e-cidadania

e-democracy tool for citizens

Documentation

Release

Cidadania S. Coop. Galega

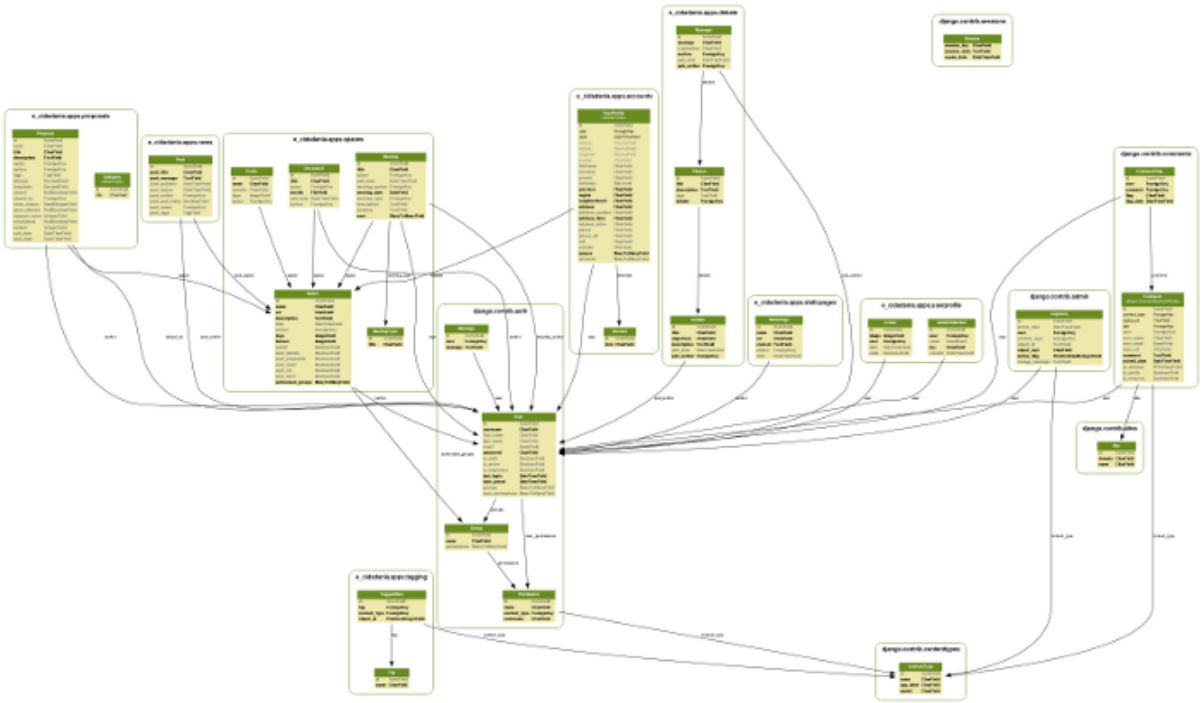
April 05, 2013

Contents

e-cidadania is an open-source e-democracy web tool intended for citizen participation.

e-cidadania was designed with complex participative processes in mind, lile a participative budget where people can decide what to do with a part of the annual bugets of a region or city, where making a simple meeting and disordered proposals doesn't meet the expectations.

This software is based on the django framework and some external libraries.



Warning: e-cidadania is in heavy development, because of that some parts of it may suffer changes, especially the data models and documentation until some releases.

CHAPTER 1

Installation & Configuration

1.1 Installation

e-cidadania installation is very simple and is done the same as any other django platforms.

1.1.1 Requirements

- Apache, nginx, or any other web server with CGI suppport
- FastCGI, CGI, Passenger or other CGI.

Dependencies

- Python 2.7 or later (not 3.x series)
- django 1.4
- PIL *(Python Imaging Library)*
- python-datetime *(version 1.5)*
- django-tagging
- django-grappelli
- feedparser
- pyyaml

You can install all the required dependencies automatically with this command:

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

1.1.2 Downloading platform

Official download page

Note: The download section in the official website is not available yet.

The are several ways to download e-cidadania. The most simple of them is going to the downloads page in the website and download the latest stable or development versions, ready to use.

GitHub packages

Other way of downloading it is through the Github downloads page, which autogenerates a .zip and .tar.gz files based on the repository tags. You can find it in:

:: https://github.com/cidadania/e-cidadania/downloads

From repository

See *development version*

Stable version

You can find the latest stable version in the download page in ecidadania.org:

```
http://ecidadania.org/en/downloads
```

Development version

Development version is available through various places. We use GIT as control version system, so you will have to install it in your computer.

