Kansha Documentation

Release 2.0.0

Net-ng

Jun 21, 2017

Contents

Kansha is an open source web application to manage and share collaborative scrum boards and more.

CHAPTER 1

Contents

Kansha installation

Quickstart guide

There are two ways to install the latest stable release of Kansha:

1. By using Docker and the public image we maintain on the Docker Registry;
2. Or by directly installing the python package on your computer.

The second method is the preferred one for production sites (for now). You can also install older stable versions with that method.

Docker image

This section assumes your are already familiar with Docker, you have a running Docker daemon somewhere and your *docker* command is correctly set up.

For the impatient

Just type:

```
$ docker run -p 8080:8080 netng/kansha
```

Now point your browser to http://localhost:8080 and enjoy!

If you are using Boot2docker or Docker Machine (Windows/MacOS), replace `localhost` *above by the IP address of your virtual machine.*

Update your local image

If we release a new image after you executed the above command, you won't be able to run it unless you explicitly update your local image:

```
$  docker pull netng/kansha
```

The provided image is not usable (yet) for production sites, but you can build your own. For that purpose, we provide the Dockerfile used to produce the public image on GitHub. Then you have to adapt it. See *Production setup* for details.

Python package

The following instructions apply to UNIX-like systems, like Linux or MacOS X.

Installation

Nagare, the framework used by Kansha, needs Stackless Python (version 2.7.X) to run.

Unfortunatly, none of the major Linux distributions offer packages for Stackless, so you have to build it from sources.

In order to install it via the sources, first ensure you have the prerequisite system dependencies, then complete the following commands:

```
$ mkdir <STACKLESS_DIR>
$ wget http://www.stackless.com/binaries/stackless-278-export.tar.bz2
$ tar xf stackless-278-export.tar.bz2
$ ./configure --prefix=<STACKLESS_DIR> && make -j3 all && make install
```

More details in its documentation.

Then, we recommend using a virtual environment for deploying Kansha. To install *virtualenv* within your fresh Stackless Python, you can execute the following commands:

```
$ wget https://bootstrap.pypa.io/ez_setup.py -O - | <STACKLESS_DIR>/bin/python
$ <STACKLESS_DIR>/bin/easy_install virtualenv
```

To create a virtual environment:

```
$ <STACKLESS_DIR>/bin/virtualenv <VENV_DIR>
```

To activate that new environment:

```
$ source <VENV_DIR>/bin/activate
```

More details at https://virtualenv.pypa.io/en/latest/

Finally, when your virtual environment is active in your shell, type:

```
$ easy_install kansha
```

easy_install caveat: **`easy_install`** ignores completely semantic versioning and may install the lastest development release instead of the latest stable. In that case, you'd better specify the version you want explicitly, for example:

```
$ easy_install kansha==2.0.4
```

Test run

To get quickly up and running, let's use the built-in web server, database and search engine with the default configuration.

1. First, initialize the database (first run only):

```
$ nagare-admin create-db kansha
$ kansha-admin alembic-stamp head
$ kansha-admin create-demo  # optional, create demo users and contents
```

2. Build the search indexes (can be safely repeated anytime):

```
$ kansha-admin create-index
```

3. Launch:

```
$ nagare-admin serve kansha
```

Now kansha is listening. Just point your browser to http://localhost:8080 and enjoy!

For production sites, we recommend you use an external web server, see *Production setup*.

Upgrading

Upgrading Kansha without loosing data is very easy (using the default configuration file):

```
$ easy_install --upgrade kansha
$ kansha-admin alembic-upgrade head
$ kansha-admin create-index
```

And then restart.

If you crafted your own configuration file, see *Upgrading a production site* for more details.

Configuration Guide

In the following, <VENV_DIR> refers to the python virtual environment where you installed Kansha.

`nagare-admin` reference

Read http://www.nagare.org/trac/wiki/NagareAdmin.

Configuration file

Beware! The configuration file format changed since Kansha 2.0.0. If you are upgrading from a 1.0.X version, you have to convert manually your configuration file to the new format below.

Kansha features can be activated and customized with a configuration file like this:

```
[application]
path = app kansha
name = kansha
debug = off
```

```
redirect_after_post = on
as_root = on
title = <<APP_TITLE>> # should be short!
banner = <<LONG_TITLE>> # or motto/slogan or empty
theme = <<CUSTOM_THEME_NAME>>
favicon = # path (optional)
activity_monitor = <<MONITOR_EMAIL>> # optional
crypto_key = <<PASSPHRASE>> # MANDATORY!!!!
disclaimer = # message to display on login screens, below the forms (optional)

[database]
debug = off
activated = on
uri = postgres://<<DBUSER>>:<<DBPASS>>@<<DBHOST>>:<<DBPORT>>/<<DBNAME>> # adapt to␣
↪your own DBMS
metadata = elixir:metadata
populate = kansha.populate:populate
# especially useful for mysql
#pool_recycle = 3600

[search]
engine = sqlite
collection = kansha
index_folder = <<DATA_DIR>>

[authentication]
# built in authentication system
[[dblogin]]
activated = <<AUTH_DB>>
# moderator email if needed
moderator = <<MOD_EMAIL>> # or empty
# automatically create an identicon for new users (on|off)
identicons = on
# default values to fill in the login form (useful for a demo board)
default_username = <<DEFAULT_USERNAME>>
default_password = <<DEFAULT_PASSWORD>>

# authenticate with LDAP
[[ldaplogin]]
activated = <<AUTH_LDAP>>
host = <<AUTH_LDAP_SERVER>>
users_base_dn = <<AUTH_LDAP_USERS_BASE_DN>>
schema = <<AUTH_LDAP_CLASS>>

# authenticate with third party apps
[[oauthlogin]]
activated = <<AUTH_OAUTH>>

# as many oauth providers as you wish
[[[<<PROVIDER>>]]]
activated = <<AUTH_OAUTH>>
key = <<AUTH_OAUTH_KEY>>
secret = <<AUTH_OAUTH_SECRET>>

[locale]
major = fr
minor = FR
```

```
[services]
[[mail_sender]]
activated = on
host = <<MAIL_HOST>>
port = <<MAIL_PORT>>
default_sender = <<MAIL_SENDER>>

[[assets_manager]]
basedir = <<DATA_DIR>>/assets
baseurl = /kansha/services
max_size = 20480

[logging]

[[logger]]
level=INFO

[[handler]]
class=logging.handlers.RotatingFileHandler
args="('<<DATA_DIR>>/logs/<<LOG_FILE>>', 'a', 10485760, 8, 'UTF-8')"
```

Just replace the <<PLACEHOLDERS>> with your actual values.

For your convenience, you can generate a configuration template into your current directory:

```
$ <VENV_DIR>/bin/kansha-admin save-config
```

The template is `kansha.cfg`. Edit it as you need. Ensure the folders you set for logs, assets… do exist.

To manage and run Kansha with your own custom configuration:

```
$ <VENV_DIR>/bin/nagare-admin create-db /path/to/your/kansha.cfg
$ <VENV_DIR>/bin/kansha-admin alembic-stamp head /path/to/your/kansha.cfg
$ <VENV_DIR>/bin/kansha-admin create-index /path/to/your/kansha.cfg
$ <VENV_DIR>/bin/nagare-admin serve /path/to/your/kansha.cfg
```

The different sections are detailled below.

