django-public-project Documentation

Release 0.5-alpha

Holger Drewes

August 21, 2014

Contents

1 Use Cases **3**

2 Features **5**

3 Manual **7**
3.1 Users 7
3.2 Developers 20

4 Ressources **29**

5 Indices and tables **31**

Django Public Project (DPP) is a specialised **content management system** written in `Python/Django` **for building an information website around big projects**. It is open source (BSD licence) and can be used by civic groups to monitor the progress of publicly funded projects but also from within governmental institutions itself to bring transparency in the proceeding of a project, inform the interested public about what is going on and thus enhance the acceptance of the project by the different stakeholders.

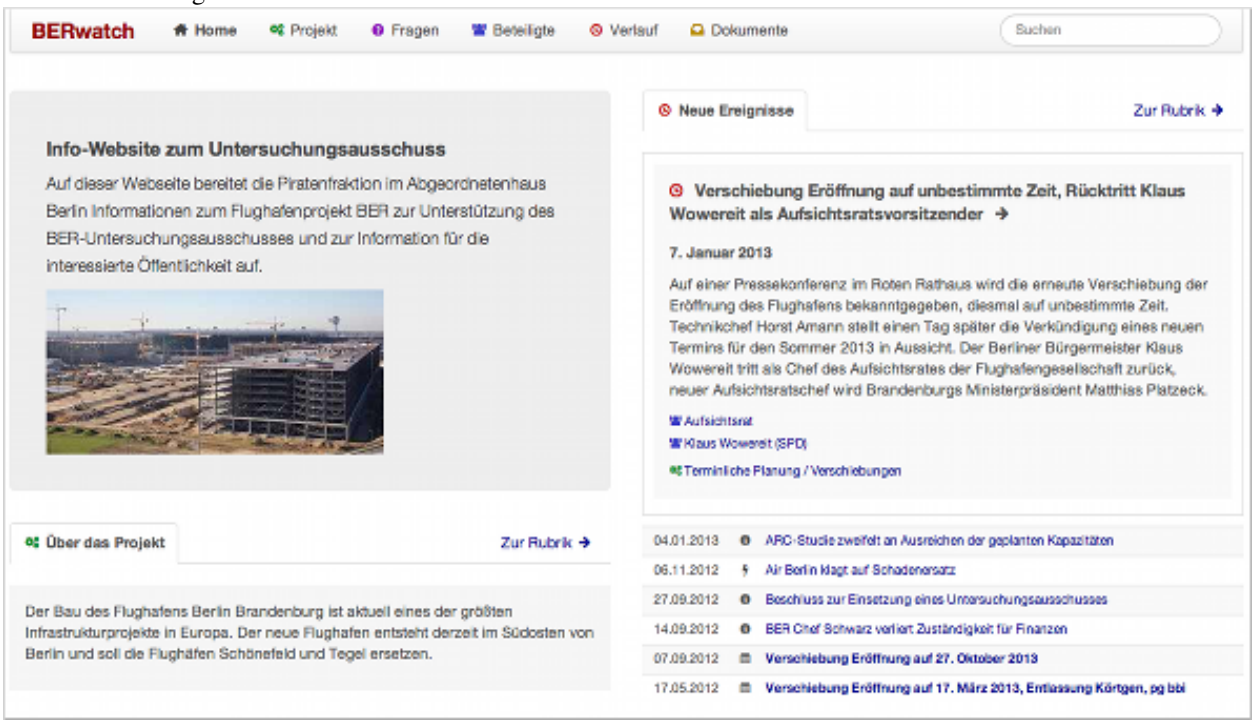
CHAPTER 1

Use Cases

This software was first used by the **pirate party in the parliament of Berlin** to monitor the progress of the **Berlin Brandenburg Airport project**, see http://ber.piratenfraktion-berlin.de to get an impression of the software (sorry, german).

Other possible use cases are:

- Construction/Infrastructure Projects
- IT/Software Projects
- Environmental Projects
- Law Making Processes



CHAPTER 2

Features

DPP provides an **admin interface** for entering/updating:

- A general project description
- The goals of a project
- The project structure
- A timeline of events
- The various actors and stakeholders
- Project documents
- Questions around the project
- Web sources wherever possible

On the **frontend side**, it lays out the project data in a clearly represented way, interlinks between the different project elements and provides:

- A dashboard on the main page bringing together the most relevant, current information
- Universal search
- A document viewer for pdfs
- User comments interlinking different project elements
- Relevant tag clouds for document content
- Supported languages: English (beta), German