GitHub *(official repository)*:

```
git clone git://github.com/oscarcp/e-cidadania.git
```

Gitorious: *(secondary repository)*:

```
git clone git://gitorious.org/e-cidadania/mainline.git
```

Repo.or.cz *(official mirror)*:

```
git clone git://repo.or.cz/e_cidadania.git
```

1.1.3 Installing

There is no proper installation process in e-cidadania, you just need to copy the *src/* directory to the folder where you want it to be, and after that configure in your CGI server how to execute it.

In the case you want to do some testing of the e-cidadania platform before getting into proper deployment, you just hace to copy the *src/* directory and execute the following commands inside it since e-cidadania comes by default configured in a development enviroment:

```
./manage.py syncdb # This will create all the database objects

./manage.py collectstatic # This will copy all the static content to *static/*

./manage.py runserver
```

This last command will execute the development server in the port 8000 of your machine, so you just need to type **localhost:8000" inside a web browser.

Now you can continue to :doc:'Configuration"

1.2 Configuration

The e-cidadania platform is almost ready-to-use after unpacking, but you will have to edit the *settings.py* file if you want to install it in production.

The e-cidadania setting have been splitted into three major files:

```
settings/
    defaults.py
    development.py
    production.py
```

1.2.1 Defaults.py

This file stablishes the common settings accross the both environments. Most of this settings are e-cidadania specific, we will specify here the ones you can modify safely.

SECRET_KEY (hash) It's **obligatory** to modify this value before deployment. Tis key is used for generating the CSRF and some other security features in django.

MEDIA_ROOT (directory) e-cidadania comes with this directory set by default for development but you will have to modify it to suit the directory where you media content or user uploaded content will be.

MEDIA_URL (URL slug) This will be the accesible URL for the media content. e-cidadania comes with it set to "uploads/" but you can modify it at any time.

STATIC_ROOT (directory) This directory works the same as MEDIA_ROOT but for static content (JavaScript, CSS, images, etc.). e-cidadania comes with it set by default for development but you will have to modify it to where you static content will be stored.

STATIC_URL (URL slug) This will be the accesible URL for the static content. e-cidadania comes with it set to "static/" but you can modify it at any time.

SELECT_ENVIROMENT (python import) This is not a proper value, but a python import. You will have to select between the two deployment files: development and production. The syntax is the same as every python import.

For development (default):: from e_cidadania.settings.development import *

For production:: from e_cidadania.settings.production import *

This are the main settign that you will have to modify to make the deployment of e-cidadania, you shouldn't need to modify the rest unless you want a very specific deployment.

1.2.2 development.py and production.py

development.py and *production.py* are minor configuration files with all the parameters we think you will need to make a development or production server.

Database

Configuring the database:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'db/sqlite.db',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
```

```
    }
}
```

First of all will be to set up the database. By default e-cidadania is set up to use a local SQLite3 database, which will be useful to make tests, but we don't recommend to use it in production.

An example of a database on a DreamHost shared server is this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'superb_database',
        'USER': 'databaseadmin',
        'PASSWORD': 'somepassword',
        'HOST': 'mysql.myserver.org',
        'PORT': '',
    }
}
```

DEBUG (Boolean True, False) The debug mode is meant for development only, it overrides some of the security systems of django and offers a detailed output when django crashes. This setting is meant to not be modified by the user.

TIMEZONE Time zone that e-cidadania will use as default.

LANGUAGE_CODE Main language that e-cidadania will use for the users. It uses the language code format, p.e. en-US (United States english)

CACHES Configure the cache backcend for django. Please refer to the django documentation so you can select a cache according to your needs.

ADMINS (python dictionary) List of *name*, *email* tuples with the administrators data. This is used in case e-cidadania has to send some report or django sends an error log.

EMAIL SETTINGS The email settings are pretty straightforward, so we will not explain them here.

> **Warning:** Django 1.4 still doesn't have support for SSL emails, you will need to use unsecure email addresses.

1.2.3 Other settings

The settings are not meant to be modified by the administrator, but he can change them if seen fit.

User profiles

ACCOUNT_ACTIVATION_DAYS (number)

This variable specifies how many days the user has to activate the user account since he receibes the confirmation e-mail.

1.2.4 Extensions

Note: This section is still on development.

Etensions are django applications that you can attach to e-cidadania to improve its functionalities.

1.3 Deployment

1.3.1 Apache 2

Note: This section is still on development.

1.3.2 nginx + uWSGI

Note: The installation through uwsgi works for other servers, not just nginx, since it's uWSGI who executes the e-cidadania instance while the server just serves the static content.

Warning: This section is still on development.

1.3.3 DreamHost

The instructions for deployment on a DreamHost shared server are too extensive to include here. You can find them in the Oscar Carballal blog

CHAPTER 2

Manuals

2.1 User manual

2.1.1 What is e-cidadania

2.1.2 How to register

The registry system will depend on what the administrator wants. By default e-cidadania allows user registration to perform some basic tasks and visit the public spaces, without participating.