Application

Here you configure the base application.

path Reference to the root component factory of the application (don't edit!).

name URL prefix of the application (`/name/...`).

as_root If `on`, the application is also available without URL prefix, directly as root URL.

debug If `on`, display the web debug page when an exception occurs. The `nagare[debug]` extra must be installed. Never activate on a production site!

redirect_after_post If `on`, every POST is followed by a GET thanks to a redirect. That way, visitors can safely use the *back* button on their browsers.

title Short name for your instance, displayed in various places of the interface. It is the identity of your site. Keep it short (less than 10 chars).

banner Longer title for your site, kind of motto or slogan. It is displayed below the logo on the login page.

theme Name of the theme you want to use, a default one is bundled with Kansha and is named “kansha_flat”.

favicon Path to a favicon file that will be applied to your site.

activity_monitor Email address or nothing. If an email address is provided, activity reports will be sent to it regularly. See *Periodic tasks*.

crypto_key **Required**: this key is used to encrypt cookies. You must change it to secure your site. Put in an hundred random chars (ask a typing monkey).

disclaimer This message is displayed below the login form. You can leave it empty of course.

Database

Kansha data are stored in an SQL database. Thanks to SQLAlchemy, we support all the major databases of the market.

Depending on the DBMS you use, you may need to create the target database first.

Configuration options:

uri SQL Alchemy URI. See http://docs.sqlalchemy.org/en/rel_0_9/core/engines.html#supported-databases

pool_recycle If you are using MySQL as your database backend, you may need to set this option if the mysql configuration sets an automatic disconnection.

Let the other options at their default values.

Note for Postgresql (recommended DBMS for production sites) users:

- install the needed dependencies:

```
$ <VENV_DIR>/bin/easy_install kansha[postgres]
```

Note for MySQL users:

- install the needed dependencies:

```
$ <VENV_DIR>/bin/easy_install kansha[mysql]
```

Note for SQLite users: SQLite is not recommmended for production environments as it does not support schema migrations. If you use SQLite, you won't be able to migrate your data when you install a new version of Kansha.

Search

You can choose one out of two search backends for the moment: SQLite or ElasticSearch. They both work independently from the database you chose to store your data in.

The SQLite backend is quite fast and capable but is only able to do prefix searches. More demanding sites may require ElasticSearch, or you may already have a running cluster on your network.

SQLite backend

This backend is based upon SQLite FTS tables. You need sqlite 3.8.0 or newer. Yet, the search engine can still work with limited functionality down to sqlite 3.7.7. As far as Kansha is concerned, it should not make any difference, since it doesn't use the missing features (for the moment).

Configuration options:

engine sqlite

index The base name of the index file (will be created).

index_folder Where to put the index file (must exist).

ElasticSearch backend

Requires ElasticSearch v2.3.0 or above.

You need to install the python driver first:

```
$ <VENV_DIR>/bin/easy_install kansha[elastic]
```

Configuration options:

engine elastic

index the name of the index on the ElasticSearch cluster (will be created).

host Optional

port Optional

Authentication

You can use up to four different systems, as modules, to authenticate your users in Kansha. You can activate as many modules as you want (at least one).

Module `dbauth`

Database authentication. Users must register first via the web interface.

Configuration options:

activated Whether to activate this module.

identicons Whether a unique avatar should be created for each new user instead of the default anonymous one (`on` / `off`). Default is `on`.

moderator If present, must be an email address. This activates moderation and all registration requests are fowarded to the moderator for approval. Otherwise, registration is free for humans. A CAPTCHA prevents robots from submitting.

Module `ldapauth`

Use this module to authenticate your users against an LDAP or Active Directory database.

You will need to install some additional packages:

```
$ <VENV_DIR>/bin/easy_install kansha[ldap]
```

Configuration options:

activated Activate only if you have some LDAP Directory.

host name or address of the LDAP server.

port (optional) port to connect to.

users_base_dn The base DN your users are under.

schema The driver to use depending on your schema:

- `kansha.authentication.ldap.ldap_auth:NngLDAPAuth` for InetOrgPerson
- `kansha.authentication.ldap.ldap_auth:ADLDAPAuth` for Active Directory

Note: the `kansha.authentication.ldap.ldap_auth:NngLDAPAuth` driver expects the fields "displayName" and "mail" to be set.

Module `oauth`

This governs the OAuth based authentication system. You need to activate it if you wish to let your users connect with their accounts on third party sites or applications.

For that, you configure a provider as a subsection of `oauth`.

The name of the subsection is the provider name (list below) in lowercase. Each subsection has the following configuration parameters:

activated `on` or `off`.

key Write here the API key of the service you intend to use (you have to register with the service first to get one)

secret Write here the secret that authenticates your site by the service you intend to use (you have to register with the service first to get one)

The availble providers are:

- Google,
- Twitter,
- Facebook,
- Github.

Example:

```
[[oauthlogin]]
activated = on

[[[google]]]
activated = on
key = xxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com
secret = XXXXXXXXXXXXXXXXXXXXXXXX

[[[facebook]]]
activated = on
key = 0000000000000000000
secret = XXXXXXXXXXXXXXXXXXXXXXXX
```

Send Mail

All notifications are sent by mail, so you'd better configure an outgoing SMTP server.

host SMTP server to use.

port The port the server listens on.

default_sender The sender address that will appear on all the messages sent by your site.

Asset Manager

You can attach files and images to cards, so you need to set where they will be stored on disk.

basedir The folder where to store uploaded files.

max_size The maximum allowed size of uploaded files, in kilobytes.

Locale

major Default language for your site, two-letter ISO language code.

minor Default region for your site, two-letter ISO country code.

Logging

This is the configuration for the standard python logger. See https://docs.python.org/2/library/logging.config.html#configuration-file-format for a complete explanation.

At a minimum, configure the path to the log file and the logging level.

Production setup

The built-in server, database and search engine are very convenient for testing, but they are not recommended for production sites (*Well, in fact, the default search engine is quite capable; do some benchmarks to decide*).

Fortunately, you can run Kansha with:

- any database supported by SQLAlchemy (complete list at http://docs.sqlalchemy.org/en/rel_0_9/dialects/index.html);
- behind any webserver which supports Fast CGI (FCGI);
- different authentication backends;
- with ElasticSearch as search engine.

For instructions on how to configure Kansha and for detailled explanations of each option, please read the *Configuration Guide*.

In this section, we concentrate on how to deploy Kansha as a multiprocess application backend behind a web server.

In the following, <VENV_DIR> is the path to the python virtual environment you've installed Kansha in.

Installation

As in the quickstart guide, follow the installation steps from *Python package*.

You also need to install:

- the database you want to use;
- memcached;
- your favorite web server with FCGI support;
- and if you choose to, ElasticSearch.

You can run memcached and ElasticSearch with their default configurations.

Then configure Kansha (*Configuration Guide*).