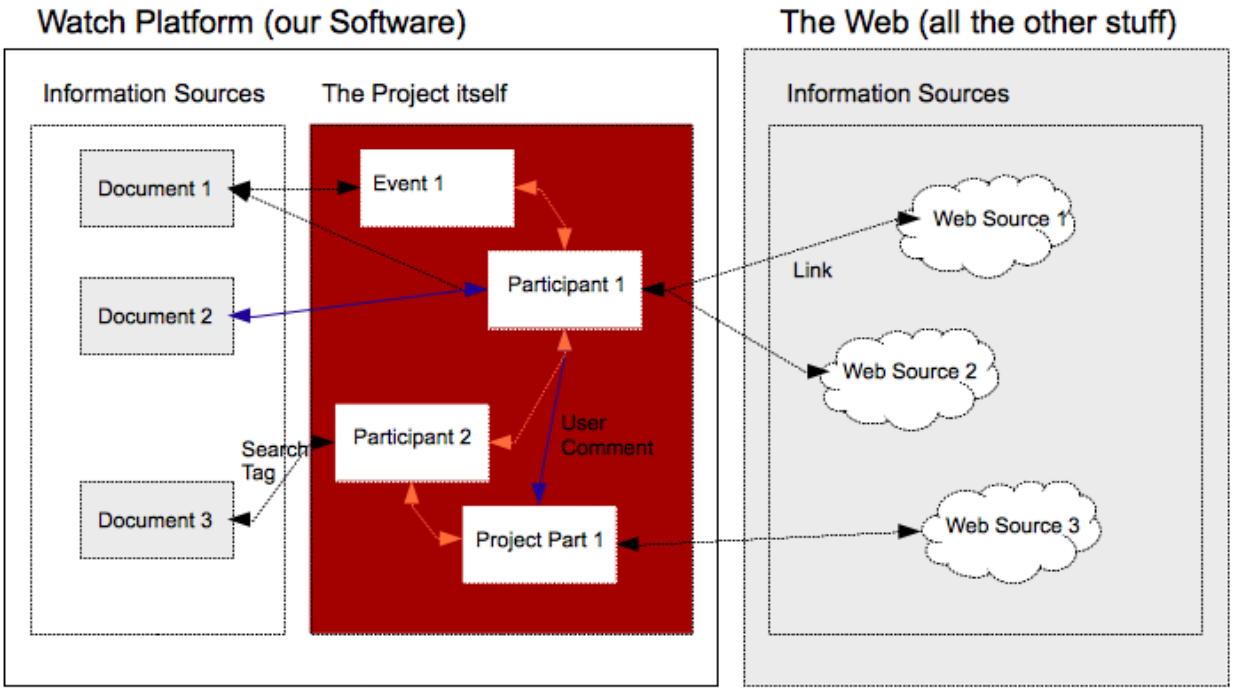
CHAPTER 3

Manual

3.1 Users

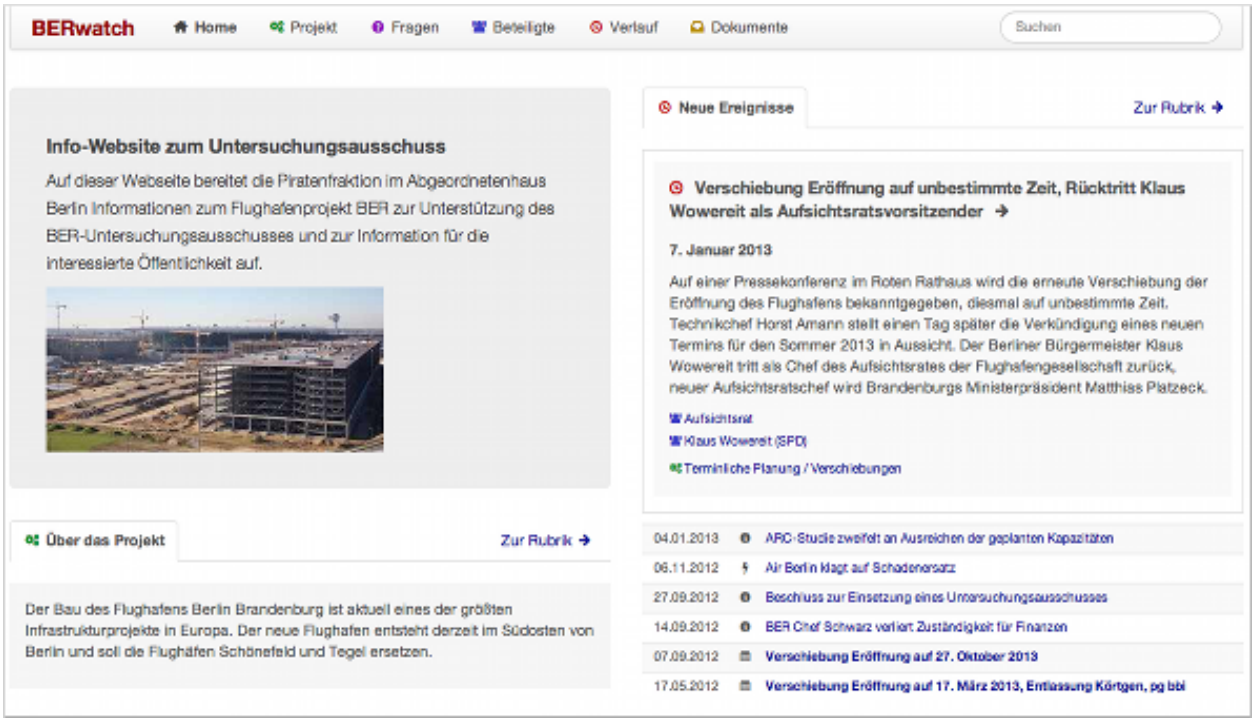
3.1.1 Introduction and Basic Concepts

Big projects are beasts and even for the people acting inside a project it is often hard to understand what is going on on a broader scale. For an interested public trying to follow the progress of a publicly funded project or to help out with local or specialized knowledge this is even harder. Lots of things are happening in different areas of the project, many actors are involved and lots of information sources with project results are produced and distributed over various places.