To register in e-cidadania you must click “Login” and in the bottom you can find the registration link. It will ask your name, e-mail and password.

2.1.3 Participating in a process

How to participate in a space

Spaces

What are they?

How do they work

Proposals

How to send a proposal

Sending proposals is very easy. You just have to access to your participative space and click on “Proposals”, after that you need to click on “Add new proposal” on the right column.

A simple form will appear with several fields:

Title The proposal title. It must be a sinthesis of the proposal.

Description An extensive description of the proposal. You can link various external elements like images, websites, videos, etc.

Latitude/Longitude Right now e-cidadania has a simple geolocation sustem. If you want your proposal to be geolocated to need to put the coordinates.

Tags Tags make proposal identification easier, for example:

```
Title: The Apple Store sidewalk is broken
Tags: broken, sidewalk, apple, store
```

Debates

How to join a debate

The debates system is new, so pay attention to not get lost, this is not like an internet forum.

We have moved in-person debate models to this platform, and one of them is the one we're going to show you. You'll see how fast you get used!

Voting

News

Comments

File repository

To see or download any document you just have to click on the document and it will start downloading. If the document tou want doesn't appear in the list you just have to make click on "See all documents" and a page will come up with all the documents related to that participative process.

2.1.4 Frequently Asked Questions

If you have any frequently asked questions that doens't appear here please tell us!

2.2 Administration manual

This is a small introductory manual to teach you about how to use e-cidadania without messing everything up :).

2.2.1 Administration panel

2.2.2 Users

Restricting user registration

Note: This behaviour will change on e-cidadania 0.2

---

e-cidadania has a basic automated registration system that the administrator can activate or deactivate at his own will. By default, e-cidadania comes with registrarion activated, but if for some reason the administrator wants to block that registration system (passing to manual mode) he just has toactivate when he considers removing the comment symbol (sharp).

apps/userprofile/urls/en.py:107:

```
# url(r'^register/$', register, name='signup'),
```

If the platform is well set up, the registration system should take care of everything.

Permissions

Permissions in e-cidadania are inherited directly from the django auth system. This way we have group based permissions and user based permissions.

Note: e-cidadania 0.2 will have a row-level permission system.

There are four basic permissions:

Creation Depending on the moderation grade that you have been given, you can add simple content or more complex. The highest moderation levels have a very high detail grade when adding content.

Editing The editing task is similar to creation, it will return a formulary with the current data based on your permission level.

Deletion Usually in forums a moderator can delete user entries. In e-cidadania that is not the objective. User generated content must be preserved, it only can be deleted by platform administrators.

Move note This permission is for the debate system. IT allows or restricts the user capability of moving notes across the debate. The notes moving task is reserved for the space administrators.

Add/Remove

Groups

Groups are a massive way to give permissions to people. In this version groups will be a way to group people inside the spaces, except that for any reason you'll have to give them another permission for a specific task.

Adding/Removing groups

Adding people to groups

2.2.3 Spaces

What are they

How spaces work

2.2.4 Modules

Debates

Debate creation

Moderation

Proposals

Creating a proposal set

How to merge proposals

Voting

Creating a set

Moderation

2.2.5 Frequent errors

The most frequest errors are due to the server or a bad administrator management in the groups/permission case.

CHAPTER 3

Appearance / Themes

3.1 e-cidadania themes

3.1.1 Other themes

This is a little index of the e-cidadania themes.

3.1.2 Create your own theme for e-cidadania

Creating a new theme for e-cidadania can be hard, but not impossible. We will give you the precise instructions here.

3.1.3 Reference

e-cidadania themes are mostly located in the proper modules. Every application manages its own themes, not regarding the style, but the distribution.

Apart from this, there are some general templates located at the "templates" directory.

General

Note: Still on development.

User profile

Note: Still on development.

Proposals

Note: Still on development.

Debates

Note: Still on development.

News

Note: Still on development.

Documents

CHAPTER 4

Development

4.1 Style guide

The style guide establish a series of rules to follow when coding in e-cidadania. These rules are unbreakable. The style guide follows closely the PEP8 document, with some exceptions provided from the internal style guide at Pocoo.

4.1.1 Python

Imports Every import must be situated in the file header, below to the comment header. The python imports must precede any others, and the external librearies or third party modules must precede the application ones.

Example:

```
import os
import sys

from extlib import function

from myapp.module import function
```

Line width (columns) The code must be always 80 columns wide except on templates.