In any case, you always need to:

```
$ <VENV_DIR>/bin/nagare-admin create-db </path/to/your/kansha.cfg>
$ <VENV_DIR>/bin/kansha-admin alembic-stamp head </path/to/your/kansha.cfg>
$ <VENV_DIR>/bin/kansha-admin create-index </path/to/your/kansha.cfg>
```

When you **first** deploy.

Deployment behind a web server

To deploy Kansha behind a web server, we use a Fast CGI (FCGI) adapter and a memcached server to allow communication between processes.

The steps are:

1. install, configure and start memcached;
2. configure kansha to start FCGI processes;
3. install, configure and start your favorite web server with FCGI connectivity to Kansha processes;
4. start Kansha.

Configure Kansha for FCGI

Append these directives to your configuration file:

```
[publisher]
type = fastcgi
host = <<FASTCGI_HOST_KANSHA>>
port = <<FASTCGI_PORT_KANSHA>>
debug = off
minSpare = <<FASTCGI_MINSPARE>>
maxSpare = <<FASTCGI_MAXSPARE>>
maxChildren = <<FASTCGI_MAXCHILDREN>>

[reloader]
activated = off
interval = 1

[sessions]
type = memcache
host = <<MEMCACHE_HOST>>
port = <<MEMCACHE_PORT>>
min_compress_len = 100000
reset = true
```

Or, if you run the web server on the same machine as Kansha, you can use unix sockets:

```
[publisher]
type = fastcgi
socket = <<SOCKET_PATH>>
umask = <<SOCKET_MASK>>
debug = off
```

```
minSpare = <<FASTCGI_MINSPARE>>
maxSpare = <<FASTCGI_MAXSPARE>>
maxChildren = <<FASTCGI_MAXCHILDREN>>

[reloader]
activated = off
interval = 1

[sessions]
type = memcache
host = <<MEMCACHE_HOST>>
port = <<MEMCACHE_PORT>>
min_compress_len = 100000
reset = true
```

Set the <<PLACEHOLDERS>> as appropriate.

A sample configuration you can start with (assuming memcached is running with defaults and you use sockets):

```
[publisher]
type = fastcgi
socket = /path/to/the/socket/you/want
debug = off
minSpare = 2
maxSpare = 4
maxChildren = 10

[reloader]
activated = off
interval = 1

[sessions]
type = memcache
host = localhost
port = 11211
min_compress_len = 100000
reset = true
```

All options are documented in this section of the Nagare documentation.

Optimize how static contents are served

Your web server is better at serving static content than Kansha, so you'd better configure it to serve the static resources itself and pass the other requests to the Kansha backend.

If you are using Apache, Nginx or Lighttpd, you'll find the detailled instructions in the deployment section of the Nagare manual.

Start Kansha

Once you have configured the FCGI publisher, you can start Kansha as usual:

```
$ <VENV_DIR>/bin/nagare-admin serve </path/to/your/kansha.cfg>
```

That command starts the backend FCGI processes.

Using a supervisor

Optional, but recommended, see Handling the FastCGI processes in the Nagare manual.

Periodic tasks

Kansha emits notifications users can subscribe to. In order for those notifications to be sent, you have to call a batch task regularly:

```
$ <VENV_DIR>/bin/nagare-admin batch <<PATHTOCONFFILE>> kansha/batch/send_
↪notifications.py <<TIMESPAN>> <<APPURL>>
```

Where the <<PLACEHOLDERS>> are correctly replaced by, respectively:

- the path to the configuration file of Kansha;
- the timespan covered by the reports (in hours);
- the url of the application.

You can locate the `send_notifications.py` file in your python virtual environment (`<VENV_DIR>/lib/python2.7/site-packages/kansha/batch/`).

Place this command in a crontab and check that the timespan matches the time interval between each run.

Of course, that assumes you have previously configured an outgoing SMTP server in the *Send Mail* section of the configuration file.

Upgrading a production site

We mean *upgrading Kansha* while keeping your data.

Beware! The configuration file format changed since Kansha 2.0.0. If you are upgrading from a 1.0.X version, you have to convert your configuration file to the new format first, see *Configuration Guide*.

Just type:

```
$ <VENV_DIR>/bin/easy_install --upgrade kansha
$ <VENV_DIR>/bin/kansha-admin alembic-upgrade head </path/to/your/kansha.cfg>
$ <VENV_DIR>/bin/kansha-admin create-index </path/to/your/kansha.cfg>
```

Or, if you want a specific version instead of the latest release (replace X, Y and Z with the actual numbers):

```
$ <VENV_DIR>/bin/easy_install kansha==X.Y.Z
```

Update the rewrite rules for static resources.

Now restart Kansha.

Development setup

How to setup your environment and install Kansha from GitHub for development.

Beside the requirements, you need to have **`git`** installed. You should already be familiar with **`git`** and GitHub. If that's not the case, check https://help.github.com/articles/good-resources-for-learning-git-and-github/.

The following instructions apply to UNIX-like systems, like Linux or MacOS X.

Install Stackless Python and Virtualenv

Nagare, the framework used by Kansha, needs Stackless Python (version 2.7.X) to run.

Unfortunatly, none of the major Linux distributions offer packages for Stackless, so you have to build it from sources.

In order to install it via the sources, first ensure you have the prerequisite system dependencies, then complete the following commands:

```
$ mkdir <STACKLESS_DIR>
$ wget http://www.stackless.com/binaries/stackless-278-export.tar.bz2
$ tar xf stackless-278-export.tar.bz2
$ ./configure --prefix=<STACKLESS_DIR> && make -j3 all && make install
```

More details in its documentation.

Then, we recommend using a virtual environment for deploying Kansha. To install *virtualenv* within your fresh Stackless Python, you can execute the following commands:

```
$ wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - |
↪<STACKLESS_DIR>/bin/python
$ <STACKLESS_DIR>/bin/easy_install virtualenv
```

Install Kansha for development

First, create a stackless virtual environment so your development environment remains isolated:

```
$ <STACKLESS_DIR>/bin/virtualenv <VENV_DIR>
```

<VENV_DIR> is whereever you want your virtual environment be created. Note that we won't be working in that directory directly, so it can be a hidden one.

Fork the Kansha project on GitHub.

Clone your project locally. Now you have a `kansha` folder (<KANSHA_DIR> in the following). That's where the actual development will take place.

Activate the virtual environment you created above:

```
$ source <VENV_DIR>/bin/activate
```

Then install Kansha in development mode:

```
$ cd <KANSHA_DIR>
$ python setup.py develop
```

The last command installs kansha in the virtual environment *in place*. That is, any modification done to the files in `kansha/kansha` will be available immediatly, without re-installing.

Finally, install all the optional dependencies of Kansha:

```
$ pip install kansha[test]
$ pip install kansha[htmldocs]
$ pip install kansha[ldap]
$ pip install kansha[postgres]
$ pip install kansha[mysql]
$ pip install kansha[elastic]
```

Configure Kansha

In the `conf` directory, copy `kansha.cfg` to `kansha.local.cfg` and edit the latter to fit your system.

Test run

For developing, let's use the built-in web server, database and search engine with the custom configuration.