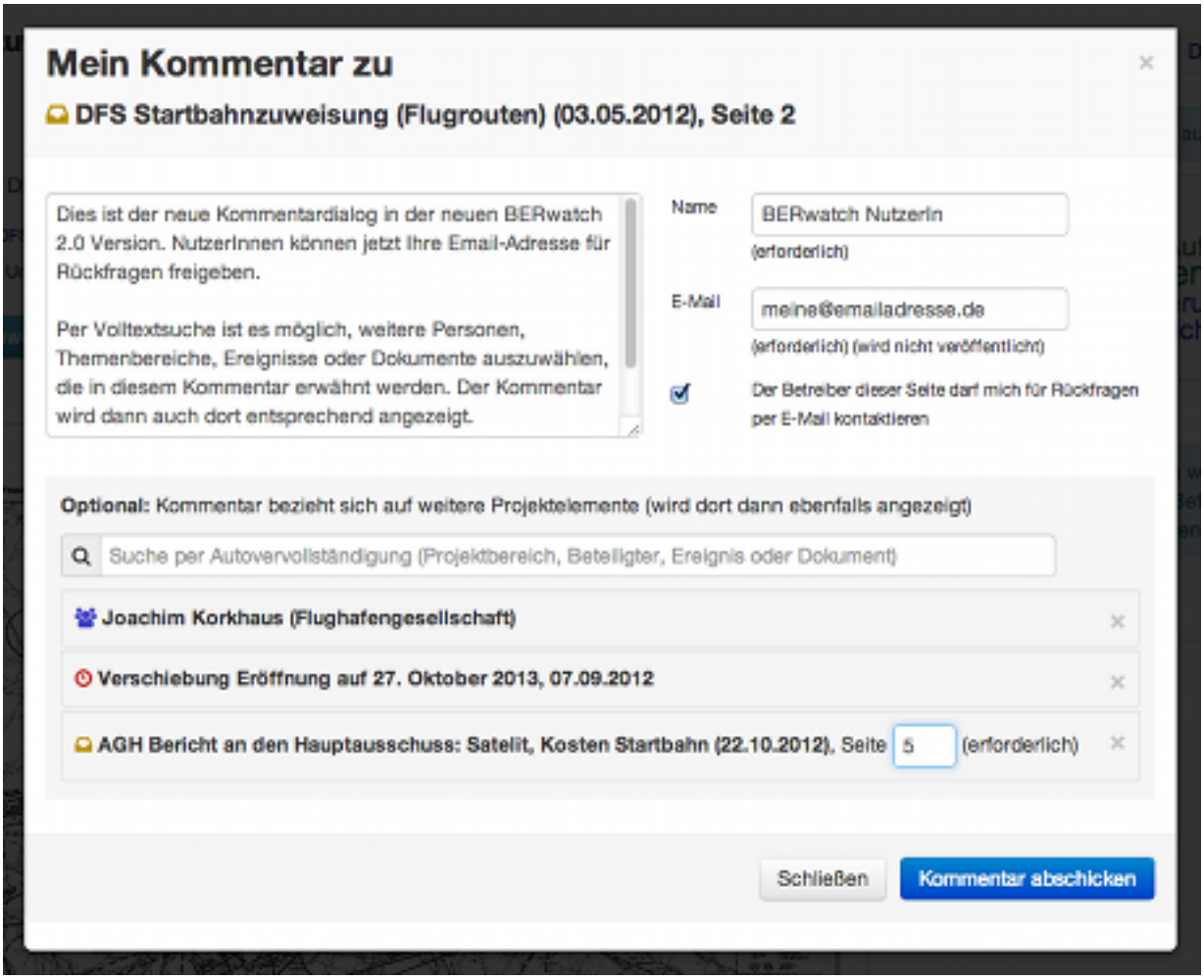


This software helps to build a central web platform around a project to make things more transparent. On the backend side it comes with an especially targeted admin interface. General information about the project, its basic structure and its goals can be provided and the admin user can enter information about events, participants and questions around the project. All this information is then layouted on the frontend website in a beautiful and suitable way.

The provided project data can be connected in various ways. Which events or questions belong to which project topics? Which participants are involved in which events? Which are the responsible actors in a project? All these interconnections are then used to structure the information on the frontend website and let people easily navigate between the different project elements.



In a project a lot of the information is often already existing, it is just not as accessible as it could (should) be. To tap into the various information sources which are available on the internet, web links can be provided for nearly every project element. Documents can be uploaded directly into DPP and connected to other project elements in various ways.



There is another type of information relevant for many public projects which cannot be reached by a Google search and is not even hidden in project documents: often people not directly involved in a project have knowledge about a special project topic, maybe because they work in a related field or they have got valuable knowledge about the local circumstances. To encourage people to share this knowledge, all project elements on the frontend website can be commented by the visiting users.

3.1.2 Where to reach your site

Before you can start you need a working DPP instance, see the *Developers* section for more about this. Since this is not as easy as a Wordpress installation, you will probably need to be or have an IT guy for this task. Like with a normal content management system the site then separates into a publicly reachable site, something like:

```
http://www.yourprojectwatchsite.org
```

and a corresponding admin site, something like:

```
http://www.yourprojectwatchsite.org/admin/
```

which should be secured by a username/password combination.

3.1.3 How to process

A focus of this software is to make it possible to define connections between different project elements. Due to the dependencies of the element types, it makes sense to keep a certain order when first entering project data. Otherwise one would jump back and forth all the time because some data is missing to be able to enter some other data.

When you have got access to the admin interface you first have to provide some basic site and project information. You can then define a structure for your project and use this structure when entering relevant project events and associated participants. Finally you can enter questions, upload documents and provide search tags for project parts, events and participants (leave them out when first entering these data types).

1. Provide basic site configuration and project description
2. Structure the project into topics
3. Enter events and participants
4. Enter questions around the project
5. Upload your documents and providing search tags

3.1.4 Basic Site Configuration and Project Description

Before you can actually see anything on the frontend website you have to provide some basic site configuration settings and project information. Before you are confronted with a screenshot like the following when opening the main url of your website:

Site Configuration

To configure some basic settings of your project website, you have to create a `SiteConfig` entry in your admin interface. The most important thing here is the definition of a title and a short variant of it beeing displayed on various parts of the site (generally, also the long title version should be relatively short). In the `SiteConfig` form you also provide some basic texts for your website like an intro text or a text for the contact page and you can change some layout settings like the color for your site titles. Some of the texts you enter can be formatted using HTML tags.

General Project Description

You provide some basic data about the project by adding a `Project` entry in the admin. The various descriptions you provide there are shown as introductory texts through out the website for the different categories.

Project Goals

Projects are normally done to reach something (if not, you should probably do another kind of website :-)). In DPP you can define a set of project goals as a `ProjectGoalGroup`. A `ProjectGoalGroup` consists of several single project goals, each having a performance figure, which can be freely defined. This can be e.g. the overall project costs, the date the project should be finished but also a short textual description of a goal ("provide village x with clean water"). You can define several `ProjectGoalGroups` if project goals change over time.

After having provided data for these three entries you should see a basic layout of your website when opening the main frontend site url in your browser.