Long declarations If a code line does not fit in 80 columns, try to reduce it declaring variables previously. If it still can not fit, you can divide the lines this way:

Parentheses:

```
website = models.URLField(_('Website'), verify_exists=True,
                          max_length=200, null=True, blank=True,
                          help_text=_('The URL will be checked'))
```

Declararations:

```
this_is_a_very_long(function_call, 'with many parameters') \
    .that_returns_an_object_with_an_attribute

MyModel.query.filter(MyModel.scalar > 120) \
             .order_by(MyModel.name.desc()) \
             .limit(10)
```

Lists, tuples and dictionariess:

```
items = [
    'this is the first', 'set of items', 'with more items',
    'to come in this line', 'like this'
]

dict = {
    ('mobile': phone),
    ('car': key),
    ('another': thing),
}
```

Indentation Indentation must be *always* 4 spaces per level, no exceptions. You can not use tabs for indenting.

Blank lines Every classes and method must be separated by two blank lines. The code inside a class or method by one blank line.

Example:

```
class ListDocs(ListView):
    ----blank line----
    """
    List all documents stored whithin a space.
    """
    paginate_by = 25
    context_object_name = 'document_list'
    ----blank line----
    def get_queryset(self):
        place = get_object_or_404(Space, url=self.kwargs['space_name'])
        objects = Document.objects.all().filter(space=place.id).order_by('pub_date')
        return objects
    ----blank line----
    def get_context_data(self, **kwargs):
        context = super(ListDocs, self).get_context_data(**kwargs)
        context['get_place'] = get_object_or_404(Space, url=self.kwargs['space_name'])
        return context
----blank line----
----blank line----
def whatever(args):
    ----blank line----
    """
    A comment.
    """
    this_is_something = 0
```

4.1.2 HTML

Columns HTML code does not have a column limit, but it must be indented in a way we can locate easily every element inside the document. The indentation preceeds rendered results in application.

Indentation The X/HTML code must be indented with four spaces like python code, no exceptions.

Note: There may be some old code that follows the old indentation rule (2 spaces per level). If you see it we would be glad to receive a path to solve it.

4.1.3 CSS

Indentation Indentation will be 4 spaces, always, like Python code.

Example:

```
body {
    background: #FAFAFA;
          padding: 0;
          margin: 0;
          font-family: Verdana, "Lucida Sans", Arial;
          font-size: 1em;
          color: #000;
          cursor: default;
}
```

Colors Colors must be always written in hexadecimal. You are allowed to use three digits abbreviations.

Font size Font size must be declared in **em's** except for presentation requirement.

4.1.4 JavaScript

The JavaScript code follows the same rules from the python code.

4.2 User accounts

The user account system in e-cidadania is abased on the django *auth* module and in django-userprofile, created by James Bennet.

It might be that for other installations of e-cidadania you need other user data instead of the provided by the e-cidadania default installation. In that case you can modify the data model of the user.

The file containing the data model for the user profile is found in *apps/accounts/models.py*. All the fields can be replaced except *age* and *spaces*.

Note: Remember, after modifying the data model you will need to rebuild your database. We encourage you to have some application for database schema migration like *django-evolution*

Users are spearated in two parts. One of them is the django user account created by the *auth* module and the other is the *profile* object containing all the profile data for the user.

Users can be created without profile, but once you try to create a profile you will have to fill it and it will be linked to the user.

4.2.1 Public profiles

Users have a public zone in their profiles, which will show the data they want to show.

The public data will be visible to anyone visiting the profile, platform user or not.

Note: Be careful with the personal data you show on the net.

Warning: Currently the data on the public profile is not configurable by the users. This is expected to change in e-cidadania 0.1.5.

4.3 Creating modules

e-cidadania is extensible through modules, even if the installation procesdures are not working yet.

4.3.1 Structure

A module is basically a django application which we integrate in e-cidadania. At this time we advocate for the django default structure in the distribution and file names.

Warning: It's recommended to have expertise in django and python before creating a new module.

A module has three basic components, the data model, the view and the template.

Note: Discuss if is worthy to explain the creation of a module.

4.4 Generating the documentation

e-cidadania documentation is generated through sphinx (1.1) in three languages by default, which are:

- English
- Spanish
- Galician

The current structure of the documentation is:

```
docs/
    en/
    es/
    gl/
    reference/
    build/
    dev/ *
    docs/ *
    theming/ *
    index.rst *
    authors.rst *
```

The folders *en*, *es* and *gl* contain the documentation for each language. The *reference* folder is the autogenerated documentation from the code comments, which will always be in english and therefore is linked to all the languages to have just one copy of the reference instead of updating three.