Place yourself at the root of the project (<KANSHA_DIR>). The virtual environment is still activated in your shell (look at the prompt); if not, activate it.

0. First, initialize the database (first run only):

   ```
   $ nagare-admin create-db conf/kansha.local.cfg
   $ kansha-admin alembic-stamp head conf/kansha.local.cfg
   ```

1. If you later need to migrate your database after a schema change in the model:

   ```
   $ kansha-admin alembic-upgrade head conf/kansha.local.cfg
   ```

2. Build the search indexes (can be safely repeated anytime, only needed at firt run actually):

   ```
   $ kansha-admin create-index conf/kansha.local.cfg
   ```

3. Launch:

   ```
   $ nagare-admin serve conf/kansha.local.cfg --reload
   ```

Now kansha is listening. Just point your browser to http://localhost:8080 and check.

The `--reload` switch is handy for development, as the server then reloads kansha whenever a python file is modified.

Later, each time you'll want to run Kansha in development mode, remember these steps:

```
$ cd <KANSHA_DIR>
$ source <VENV_DIR>/bin/activate
$ nagare-admin serve conf/kansha.local.cfg --reload
```

Development cycle

Now that your environment is ready and kansha is running is development mode, let's hack!

Generic workflow:

1. Develop;
2. translate (if appliable);
3. document;
4. write unit tests for internal funtionality and API (*for the latter, write the tests first, then develop*);
5. test;
6. repeat from 1. until your tests (automatic and/or manual) pass;
7. commit with appropriate message;
8. go to 1 until your work is done;
9. push;

10. submit a pull request on github.

Specific recommendations and workflows are described in theses sections:

- *Contribute to the documention*
- *Translate*
- *Fix bugs and code new features*

It's highly recommended that you subscribe to the mailing list: http://groups.google.com/group/kansha-users

First connection

Kansha works with Firefox, Chrome, Internet Explorer 9 and above, Safari 7 and above.

Login screen

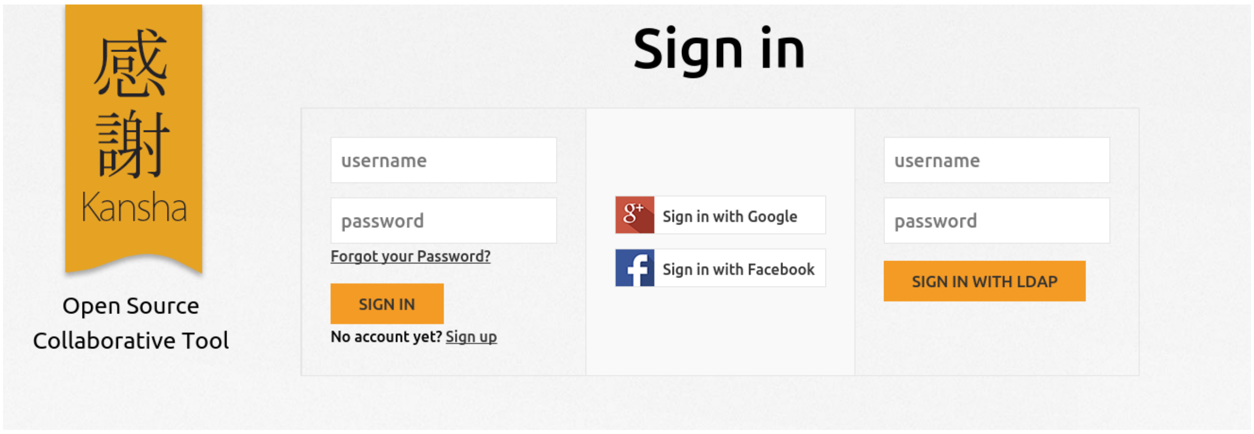


Fig. 1.1: Login screen with all authentication methods enabled

Kansha supports three different authentication schemes:

- *Database authentication*
- *LDAP authentication*
- *OAuth authentication* (Google, Facebook, Twitter. . .)

The administrator of Kansha may choose to enable one or several of those authentication methods. So, your actual login screen may differ from the one pictured above.

The installer created several demonstration accounts for you. They are: user1, user2, user3. They all have the same password: *password*.

Database authentication

To be able to login with that form, you need to register an account first, by clicking on the *create one* link.

When you submit the registration form, an email is sent to you to verify your email address, or to a moderator who will verify your identity. The actual registration process depends on the site policy.

If Kansha sends you an email, just follow the instructions. Basically, you have to click on a link to confirm your email address. Then your can login with your credentials.

If the registration is moderated, you may be contacted by the moderator. Eventually, the moderator will inform you when your account is activated or denied.

LDAP authentication

If LDAP authentication is enabled, you can login to Kansha with the credentials you already use to sign on the other applications of your company.

OAuth authentication

OAuth authentication allows users of third party applications to log in Kansha.

The administrator of Kansha may grant access to users of:

- Google,
- Twitter,
- Facebook,
- Github.

Home

When you first login, you arrive straight on your *Home page*.

On the home screen you have access to:

- the list of the boards you can participate in (see *Manage access rights*);
- your profile, which you can edit.

On the board tab, the boards you have access to are organized as follows:

- The 5 last modified boards, for direct access to hot boards;
- your boards, i.e. the boards you are manager of;
- guest boards, i.e. the boards you are just member of;
- shared boards, i.e. all the boards that are public and shared on this Kansha instance.

If you are logging on a freshly installed Kansha instance, the board list is empty.

On your profile, you can change the language of the interface. If your favorite language is missing, consider *contributing*.

You are encouraged to upload a picture of your face on your profile.