3.1.5 Entering the Project Elements

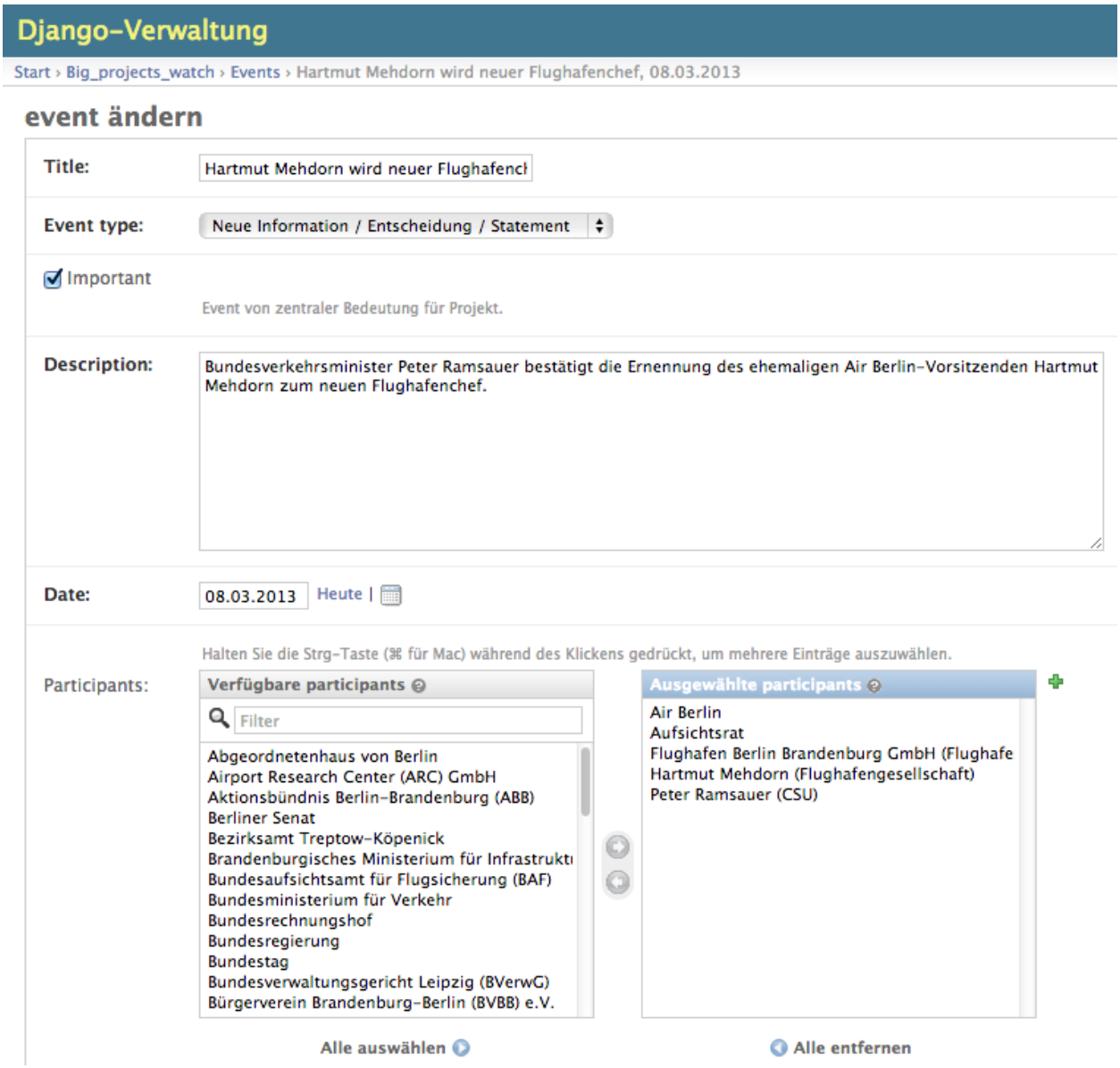
The definition of the different project elements is the heart of the system and you should take your time for it, think more in "days" than in "hours". If your system shall be useful you have to provide a certain density of information, and normally you have to do some investigate work for it. When you first enter your project elements, don't add search tags yet.

Define your Project Structure

Your project is structured by defining `ProjectParts` in the admin interface. Try to find 6-12 subject topics which are relevant for the project and cover the different project parts and provide a meaningful description for these topics.