The same occurs to the last folders and files. For compatibility reasons we left the "typical" structure of sphinx projects, although they are just links to the english documentation. This trick allows us to publish the documentation in places like *'Read The Docs <http://readthedocs.org>'_*

We divide the documentation in three big categories:

```
dev/     -- Development or administration related (installation, configuration and deployment)
docs/    -- Usual documentation: user manual, administrator manual, etc.
theming/ -- Theming related tutorials, index of themes, etc.
```

The documentation is done to be the most automatic possible. The three languages are generated at the same time when *make html* is executed.

Once that command is executed the documentation will be in the *build* folder which is divided this way:

```
build/
    doctrees/
    html/
        es/
        en/
        gl/
    latex/
```

```
es/
en/
gl/
```

The folders *es*, *en* and *gl* contain the documentation for their respective languages. The *doctrees* folder is just for reference when generating the documents.

As you can see, there are two formats of documentation: *latex* and *html*. e-cidadania has three officially supported formats for the documentation: html, latex and PDF (generated through pdflatex and stored in the *latex* directory).

Note: The way we designed the multilanguage system it should allow to build every format supported by sphinx (like epub, manpages, htmlhelp, etc.), but we don't guarantee it.

In the same way that you can build all the languages at once, you can also build them separately, executing the same *make* command but inside the language folder.

4.4.1 Linguistic rules

Not applicable to english translation at this time.

4.4.2 Fuzzy words

Not applicable.

4.5 Translations

For the translation we use two tools:

- django-rosetta
- gettext

Both ways of translating are simple thaks to the Django *middleware*.

Instead of making one global translation file, we decided to keep one transaltion file for every application, this way even if the modules change the translations will keep intact.

4.5.1 Translating with rosetta

To make translations with rosetta, you will need to have an account on the system and belong to the **'translators'** group. Once you have done that, the rest is simple.

Just access to the translation URL and the first you will see is a list with the languages available to translate.

Rosetta

Inicio › Language selection

Espanol

Application	Progress	Mensaxes	Translated	Fuzzy	Obsolete	File
Debate	100,00%	5	5	0	0	/home/oscar.carballal/devel/ec
Spaces	100,00%	63	63	0	0	/home/oscar.carballal/devel/ec
E_Cidadania	100,00%	65	65	0	0	/home/oscar.carballal/devel/ec
Accounts	100,00%	24	24	0	0	/home/oscar.carballal/devel/ec
News	100,00%	23	23	0	0	/home/oscar.carballal/devel/ec
Proposals	100,00%	20	20	0	0	/home/oscar.carballal/devel/ec
Userprofile	38,00%	375	145	0	0	/home/oscar.carballal/devel/ec

Inglés

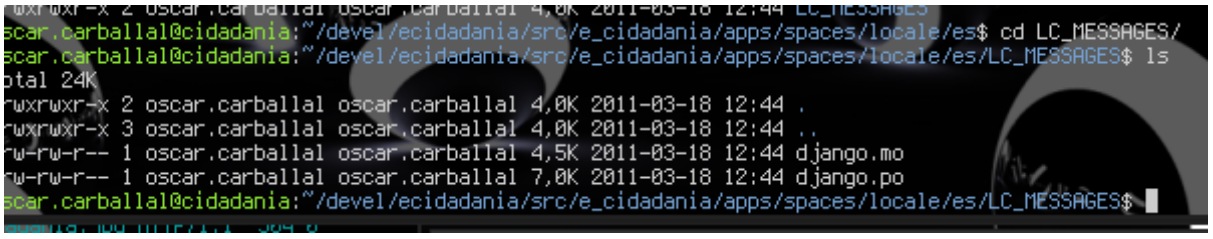
Application	Progress	Mensaxes	Translated	Fuzzy	Obsolete	File
Spaces	0,00%	27	0	0	0	/home/oscar.carballal/devel/ec
News	0,00%	9	0	0	0	/home/oscar.carballal/devel/ec
Userprofile	0,00%	375	0	0	0	/home/oscar.carballal/devel/ec

Click on the component you want to translate and start translating (the translation is done from english to other languages). If you find yourself stuck you can use the option "Suggest" which will make a query to the Google Translate database and write for you the translation.

Warning: Never trust the results of the "suggest" button. In most occasions will be incorrect.

4.5.2 Translating with gettext

Gettext is a well known tool by all the translators around the world. Its a standard. Thanks to the django *middleware* out work will be minimum, we only have to edit the .po files in the source files.



Instead of making one global translation, we decided to keep a translation file for every module, that way the translations will keep even if the modules are moved.

The location of the strings is usually a directory called **locale** inside the module. Inside it, you can find directories with the country code (en, es, us, gl, fr, etc.) and inside this one, the PO and MO files.

To translate, you must edit the PO file, which is a plain text file.

The MO file is the compiled version of the translation so the machine can read it to use it.

Warning: Stablish a workflow for translators and explain it here.

