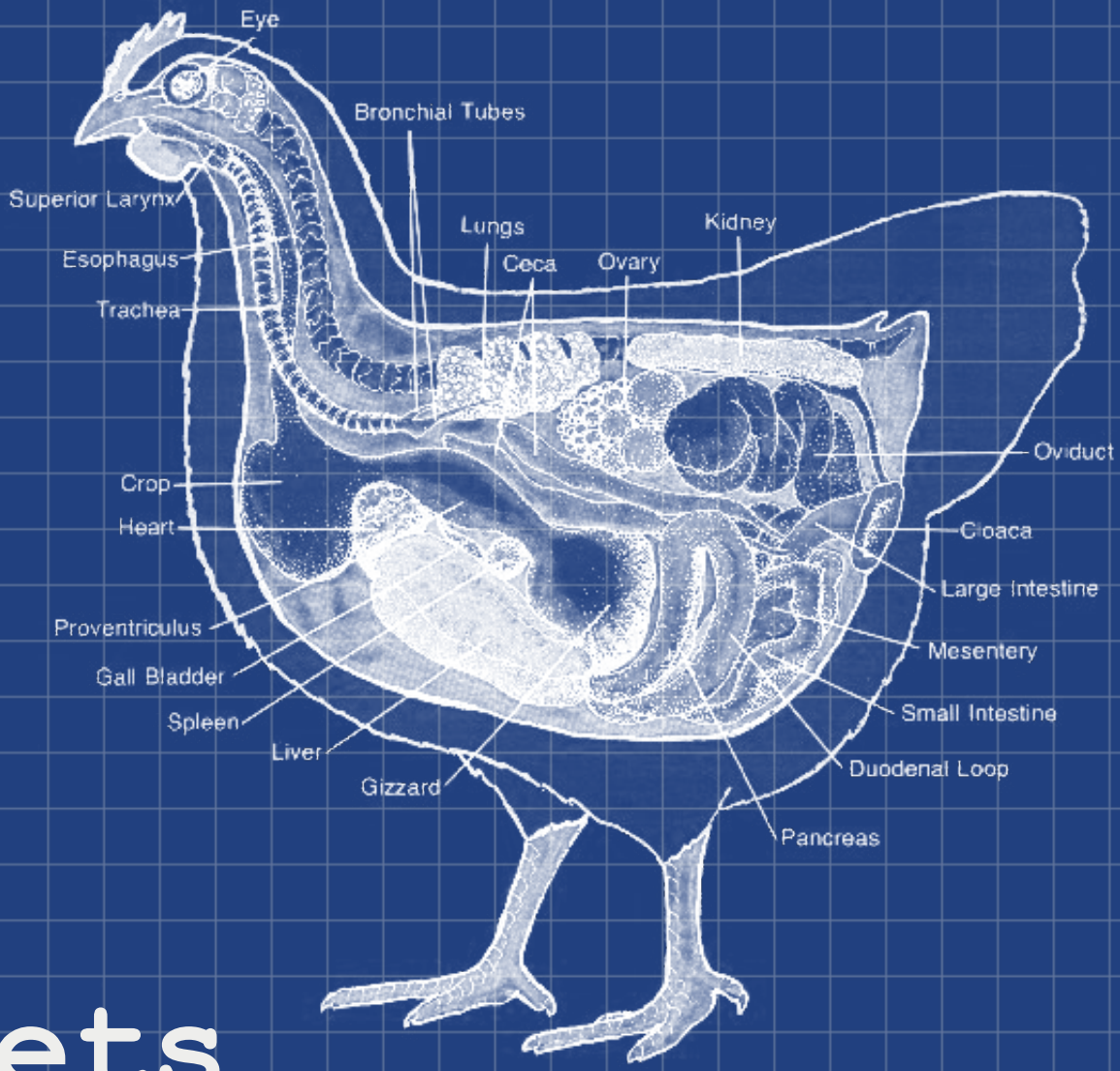
The ability to extend and script

A supported API to work with

Components they can reuse

Users are the most vocal.