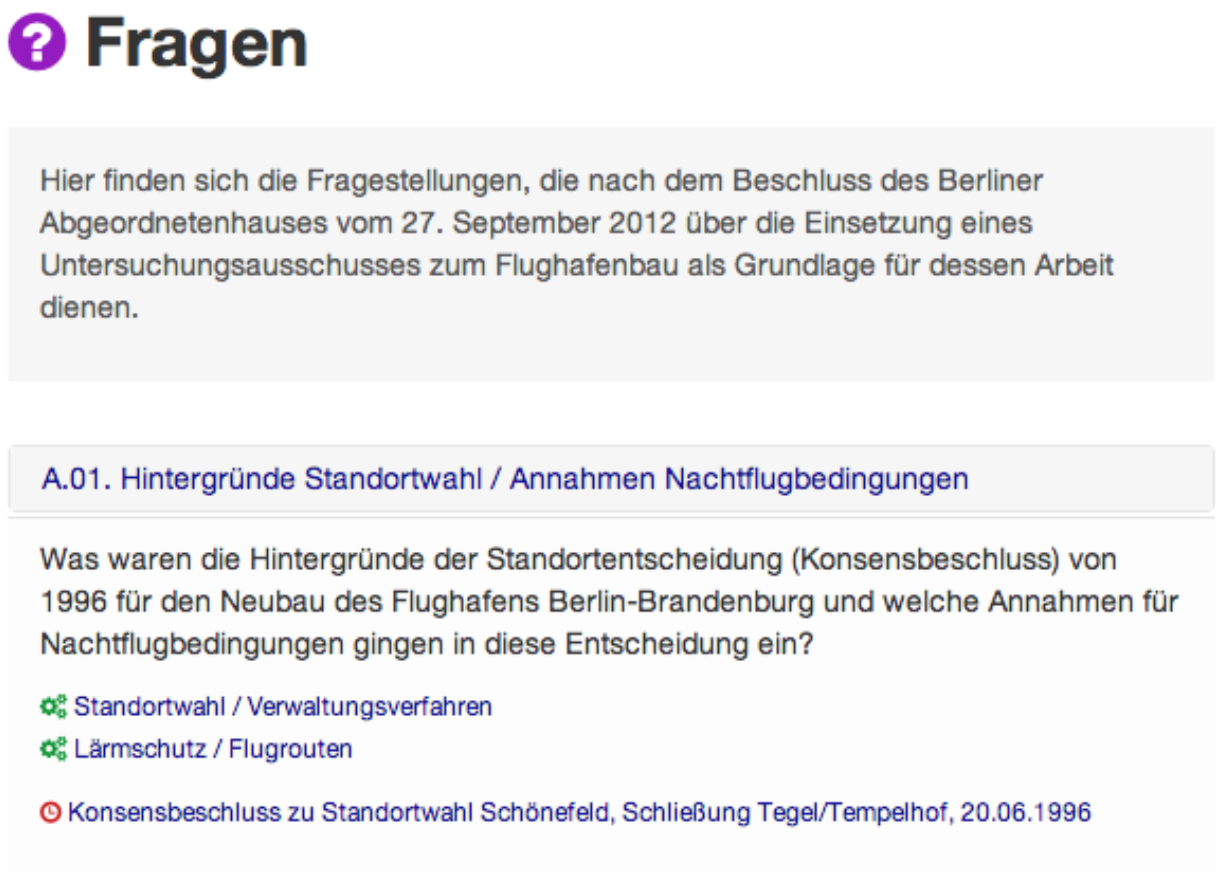
Enter Events and Participants

It is a bit difficult to think about all the relevant actors of a project, if you want to add them one after the other. So one strategy is here to start to reconstruct the relevant events around a project and add the participants alongside.



Questions around the Project

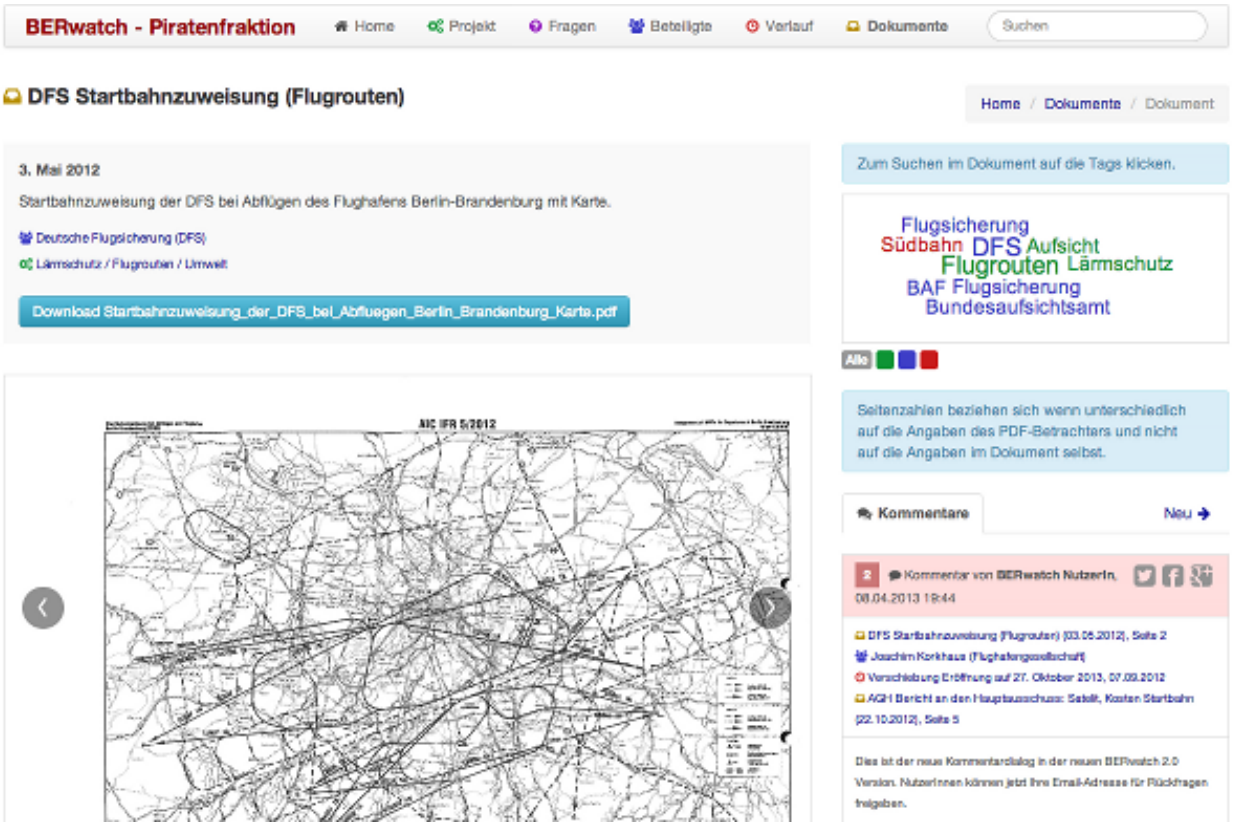
If you are running a monitoring website about a big publicly funded project, you have got probably questions like: What are the reasons that the project is running so late? Who was responsible for certain actions? If you are running a website from within an institution to inform the public about your project, you might have questions like: What do you think are the advantages for location A for our construction project? Where can we get information about xy? You can enter these questions in the admin and associate project parts, participants and events with them, they will then appear in the correspondig category on the website.