CHAPTER 5

Reference

5.1 apps.ecidadania.cal — Calendar and events

5.1.1 cal.models — Data models

The calendar module calls a version of Python HTML Calendar and adds some functions to use django objects with it.

The source code is based on the work of Eivind Uggedal <eivind@uggedal.com>

class apps.ecidadania.cal.models.**EventCalendar**(*LocaleHTMLCalendar*)
Event calendar is a basic calendar made with HTMLCalendar module and its instance LocaleHTMLCalendar for translation.

Attributes LocaleHTMLCalendar

Methods formatday, formatmonth, group_by_day, day_cell

day_cell(*cssclass*, *body*)
Create the day cell.

formatday(*day*, *weekday*)
Format the day cell with the current events for the day.

formatmonth(*year*, *month*)
Format the current month wuth the events.

group_by_day(*events*)
Group the returned events into their respective dates.

5.1.2 cal.views — Views

class apps.ecidadania.cal.views.**calendar**(*request*, *space_name*, *year*, *month*)
Returns an localized event calendar with all the Meeting objects.

Context calendar, nextmonth, prevmonth, get_place

Returns Localized HTML Calendar

5.2 apps.ecidadania.debate — Debates

5.2.1 debate.admin — Administration

5.2.2 debate.models — Data models

This file contains all the data models for the debate module.

class apps.ecidadania.debate.models.**Debate**(*models.Model*)
Debate object. In every space there can be unlimited debates, each one of them holds all the related notes. Debates are filtered by space. Start/End dates are for letting users use the debate or not. New in version 0.1b.

class apps.ecidadania.debate.models.**Note**(*models.Model*)
The most important object in every debate, the message. It has a coordinates value to determine the position of the note in its debate. New in version 0.1b.

5.2.3 debate.forms — Forms

This file contains all the forms for the debate modules.

class apps.ecidadania.debate.forms.**DebateForm**(*ModelForm*)
Returns an empty form for creating a new Debate.

Return type HTML Form

New in version 0.1b.

class apps.ecidadania.debate.forms.**NoteForm**(*ModelForm*)
Returns an HTML Form to create or edit a new 'note' or 'proposal' like it's called on the sociologists argot.

Return type HTML Form

New in version 0.1b.

5.2.4 debate.views — Views

These are the views that control the debates.

apps.ecidadania.debate.views.**add_new_debate**(*request*, *space_name*)
Create a new debate. This function returns two forms to create a complete debate, debate form and phases formset. New in version 0.1.5.

Attributes debate_form, row_formset, column_formset

Context form, rowform, colform, get_place, debateid

apps.ecidadania.debate.views.**get_debates**(*request*)
Get all debates and serve them through JSON.

apps.ecidadania.debate.views.**create_note**(*request*, *space_name*)
This function creates a new note inside the debate board. It receives the order from the createNote() AJAX function. To create the note first we create the note in the DB, and if successful we return some of its parameters to the debate board for the user. In case the petition had errors, we return the error message that will be shown by jsnotify. New in version 0.1.5.

apps.ecidadania.debate.views.**update_note**(*request*, *space_name*)
Updated the current note with the POST data. UpdateNoteForm is an incomplete form that doesn't handle some properties, only the important for the note editing.

apps.ecidadania.debate.views.**delete_note**(*request*, *space_name*)
Deletes a note object.

class apps.ecidadania.debate.views.**ViewDebate**(*DetailView*)
View a debate.

Context get_place, notes, columns, rows

get_context_data(***kwargs*)

class apps.ecidadania.debate.views.**ListDebates**(*ListView*)
Return a list of debates for the current space.

Context get_place

5.3 core.spaces — Spaces/Processes

5.3.1 spaces.admin — Administration

e-cidadania administration models for django-admin. This administration models will make their respective data models available for management.

class core.spaces.admin.**EntityAdmin**(*admin.ModelAdmin*)
Entities administration model.

List fields name, website, space

Search fields name

class core.spaces.admin.**EntityInline**(*admin.TabularInline*)
TabularInline view for entities.

model
alias of Entity

class core.spaces.admin.**SpaceAdmin**(*admin.ModelAdmin*)
Administration view for django admin to create spaces. The save() method is overriden to store automatically the author of the space.

List fields name, description, date

Search fields name

class core.spaces.admin.**DocumentAdmin**(*admin.ModelAdmin*)
Administration view to upload/modify documents. The save() method is overriden to store the author automatically.

List fields title, space, docfile, author, pub_date

Search fields title, space, author, pub_date

class core.spaces.admin.**EventAdmin**(*admin.ModelAdmin*)
Meetings administration model.