Development


Eye
Bronchial Tubes
Superior Larynx
Esophagus
Trachea
Lungs
Ceca
Ovary
Kidney
Oviduct
Crop
Heart
Cloaca
Large Intestine
Proventriculus
Gall Bladder
Mesentery
Spleen
Small Intestine
Liver
Duodenal Loop
Gizzard
Pancreas

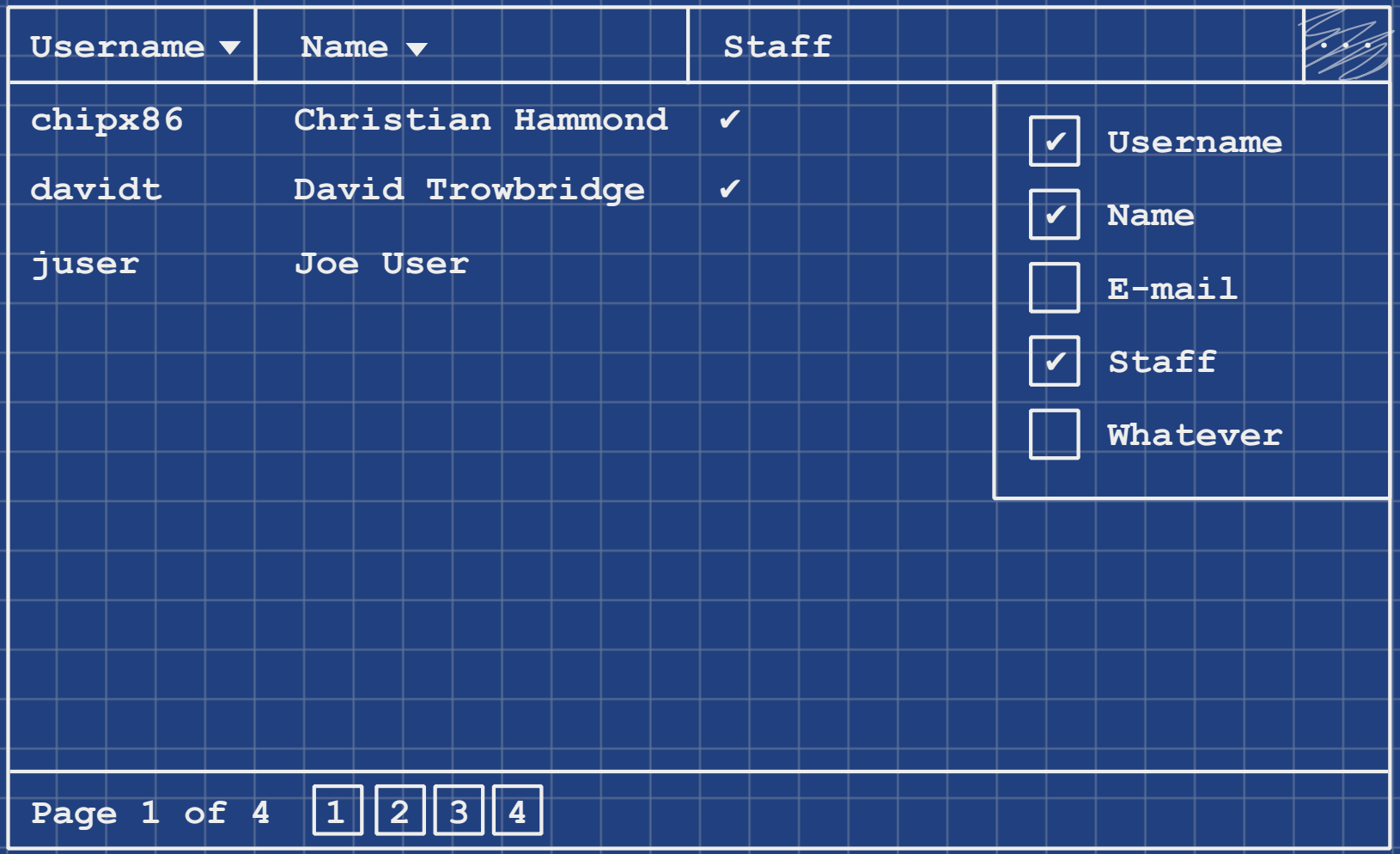
Djblets

datagrids

Username ▼	Name ▼	E-mail	...
chipx86	Christian Hammond	chipx86@chipx86.com	
davidt	David Trowbridge	trowbrds@gmail.com	