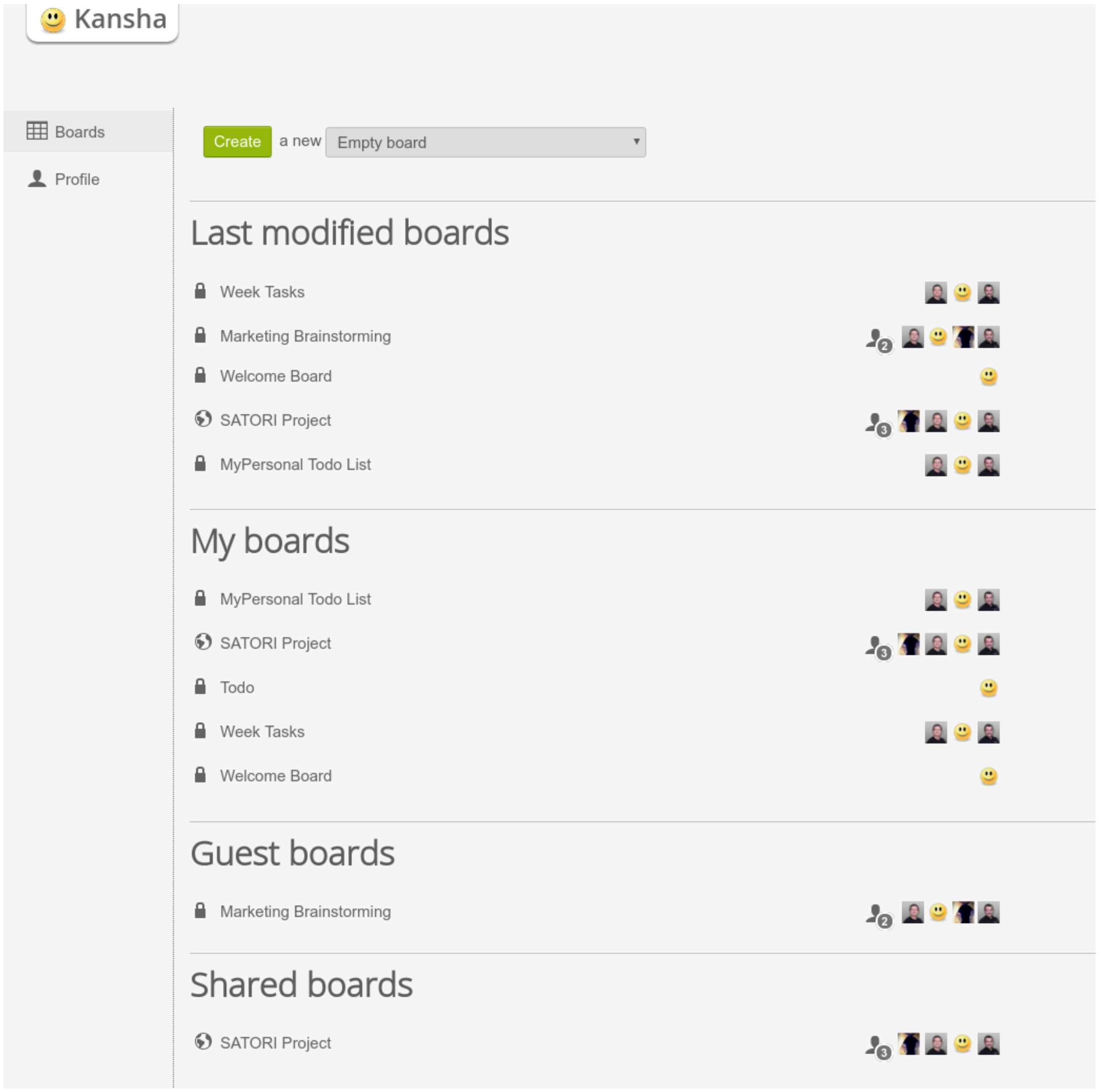
Your first board

Let’s create your first board!

In the template drop-down menu, chose “Todo” then click on the “Create” button.

Your newly created board automatically opens.

Now adds a few card by using the menus at the bottom of the lists. Columns are called lists in Kansha.



Last modified boards

- Week Tasks
- Marketing Brainstorming
- Welcome Board
- SATORI Project
- MyPersonal Todo List

My boards

- MyPersonal Todo List
- SATORI Project
- Todo
- Week Tasks
- Welcome Board

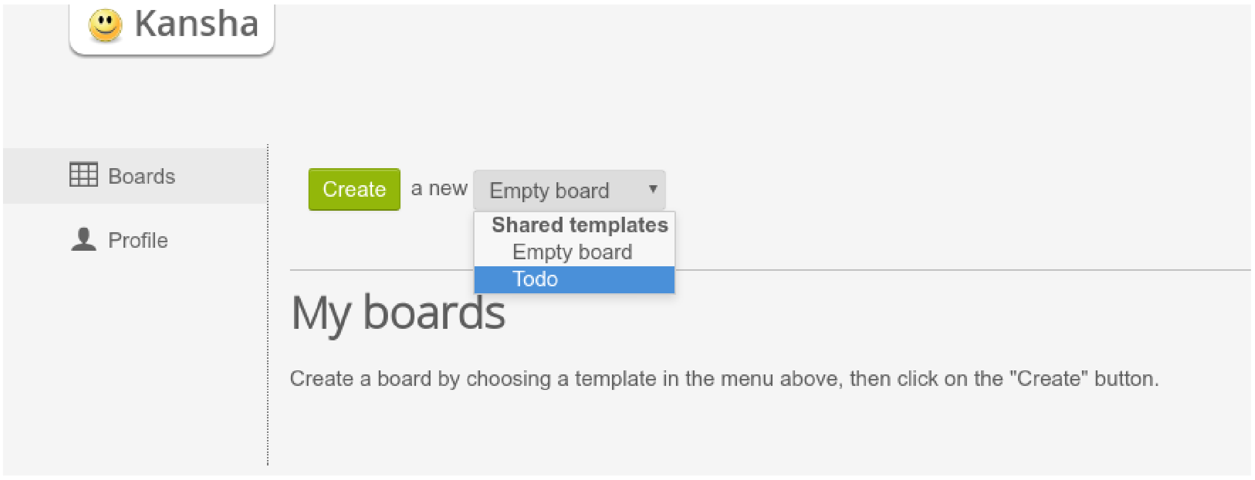
Guest boards

- Marketing Brainstorming

Shared boards

- SATORI Project

Fig. 1.2: The home page of a very active user.



Create a board by choosing a template in the menu above, then click on the "Create" button.

Fig. 1.3: Create your first board.