List fields title, space, meeting_date

Search fields title

5.3.2 spaces.models — Data models

class core.spaces.models.**Space**(*models.Model*)
Spaces model. This model stores a "space" or "place" also known as a participative process in reality. Every place has a minimum set of settings for customization.

class core.spaces.models.**Entity**(*models.Model*)
This model stores the name of the entities responsible for the creation of the space or supporting it.

class core.spaces.models.**Document** (*models.Model*)
This models stores documents for the space, like a document repository, There is no restriction in what a user can upload to the space

class core.spaces.models.**Event** (*models.Model*)
Meeting data model. Every space (process) has N meetings. This will keep record of the assistants, meeting name, etc.

5.3.3 spaces.forms — Forms

This module contains all the space related forms, including the forms for documents, meetings and entities. Most of the forms are directly generated from the data models.

class core.spaces.forms.**SpaceForm** (*ModelForm*)
Returns a form to create or edit a space. SpaceForm inherits all the fields from the Space data model.

Return type HTML Form

New in version 0.1.

class core.spaces.forms.**DocForm** (*ModelForm*)
Returns a form to create or edit a space related document, based on the spaces.Document data model.

Return type HTML Form

New in version 0.1.

class core.spaces.forms.**EventForm** (*ModelForm*)
Returns a form to create or edit a space related meeting, based on the spaces.Meeting data model.

Return type HTML Form

New in version 0.1.

EntityFormSet — modelformset_factory

5.3.4 spaces.views — Views

These are the views that control the spaces, meetings and documents.

General spaces views

class core.spaces.views.**ListPosts** (*ListView*)
Returns a list with all the posts attached to that space. It's similar to an archive, but without classification or filtering.

Return type Object list

Context post_list

Spaces views

class core.spaces.views.**GoToSpace** (*RedirectView*)
Sends the user to the selected space. This view only accepts GET petitions. GoToSpace is a django generic RedirectView.

Attributes **self.place** - Selected space object

Return type Redirect

class core.spaces.views.**ListSpaces** (*ListView*)
Return a list of spaces in the system (except private ones) using a generic view. The users associated to a private spaces will see it, but not the other private spaces. ListSpaces is a django generic ListView.

Return type Object list

Contexts object_list

class core.spaces.views.**ViewSpaceIndex**(*DetailView*)
Returns the index page for a space. The access to spaces is restricted and filtered in the get_object method. This view gathers information from all the configured modules in the space.

Attributes space_object, place

Return type Object

Context get_place, entities, documents, proposals, publication

class core.spaces.views.**DeleteSpace**(*DeleteView*)
Returns a confirmation page before deleting the space object completely. This does not delete the space related content. Only the site administrators can delete a space.

Return type Confirmation

core.spaces.views.**edit_space**(*request*, *space_name*)
Returns a form filled with the current space data to edit. Access to this view is restricted only to site and space administrators. The filter for space administrators is given by the edit_space permission and their belonging to that space.

Attributes

- place: current space intance.
- form: SpaceForm instance.
- form_uncommited: form instance before commiting to the DB, so we can modify the data.

Parameters **space_name** – Space URL

Return type HTML Form

Context form, get_place

core.spaces.views.**create_space**(*request*)
Returns a SpaceForm form to fill with data to create a new space. There is an attached EntityFormset to save the entities related to the space. Only site administrators are allowed to create spaces.

Attributes

- space_form: empty SpaceForm instance
- entity_forms: empty EntityFormSet

Return type Space object, multiple entity objects.

Context form, entityformset

Document views

class core.spaces.views.**ListDocs**(*ListView*)
Returns a list of documents attached to the current space.

Return type Object list

Context object_list, get_place

class core.spaces.views.**DeleteDocument**(*DeleteView*)
Returns a confirmation page before deleting the current document.

Return type Confirmation

Context get_place

class core.spaces.views.**AddDocument**(*FormView*)
Upload a new document and attach it to the current space.

Return type Object

Context form, get_place

class core.spaces.views.**EditDocument**(*UpdateView*)
Returns a DocForm filled with the current document data.

Return type HTML Form

Context doc, get_place

Meeting views

class core.spaces.views.**ListEvents**(*ListView*)
List all the events attached to a space.

Return type Object list

Context event_list, get_place

class core.spaces.views.**ViewEvent**(*DetailView*)
View the content of a event.

Return type Object

Context event, get_place

class core.spaces.views.**DeleteEvent**(*DeleteView*)
Returns a confirmation page before deleting the Meeting object.

Return type Confirmation

Context get_place

class core.spaces.views.**AddEvent**(*FormView*)
Returns an empty MeetingForm to create a new Meeting. Space and author fields are automatically filled with the request data.

Return type HTML Form

Context form, get_place

class core.spaces.views.**EditEvent**(*UpdateView*)
Returns a MeetingForm filled with the current Meeting data to be edited.