Uplad your Documents and providing Search Tags

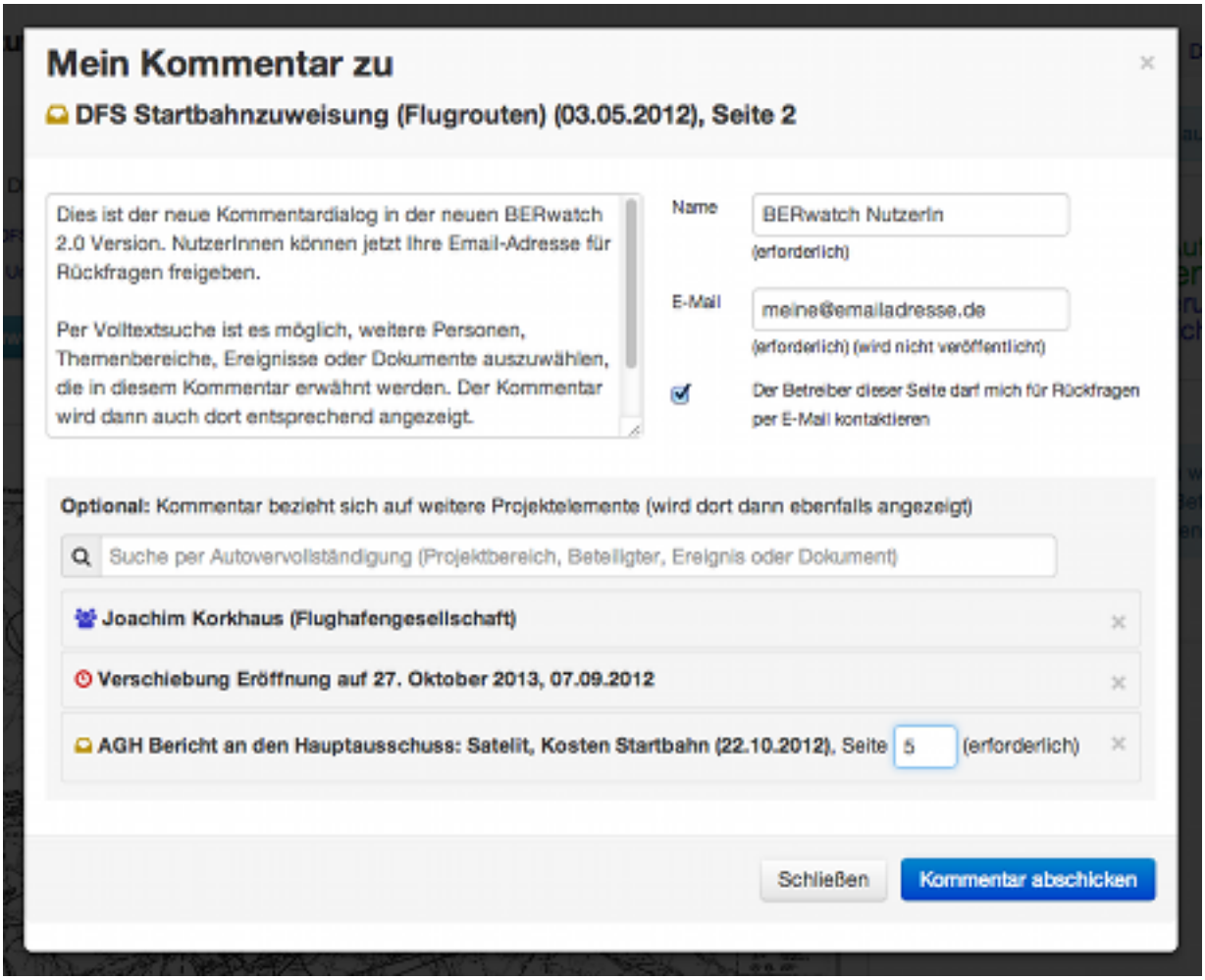
`Django Public Project` comes with an integrated PDF viewer directly on the website, where users can comment single document pages. You can upload documents about or generated in the project in the `Document` section of the admin and associate the documents with other project elements.

DPP comes with a mechanism to associate corresponding document content to different project elements called `SearchTags`. After you uploaded your documents you can enter these search tags for your project parts, events and participants. `SearchTags` are keywords, which do a document search. All documents found by a certain keyword for a project element are then associated with this element. For a project part like “Financing”, you could define search tags like “Credit”, “Budget” or “Investment”. For a participant often the name/last name of a person/institution is sufficient, e.g. “Obama” for Barack Obama (whichever project that is :-)). For an event, it’s a bit hard to find good search tags, don’t use to general terms there. Play around with the site search a bit to get a feeling for choosing relevant search tags.



3.1.6 User Comments

On DPP websites users can comment on nearly everything and connect their comments to various elements of the project. Comments are then shown alongside every connected element. All admin users having activated the “Receive new comment emails” in the admin user settings receive an email after someone sent a comment and can then activate the comment to be shown on the website.



3.1.7 Where to go from here?

These basic informations should help you to get started setting up a project watch website. Keep in mind that this is just the one half of the story.

Keep track of new events

For keeping your site useful you should keep pace with new events and actors and update your site regularly. Spend a similar amount of time on updating and enhancing the connections between elements like you spend on updating and adding new data.

Social Media

DPP sites live by the interaction with the users. It makes a lot of sense to run corresponding social media presences on Twitter, Facebook, Google+, App.net and the like and inform your users about new events, participants, ... in the system, ask them questions, encourage them to comment on certain documents and so on. There will be more features to support these kind of things in future releases of this software.

3.2 Developers

3.2.1 Introduction

DPP comes as a `Django` app providing all the data models necessary and the templates for the front end layout. This app can be integrated in a `Django` project representing the concrete project to be targeted.

Warning: This software currently is in `alpha` status. It can be used in a productive environment, but await sudden changes of Django model declarations, software dependencies or config settings.