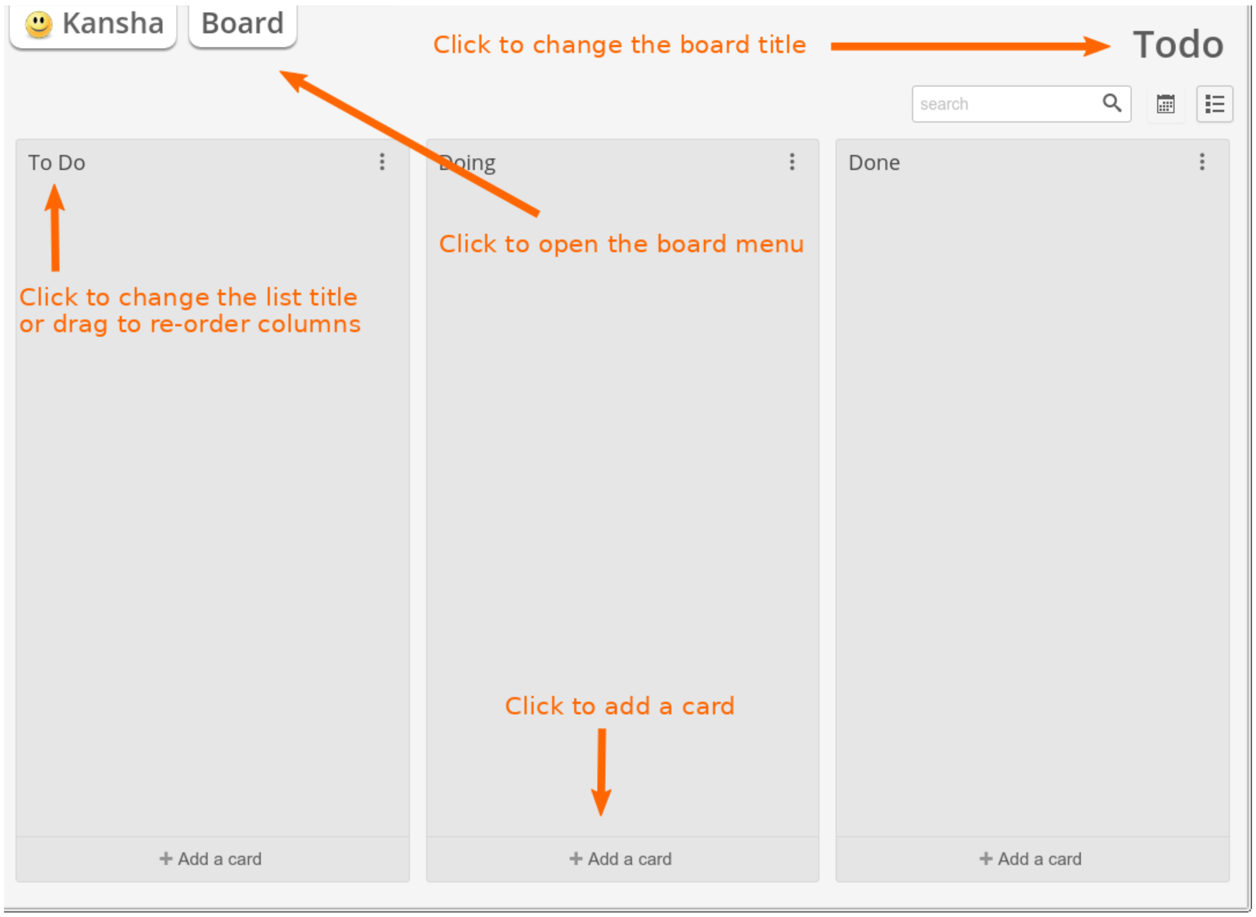
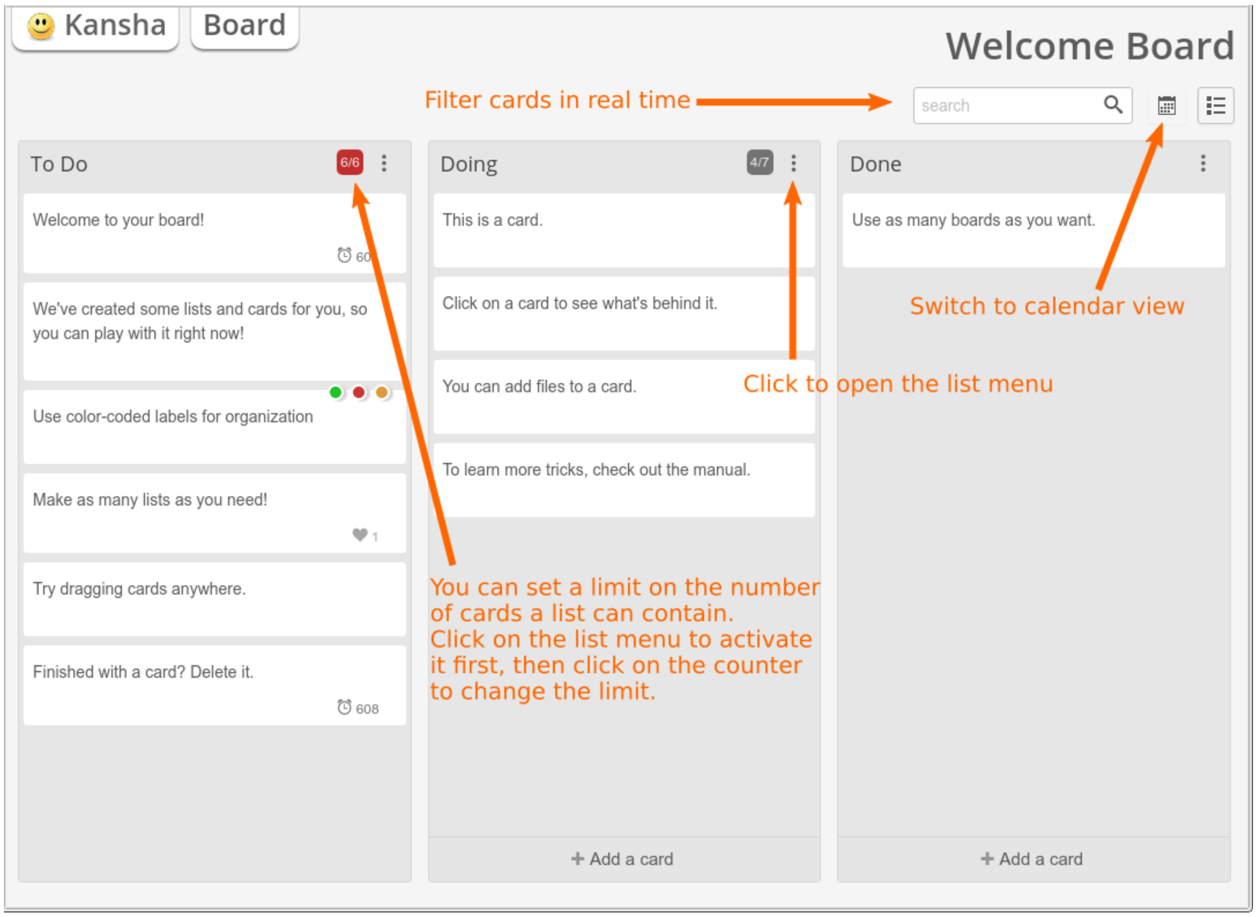

Kansha
Board
Click to change the board title
Todo
search
To Do
Doing
Done
Click to open the board menu
Click to change the list title
or drag to re-order columns
Click to add a card
+ Add a card
+ Add a card
+ Add a card

Fig. 1.4: This is how your first todo board looks like

Now try these:

- add new lists by using the board menu;
- change the board title by clicking on it;
- drag some cards around;
- open a card by clicking on it and discover its features;
- explore the board menu;
- make your own experiments.

Anatomy of a board



Welcome to your board!

We've created some lists and cards for you, so you can play with it right now!

Use color-coded labels for organization

Make as many lists as you need!

Try dragging cards anywhere.

Finished with a card? Delete it.

This is a card.

Click on a card to see what's behind it.

You can add files to a card.

To learn more tricks, check out the manual.

Use as many boards as you want.

You can set a limit on the number of cards a list can contain. Click on the list menu to activate it first, then click on the counter to change the limit.

Fig. 1.5: A board with some cards and list limits.

In Kansha, a board is made of columns, also known as lists, that contain cards. You can add as many columns as you wish to a board.

Columns can be reordered by dragging and dropping them. Cards can be moved accross columns and reordered the same way.

To open a card, just click on it.

Take some time to play with the cards on your board. For now your board is private and you can safely experiment without causing trouble to other users.

On a card you can:

- edit the title;
- add/remove *labels* (tags);
- edit a description;
- comment;
- add and check check-lists;
- add files;
- vote (if activated by the board owner, see *Configure your boards*);
- give it some weight (if activated by the board owner, see *Configure your boards*);
- set a due date;
- assign members to it (if you have invited other users to your board, see *Manage access rights*).