juser	Joe User	juser@example.com	

Page 1 of 4 1 2 3 4

datagrids


Username
Name
Staff
chipx86
Christian Hammond
✓
davidt
David Trowbridge
✓
juser
Joe User
✓ Username
✓ Name
E-mail
✓ Staff
Whatever
Page 1 of 4
1
2
3
4

datagrids

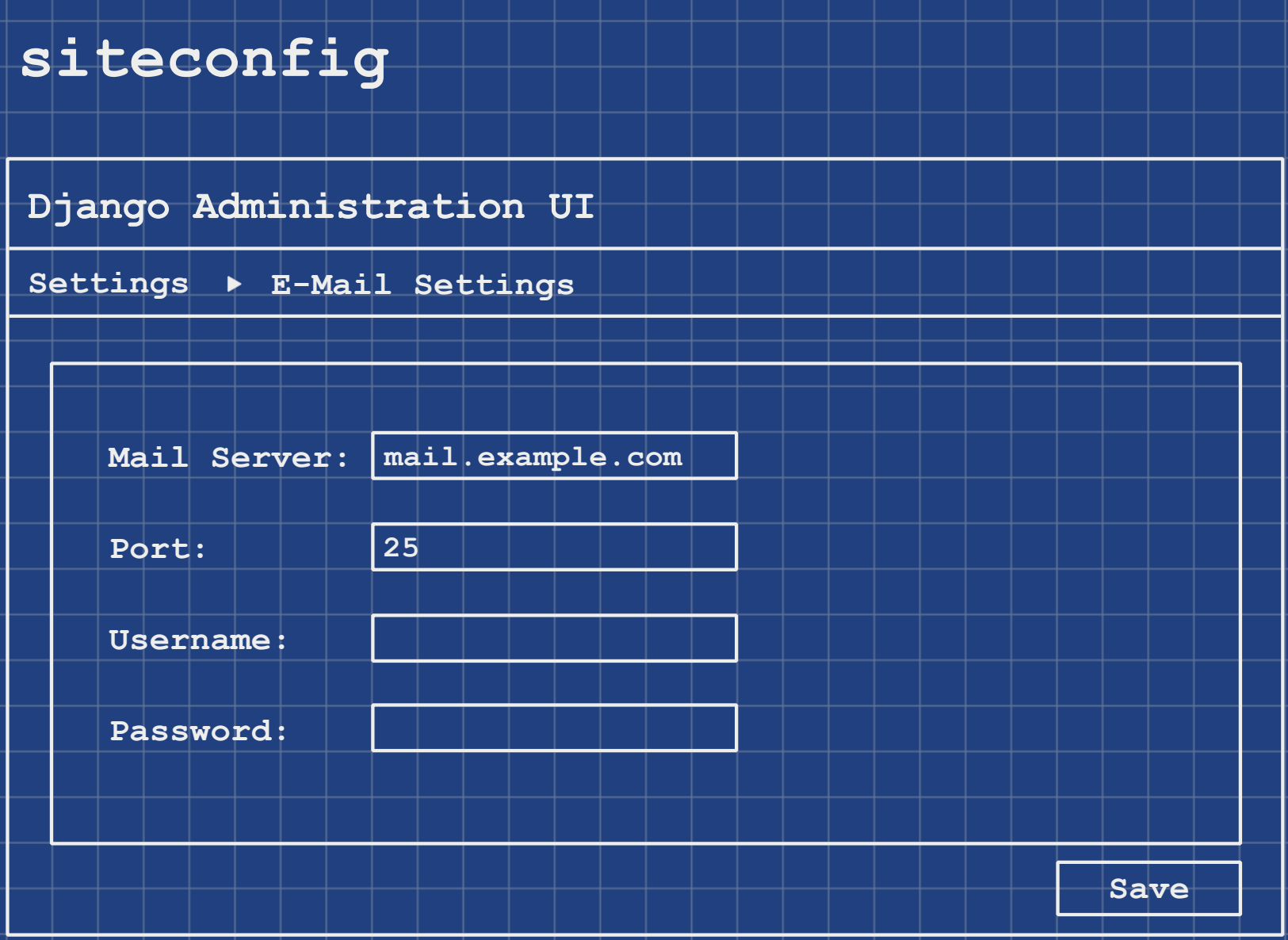
- Pagination
- Sortable columns
- Reorderable columns
- User chooses which columns to show
- Settings saved in the user's profile