Return type HTML Form

Context event, get_place

5.4 apps.ecidadania.proposals — Proposals

5.4.1 proposals.admin — Administration

The proposal administration allows to edit every proposal made in the system.

class apps.ecidadania.proposals.admin.**ProposalAdmin**(*admin.ModelAdmin*)
Basic proposal administration interface since most of the work is done in the website.

List display title, author, tags

Search none:

5.4.2 proposals.models — Data models

Proposal data models are the ones to store the data inside the DB.

class apps.ecidadania.proposals.models.**Category**(*BaseClass*)
Dummy class for proposal categories. Inherits directly from BaseClass without adding any fields.

class apps.ecidadania.proposals.models.**Proposal**(*models.Model*)
Proposal data model. This will store the user proposal in a similar way that Stackoverflow does. Take in mind that this data model is very exhaustive because it covers the administrator and the user.

Automatically filled fields Space, Author, Pub_date, mod_date.

User filled fields Title, Description, Tags, Latitude, Longitude.

Admin fields (manual) Code, Closed, Close_reason, Anon_allowed, Refurbished, Budget.

Admin fields (auto) Closed_by

Extra permissions proposal_view

CLOSE_REASONS for :class:Proposal data model is hardcoded with four values, which will fit most of the requirements.

5.4.3 proposals.forms — Forms

Proposal forms.

class apps.ecidadania.proposals.forms.**ProposalForm**(*ModelForm*)
ProposalForm is a basic form autogenerated form for Proposal model.

5.4.4 proposals.views — Views

Proposal module views.

class apps.ecidadania.proposals.views.**ListProposals**(*ListView*)
List all proposals stored whithin a space. Inherits from django ListView generic view.

Return type Object list

Context proposal

class apps.ecidadania.proposals.views.**ViewProposal**(*DetailView*)
Detail view of a proposal. Inherits from django DetailView generic view.

Return type object

Context proposal

class apps.ecidadania.proposals.views.**DeleteProposal**(*DeleteView*)
Delete a proposal.

Return type Confirmation

Context get_place

class apps.ecidadania.proposals.views.**AddProposal**(*FormView*)
Create a new proposal.

Parameters space_name

Return type HTML Form

Context form, get_place

class apps.ecidadania.proposals.views.**EditProposal**(*UpdateView*)
The proposal can be edited by space and global admins, but also by their creator.

Return type HTML Form

Context get_place

Parameters space_name, prop_id

5.5 apps.ecidadania.news — News

5.5.1 news.admin — Administration

5.5.2 news.models — Data models

class apps.ecidadania.news.models.**Post**(*models.Model*)
Model of a news post.

5.5.3 news.forms — Forms

class apps.ecidadania.news.forms.**NewsForm**(*ModelForm*)

5.5.4 news.views — Views

class apps.ecidadania.news.views.**ViewPost**(*DetailView*)
View a specific post.

get_context_data(***kwargs*)
Get extra context data for the ViewPost view.

class apps.ecidadania.news.views.**AddPost**(*FormView*)
Create a new post. Only registered users belonging to a concrete group are allowed to create news. only site administrators will be able to post news in the index page.

Parameters space_name

Context get_place

class apps.ecidadania.news.views.**EditPost**(*UpdateView*)
Edit an existent post.

Parameters space_name, post_id

Context get_place

class apps.ecidadania.news.views.**DeletePost**(*DeleteView*)
Delete an existent post. Post deletion is only reserved to spaces administrators or site admins.

get_context_data(***kwargs*)
Get extra context data for the ViewPost view.

CHAPTER 6

Finale

6.1 Getting help

- Try looking the FAQ.
- e-cidadania mailing lists
  - ecidadania-users@freelists.org
  - ecidadania-dev@freelists.org
  - ecidadania-es@freelists.org
- Bug reporting in the e-cidadania bug tracker or contact us through e-mail at soporte@ecidadania.org

6.1.1 About this document

Authors Oscar Carballal <oscar.carballal@cidadania.coop> <info@oscarcp.com>

Corrections Ana Lorenzo Vila <ana.lorenzo@cidadania.coop>

Translations

Galician Armando Broz Fidalgo <armando.broz@cidadania.coop>

Rosa Muñiz Castro <rosa.munhiz@cidadania.coop>

Antón Meixome <meixome@certima.net>

Luisa Vicente Comojo <luisa.vicente@cidadania.coop>

Spanish Oscar Carballal Prego <oscar.carballal@cidadania.coop>

Thanks

- Algueirada
- Xunta de Galicia
- CESGA (Centro de Supercomputación de Galicia)
- Concello de Ferrol
- NovaCaixaGalicia
- Google

Python Module Index

a

c