The columns may have a limit on the number of cards they accept. This limit is displayed after the slash in the column counter. To activate the counter, click on the list menu. To set the limit, just click on the counter.

To change titles just click on them. That works for:

- cards;
- columns;
- board.

Now, look at the switches in the upper right corner of the screen. By default, *board mode* is activated. If you click on *calendar mode*, the screen displays a view of the current month where you can see the cards that expire that month.

Last, consider the main tabs. The **Kansha** one gives you access to your *home* (next section). The **Board** one contains everything you need to manage the current board.

Board operations available in the **Board** tab (for members only):

Preferences This menu allows you to configure the board and to subscribe to notifications. Board configuration is covered in *Configure your boards*. Notifications will be sent to you by email.

Add list Add a new column.

Edit board description Describe here what the board is for.

Export board Export all cards as lines in an XLS file.

Save as template (requires management role) Save the current board as a template.

Action Log The *Action log* displays the history of the actions that happened on the current board. Open it and see what you have done in this board so far.

Delete board / Leave this board Respectively on boards you own and boards you are simply a member of, those actions just do what you would expect.

Searching

Use the search input to search the cards.

Type your query terms here: the irrelevant cards are filtered out as you type and the matching cards are highlighted.

The search engine looks at the title, description, comments and labels of cards.

Board management

Manage access rights

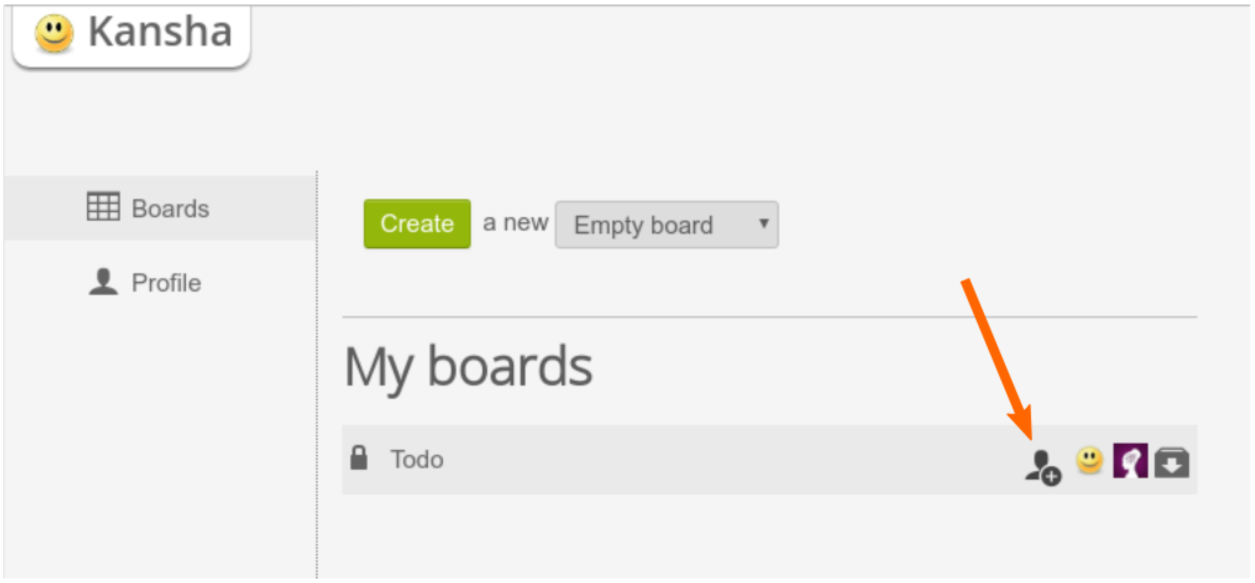


Fig. 1.6: The list of the boards the logged used has access to.

In the board list on your home, you can see all boards you have access to:

- the boards you own;
- the boards you've been invited to;
- the shared boards open the everybody (read only).

When you hover a board, action icons appear.

On the boards you own, you can:

Archive the board: The board will not be accessible anymore, but it is not destroyed. You can restore it anytime.

Add members: The *add member* icon (pointed by the orange arrow on the screenshot above) allows you to invite members to your board. Invited people can already be users of the application, in which case you can find them by username or email, or you can invite new people by entering their emails.

Revoke members: Click on a member, then on "Remove".

On the boards you are simply a member of, you can opt-out anytime.

Configure your boards

Go to the board of yours you want to configure. In the **Board** main tab, chose *Preferences*. The different entries are described below:

Profile

Visibility Private boards are only for invited members. Public boards are open to everybody who can log in Kansha, but a needs to know the exact URL (or a link) to the public board to access it. Shared boards are public boards that are visible on the home page of every user.

Comments You choose who can post comments on cards.

Votes Same for voting.

Archive Deleted cards are still recoverable for they are actually archived. Show the archive column to see them. If you purge the archive (action in the archive column menu), archived cards are definitetly destroyed.

Notifications This preference is open to all members so they can subscribe to activity digests for this board.

Card labels

Card labels are kind of tags users can set on cards. They are predefined here. Give them a name and a color. You can't add nor remove labels from this list.

Card weights

Cards can be weighted by order of importance. If you enable that ranking, you can define a predefined sequence of weights to choose from or let your members freely weight the cards.

Background

Here you can set a wallpaper to your board and change the board title color to match. In particular, if the background image is dark, set a lighter title color for better readability.

How to contribute to Kansha

You don't have to be a developer to help Kansha get better. You can contribute in many ways depending on your skills and preferences.

Spread the word!

If you enjoy Kansha, share it! It's open source and free, even for companies. Don't hesitate to promote it inside your organization or working team. It's a profitable tool for collaborators on a project.

Give some feedback

Bug reports

Help us improve Kansha by reporting the bugs or unexpected behaviors you may encounter, by following the steps below.

Can you reproduce the issue? Try to reproduce your bug using a recent version of the software, to see whether it has already been fixed. The easiest way to test the latest stable version of Kansha is on the demo.

Has someone else already reported the issue? Use the search box on GitHub Issues to see if your bug has already been reported. If it is the case, you can contribute more information by commenting the issue.

Reporting a new bug. If you reproduce the bug and nobody has reported it already, go to GitHub Issues and there:

1. Click on the green button "New issue".

2. You will be asked to log in (or register) if you have not already done so.
3. As title, enter a short one-sentence summary that explains the problem, biginning with “[Bug]”.
4. As comment, tell us:
   - The version of the Kansha software on which you raised the bug (visible in the footer of your home page, or of the login page).
   - Steps to reproduce: Easy-to-follow steps that will trigger the described problem. Include any special setup steps.
   - Actual results: What the application did after performing the above steps.
   - Expected results: What the application should have done, if there was no bug.
   - The web browsers or computer systems you’ve seen the bug on.
   - Whether the problem appears every time, only occasionally, only on certain pages, or only in specific circumstances.
5. You may also attach a log file or screenshot (but make sure that no confidential data is included or shown).
6. Submit the issue.

Recommendations:

- Be precise.
- Be clear: explain how to reproduce the problem, step by step, so others can reproduce the bug.
- Include only one problem per report.

Feature requests

Your are missing some features in Kansha? Fill a feature request!