datagrids

```
# datagrids.py
class UserGrid(DataGrid):
    username = Column("Username", link=True, sortable=True)
    name     = Column("Name", field_name="get_full_name",
                      link=True, expand=True)
    email    = Column("E-mail Address", sortable=True)
    is_staff = Column("Staff")

    def __init__(self, request):
        DataGrid.__init__(self, request, User.objects.all(),
                          "Users")
        self.default_sort = ["username"]
        self.default_columns = ["username", "name", "email"]


# views.py
def user_grid(request, template_name="myapp/common_grid.html"):
    return UserGrid(request).render_to_response(template_name)
```


siteconfig
Django Administration UI
Settings ▸ E-Mail Settings
Mail Server:
mail.example.com
Port:
25
Username:
Password:
Save

siteconfig

- Settings stored in database
- Serialized to JSON
- Syncs with Django settings
- Replaces settings.py for most settings
- Automatic admin UI generation
- Auto-scan for siteconfig.py files in third party apps (soon)

siteconfig

```
# forms.py
class EMailSettingsForm(SiteSettingsForm):
    mail_host          = forms.CharField("Mail Server")
    mail_port          = forms.IntegerField("Port")
    mail_host_user     = forms.CharField("Username")
    mail_host_password = forms.CharField("Password")

    class Meta:
        title = "E-Mail Settings"

# urls.py
urlpatterns = patterns('djblets.siteconfig.views',
    (r'^admin/settings/email/$', 'site_settings',
     {'form_class': EMailSettingsForm})
)

# some_file.py
def my_feature():
    siteconfig = SiteConfiguration.objects.get_current()

    if siteconfig.get("my_feature_enabled"):
        # Do something fantastic.
```

webapi

- Outputs JSON (soon XML)
- Nested serializers (apps can provide their own)
- Decorators for defining API handlers
- Response types for data, errors, form errors

webapi

```
# views.py
@webapi_login_required
def user(request, username):
    user = User.objects.get(username=username)

    if user:
        return WebAPIResponse(request, {'user': user})
    else:
        return WebAPIResponseError(request, DOES_NOT_EXIST)


# urls.py
urlpatterns = never_cache_patterns('',
    (r'^api/json/user/(?P<username>[A-Za-z0-9_-]+)/$',
     'myapp.views.user')
)
```

webapi

```
class BasicAPIEncoder(WebAPIEncoder):
    def encode(self, o):
        if isinstance(o, User):
            return {
                'id':         o.id,
                'username':   o.username,
                'first_name': o.first_name,
                'last_name':  o.last_name,
                'fullname':   o.get_full_name(),
                'email':      o.email,
                'url':        o.get_absolute_url(),
            }
        else:
            return super(WebAPIEncoder, self).encode(o)
```

thanks

http://www.review-board.org/

reviewboard@googlegroups.com

Furby picture - http://flickr.com/photos/kgantz/38060045/

Baby on Mac - http://flickr.com/photos/ntr23/730385992/