3.2.2 Requirements

You need the following `Python/Django` libraries, probably best installed in an own `virtualenv` environment:

- Python 2.7+ (earlier versions untested)
- Django 1.5+
- PDFMiner
- Python Image Library PIL 1.1.7+ (for Django ImageField type)
- Tastypie 0.9+ (for API access)
- South 0.7.6+ (to keep track with DB changes)

For PDF conversion to jpg files for having an IE compatible PDF viewer, you need to have the `ImageMagick` library with the `convert` command installed in your shell environment:

- ImageMagick (convert tool)

3.2.3 Installation

Installation with Pip

`DPP` is now on the `Python Package Index` and you can install the software with all dependencies with:

```
pip install django-public-project
```

Manual Installation

If you want to have the latest version of `DPP`, you can install the sources manually with `PIP` (or directly clone the GitHub repository):

```
pip install -e git+https://github.com/holgerd77/django-public-project.git@master#egg=django-public-pr
```

Then install the requirements above. There is a `requirements.txt` file in the main directory of the repository you can use:

```
pip install -r requirements.txt
```

Project Creation

Create your `Django` project:

```
django-admin.py startproject myprojectwatch
```

Add the Django apps installed to your `settings.py` file (of course you also need the admin app which is essential for DPP):

```
INSTALLED_APPS = (
    ...
    'django.contrib.admin',
    'tastypie',
    'public_project',
    'south',
)
```

Sync your database respectively use migrations for DPP:

```
python manage.py syncdb (due to database dependencies, don't create a superuser yet)
python manage.py migrate
python manage.py createsuperuser
```

3.2.4 Configuration

DPP is not really an app which you would install beside many other Django apps and integrate it in a more complex website. It is more a content management system already coming with an url structure and a given layout capsuled in a single app. So DPP takes control of more things than the normal Django app.

URL structure

The urlpatterns for your project are completely coming from DPP, with an exception of the admin url, which should be adoptable for security reasons. So your minimal urls.py should look similar to this, importing the main url patterns from `public_project.urls`:

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

from public_project.urls import urlpatterns

urlpatterns += patterns('',
    url(r'^admin/', include(admin.site.urls)),
)
```

Now you should be able to enter the admin view when you start a dev server. The site itself is not yet ready for prime time at this moment.

Basic settings

Since I'm not sure, if there are still some static references to static or media files somewhere in the code, you should use the following `STATIC_URL` and `MEDIA_URL` settings:

```
MEDIA_URL = '/media/'
STATIC_URL = '/static/'
```

For being able to get email notifications about comments and document relations, you need to configure the Django email settings properly:

```
EMAIL_FROM = 'admin@yourmailaccount.com'
EMAIL_HOST = 'smtp.yoursmtpserver.com'
EMAIL_HOST_USER =  'YOURUSERNAME'
EMAIL_HOST_PASSWORD = 'YOURSECUREPASSWORD'
```

DPP uses the request template context processor in its views, so add it to the `settings.py` file:

```
TEMPLATE_CONTEXT_PROCESSORS = (
         "django.contrib.auth.context_processors.auth",
         "django.core.context_processors.debug",
         "django.core.context_processors.i18n",
         "django.core.context_processors.media",
         "django.core.context_processors.static",
```

```
        "django.core.context_processors.tz",
        "django.contrib.messages.context_processors.messages",
        "django.core.context_processors.request", #this line!
)
```

And finally you need to enter a correct domain name (no leading `http://`) for your Site app in the Django admin. This is for links in mails to work properly.

Language Selection

At the moment DPP supports the following languages:

- English (en) (experimental and not yet used in production, probably you have to correct some stuff)
- German (de)

The language is chosen depending on the `LANGUAGE_CODE` param in the `settings.py` module, e.g.:

```
LANGUAGE_CODE = 'de-de'
```

Document upload/viewer

The document viewer in DPP is based on the Mozilla pdf.js library (included in DPP) when using modern browsers like Google Chrome, Firefox or Safari.

For being able to view on site pdf documents with the Microsoft Internet Explorer there exists a basic alternative pdf viewer. For this viewer, single pages are converted to png files and are stored on disk and you need to have the `ImageMagick` library installed and make sure that the `convert` command from this library can be used from within your project path.

Since this approach can take a lot of disk space for large documents and root access to the server is needed, you have to activate IE compatible pdf viewer usage with the following setting in your `settings.py` file:

```
DPP_IE_COMPATIBLE_PDF_VIEWER = True
```

If this setting is set to false (default) a warning message will be shown on the document page for IE users, prompting them to use an alternative browser.

If this setting is set to true, documents are saved as the original pdf file and a corresponding document_x folder containing the pngs in your media folder. Please test-upload a pdf document and see if these files are generated. Then test the url with the pdf viewer for this document in both the MSIE and another browser.

Note: The conversion process of a pdf document takes place in the background and may take a while for large documents.

Initial project data

For the site to be properly displayed, you have to enter some initial project data. When you open the main url of your dev server, you should see a message similar to the following. Please follow the instructions.

JSON API

Since `v.0.4` DPP comes with a public API, which let developers access the public data of the system, leaving out internal comments and user comments. The API supports no authentication mechanism yet and will be accessible by everyone without limitation. To activate the API, add the following to your `settings.py` file:

```
DPP_PUBLIC_API = True
```

For the API to work you have got to have Tastypie 0.9.15+ installed:

```
pip install django-tastypie
```

And add `tastypie` to your `INSTALLED_APPS`.

When the API is working there will be an extra link in the footer leading to to API overview page:

```
http://yourproject.org/api/
```

Note: The API is still in an experimental/early stage, many features are missing and usage params will probably change in the future.

3.2.5 How to contribute: Translation

General How-To

The main area for contribution for this project is translation, since the scope of the software is relatively wide. So if you have got some time, speak English as a base language and another language like Spanish, Russian, French,... you are very welcome to help out (you don't need to be a developer for this task)!

You find the basic english language file called `django.po` on the DPP GitHub Page in the following folder:

```
public_project/locale/en/LC_MESSAGES/
```

Open this file and copy its contents. Then write the translation of the `msg` id strings between the double quotes after the `msstr` attribute. For longer strings you can use a format like this:

```
#: models.py:123
msgid "Structural parts of the project being stable over time."
msgstr ""
"Structural parts of the project being stable over time, e.g. 'Terminals', "
"'Gates', 'Traffic Control', 'Integration of Public Transportation', not too "
"much (<10), often useful as well: one entry for the project as a whole."
```

Just replace the `msgstr` with the translation in your language. If there is already a `msgstr` in english in the `django.po` file, use this string as a translation basis instead of `msgid` and replace the english string with your language translation.

When you are ready with your translation open an issue on GitHub and past your text there or (advanced developer version) make a pull request.

Note: If you have got limited time: please choose accuracy over speed, it's more helpful if you translate 20 strings in an appropriate manner and take some time to think about the translation than translating 50 strings and often missing the context or have spelling errors!

Generating/compiling message files

For generating the message files for a specific locale from the source identifiers, change to the `public_project` app directory and generate the message file for the desired locale with:

```
django-admin.py makemessages -l de
```

Then translate the missing identifier strings and compile the message files with:

```
django-admin.py compilemessages
```

3.2.6 Running the Test Suite

Tests for `DPP` are organized in an own django project called `bpw_tests` located under the folder `tests`. For running the test suite the following libraries are required:

- Selenium 2.3+
- django-dynamic-fixture 1.6+

The following types of tests are implemented:

Test Server

A test server is necessary to run some of the tests (e.g. testing RSS feeds). The test server can be started with:

```
./testserver.sh
```

Browser/Selenium Tests

The purpose of Selenium tests is to test the front-end functionality of the site. Tests are organized in the app `browser` and can be run from within the `tests` directory with:

```
python manage.py test browser #whole test suite
python manage.py test browser.GenericTest #one test case
python manage.py test browser.GenericTest.test_main_page #a single test method
```

Testing the 404 Template

When `DEBUG` is set to `True` in `settings.py`, `404 template` can be tested via the following url:

`http://yourdevelopmenturl/404test/`

3.2.7 Release Notes

Changes in version 0.5-alpha (Renaming Release) (2013-05-27)

This release is just for renaming the Django app. Due to the development of the software it came up, that the focus of the software is broader than actually thought, so the name `django-public-project` (DPP) is misleading and the software was renamed to `django-public-project` (DPP). This comes with a lot of hassle and won't happen again in the lifecyle of this software, but I felt, that in this early stage of the software, it is the only chance to make such a step.

If you already have a deployment of the software installed and have problems upgrading please contact me (@HolgerD77).

On `GitHub` the software moved to a new repository https://github.com/holgerd77/django-public-project with a new commit history. The `South` history has been restarted as well.

Steps to manually upgrade:

1. BACKUP YOUR DATABASE! BACKUP YOUR PROJECT FOLDER!
2. Create a JSON dump of your project with the `-n` option for preserving natural keys, leave out the `South` tables: `python manage.py dumpdata -n -e contenttypes -e auth.Permission -e south > bpw_dpp_dump.json`
3. Rename the suffix of `django-public-project` specific settings in `settings.py` from `BPW` to `DPP`
4. Remove `big_projects_watch` from `INSTALLED_APPS` in your `settings.py` file and add `public_project`.
5. Enter a new database name (for security reasons, leave old DB untouched) in your `settings.py`.
6. Run `python manage.py syncdb`, `python manage.py migrate`, don't create a superuser
7. Search and replace all occurrences of `big_projects_watch` in your JSON DB dump with `public_project` (e.g. in vi use ":%s/big_projects_watch/public_project/g"), keep a copy of the unmodified file!
8. Load your JSON dump in the new DB with `python manage.py loaddata yourjsonfile.json`.
9. Test your application. Sorry for the inconvenience.

Changes in version 0.4-alpha (2013-05-04)

- New **activity feed on main page**, integrating different activities in the system like an admin user adding a new object (e.g. a new event, participant, ...) or an visitor on the website commenting on an object. New model `ActivityLog` (see Migration 0016), activities are always bound to objects in the system, concept is flexible and expandable so that new activities around system objects can be added in the future
- **RSS feeds** for various pages of the system, closely connected to the activity concept. Feeds for the different new system objects, new comments on certain objects, a general activity feed, a general comment feed and a feed for new research requests (see further down)
- **Own pages/urls for questions, expanded editorial possibilities**: every question now has an own url and expanded possibilities to be described, new model fields for `Question` model class (see Migration 0018)

- **Integration of questions in system comments**: questions can now be referenced by site visitors in there comments and questions can be commented itself as well
- **New research requests associated with questions**: site owners can now give research requests to the crowd, describing tasks to be done or information to be found in documents. A research request is always associated with a question and can further - similar to comments - be associated with different system objects. Site admins can directly enter new requests on the associated question page.
- **Experimental version of a public API** Various objects in the system can now be accessed via a public JSON API if desired

Changes in version 0.3-alpha (2013-04-08)

- Layout overhall (category colors, bigger headlines, breadcrumb navigation, UI tweaks)
- `WITH_PUBLIC_DOCS` setting in `settings.py` replaced with `BPW_IE_COMPATIBLE_PDF_VIEWER` (see: *Introduction*)
- New detail info boxes for events, documents, used on main page to highlight newest events, documents
- Introduced search tags as new information concept (new DB models `SearchTag`, `SearchTagCacheEntry`, use `South` when upgrading): provided in Django admin for Events, Participants, ProjectParts, used for tag cloud generation and displaying documents containing these search tags on detail pages for Events, Participants, ProjectParts
- Search tag clouds (click induces search) on main page, document pages
- One unified crowdsource concept, merging the former concepts `DocumentRelations` into a broader `Comments` concept. ATTENTION! THESE CHANGES COME ALONG WITH HEAVY DB CHANGES AND NEED MANUAL WORK TO GET THINGS WORKING AGAIN!
  - When upgrading create a dump from your `DocumentRelation`, `Comment` table entries first
  - `DocumentRelation` model is completely removed, entries have to be manually copied into `Comment` table

Changes in version 0.2-alpha (2013-01-22)

- Layout based on Twitter Bootstrap
- Participants, ProjectParts, ProjectGoals, Events as basic project entities
- Modeling of questions around the project
- Document upload / PDF viewer based on pdf.js
- Crowdsourcing of comments / document relations

Changes in version 0.1-pre-alpha (2012-08-08)

- Initial verion

CHAPTER 4

Ressources

- Source Code on GitHub

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*