Make sure the feature does not already exist! Have a look at the latest documentation on http://kansha.readthedocs.org/latest/, or play with the latest stable version of Kansha on the demo.

Has someone else already requested the feature? Use the search box on GitHub Issues to see if your feature has already been requested. If it is the case, you can contribute to the user story by commenting the issue.

Requesting a new feature. If you are sure the feature does not exist yet, and that nobody has asked for it, go to GitHub Issues and there:

1. Click on the green button “New issue”.
2. You will be asked to log in (or register) if you have not already done so.
3. As title, enter a short one-sentence summary that explains the expected feature, beginning with “[Feature request]”.
4. As comment, tell us:
   - A description of what you would like to achieve, and why. A user story is an effective way of conveying this.
5. Submit the issue.

Contribute to the documention

We try hard to keep the documentation accurate and up-to-date, and volunteers are welcome.

If you find typos, inaccuracies, mistakes or misleading information; if you think some features of Kansha deserve more detailled explanations; if you missed some tricks or tidbitts you learned the hard way later; please contribute directly to the manual.

Direct contribution

First, you'll need to prepare your *Development setup*. Keep the virtual environment activated, or activate it.

Like many other Python projects, we use reStructuredText to write the documentation, and Sphinx and Readthedocs to format it into HTML pages.

Besides plain reST, we also make heavy use of Sphinx extended directives.

The source documentation is located in `<KANSHA_DIR>/doc`. To build the html version locally, just type:

```
$ cd <KANSHA_DIR>/doc
$ make html
```

You will then find the HTML files in `<KANSHA_DIR>/doc/_build/html/`.

Documentation workflow:

1. Redact;
2. check your grammar, spelling and syntax;
3. build the HTML;
4. proofread;
5. repeat from 1. until your text is clear, complete and correct;
6. commit with appropriate message;
7. go to 1 until your work is done;
8. push;
9. submit a pull request on github.

To avoid duplicate work or conflicts, you'd better fill an issue first, to announce what you are going to do , on GitHub Issues. For that, proceed as below *Indirect contribution*, except you don't have to redact your contribution inside the issue. Instead, you assign it to you.

Indirect contribution

If the workflow described above is too complicated for you, there is an alternative, yet much less effective: submit an *enhancement* issue on GitHub Issues and wait for a volunteer to implement it.

1. Click on the green button "New issue".
2. You will be asked to log in (or register) if you have not already done so.
3. As title, enter a short one-sentence summary that explains the proposed prose, beginning with "[Docs]".
4. As comment, you:
    - tell us whether you propose a fix or new paragraphs/sections;

- precise where in the manual you contribution should go;
- **redact** the part of the manual you want to add or fix.

5. Submit the issue.

And, *maybe*, a direct contributor will discuss, pick and implement your request.

Translate

Contributing to Kansha localization is easy!

Just use the online interface provided by Transifex: https://www.transifex.com/net-ng/kansha/

Fix bugs and code new features

You want to actively contribute to the code: welcome aboard!

Pick up (*or fill in*) a bug or a feature request in the GitHub Issues and let's go!

Preparation

First, check the issue is not already assigned to someone. If it is free, declare your intentions by posting a comment on the issue. That will start a discussion with the other developers, who may give you valuable advice. If you are new to Nagare development, you should choose to fix some bugs first, before implementing new features. If everything goes well, you'll be assigned to that issue.

If not already done, prepare your *Development setup*. Keep the virtual environment activated, or activate it.

Now you can code. Kansha is developed upon the Nagare Framework. If you are not already familiar with Nagare development, these are useful resources:

- The Nagare tutorial.
- The Nagare documentation.
- The Nagare API.

Guidelines

Backend

Nagare applications are based on components, so always think *component*.

Components should be reusable. Components should not necessarily match business/domain objects. Components correspond to functional parts of the user interface and business logic. They may use several domain objects. Domain objects must not be aware of components.

As a consequence, the main view of a component should generate one and only one DOM tree (i.e. only one root, usually a `div`).

Kansha uses semantic HTML5 for the UI. Avoid presentation specific markup and inline styles.

Python code should comply with PEP8. That requirement may be relaxed when it is difficult or impossible to follow, e.g. in views with many context managers (`with ... :`).

Since we use service injection, **pass optional parameters to functions/classes as explicit keyword arguments instead of positional arguments**, unless they are services.

If you add new services, provide a mockup version for tests.

Use docstrings and comments to make it easier for other developers to understand your code.

All UI messages and labels should be in UTF8 and marked for localization.

Write unit tests for internal functions, classes and API (we use **`nose`**).

If you need to update the data schema, use the integrated alembic commands: **`kansha-admin alembic-revision [application]`**, **`kansha-admin alembic-upgrade [application]`**...

Frontend

The frontend must work on Internet Explorer >=9, Firefox, Chrome and Safari. Test before submitting a Pull Request.

CSS Please don't overqualify your selectors! Overqualified selectors are slow, add clutter, are difficult to read and go against component reusability. That rule was not followed in the past and it was a mistake.

To minimize git merge conflicts and to improve readability, write one property per line, with 4 space indentation (like python code). Selectors are not indented. Always put a space after the colon (:).

Tools like CSS lint can help.

Javascript Most of these recommendations apply here as well. Use what is already in place instead of adding new layers of code. Don't do in Javascript what can be done with asynchronous Nagare views. Always favor the Nagare way.

Tools like jshint or jslint can help.

General

If you add a new feature, or if you modify the behavior or UI of an existing feature, please *update* the documentation.

Now that the translations are managed by Transifex, the `.PO` files must no be edited manually anymore.

Workflow

1. Develop;
2. check your style;
3. update the manual (if new feature, modified UI or behavior);
4. write unit tests for internal funtionality and API (*for the latter, write the tests first, then develop*);
5. test (IE9/Firefox/Chrome/Safari);
6. repeat from 1. until your tests (automatic and/or manual) pass;
7. commit with appropriate message;
8. go to 1 until your work is done;
9. push;
10. submit a pull request on github.

Review Pull Requests

An other appreciated contribution is when you review others' pull requests. That' s also an excellent way to socialize and be part of the community.

Mailing list

It's highly recommended that you subscribe to the mailing list if you plan to contribute to Kansha: http://groups.google.com/group/kansha-users

Frequently Asked Questions

How can I change the design of a Kansha app?

Kansha supports custom themes, a default theme called *kansha_flat* is bundled with the app.

A theme is made of 4 CSS files grouped in a folder named as your theme:

- `kansha.css` is included in every page, it should contain common style rules used all around the app
- `board.css` is included in the board view, it defines the styles of the boards, colums and cards
- `home.css` is only included in the homepage listing your boards and in user profile/user cards
- `login.css` is included in the login page only

For development purposes you can store it in the folder */static/css/themes/* or configure a web server to serve these files according to deployment section of the Nagare manual.

To activate it, you have to edit the *Application* section of your configuration file, changing the `theme` parameter to fit your theme name.

Once done, restart your app and enjoy.

See `static/css/themes/kansha_flat` folder for a working example.

CHAPTER 2

Index