Kanboard Documentation

The Kanboard Authors

Jan 15, 2021

Table of Contents

Kanboard is a free and open source Kanban project management software.

- Official website: https://kanboard.org
- Bug Tracker: https://github.com/kanboard/kanboard/issues
- Forum: https://kanboard.discourse.group/
- RSS Feed: https://github.com/kanboard/kanboard/releases.atom
- Mastodon: https://mastodon.social/@kanboard
- IRC: #kanboard on Freenode

CHAPTER 1

User's Guide

End user help and documentation.

1.1 Introduction

1.1.1 What is Kanban?

Kanban is a methodology originally developed by Toyota to be more efficient.

There are only two constraints imposed by Kanban:

- Visualize your workflow
- Limit your work in progress

Visualize your workflow

- Your work is displayed on a board so that you have a clear overview of your project
- Each column represents a step in your workflow

Limit your work in progress

- Encourages focus by avoiding multitasking
- Each phase can have work-in-progress limits
- Limits help identify bottlenecks
- Limits help avoid working on too many tasks at the same time

Performance Measurement

Kanban uses lead and cycle times to measure performance:

- **Lead time**: Time between task creation and completion
- **Cycle time**: Time between task start and completion

For example, you may have a lead time of 100 days but only have to work 1 hour to complete the task.

1.1.2 Kanban vs Todo lists

Todo lists:

- Single phase (just a list of items)
- Multitasking possible (not efficient)

Kanban:

- Multiple phases, each column represents a step
- Bring focus and avoid multitasking by setting a work-in-progress limit per column

1.1.3 Kanban vs Scrum

Scrum:

- Sprints are time-boxed, usually 2 or 4 weeks
- Do not allow changes during the iteration
- Estimation is required
- Uses velocity as default metric
- Scrum board is cleared between sprints
- Scrum has pre-defined roles like scrum master, product owners and the team
- A lot of meetings: planning, backlogs grooming, daily stand-up, retrospective

Kanban:

- Continuous flow
- Changes can be made at any time
- Estimation is optional
- Use lead and cycle time to measure performance
- Kanban board is persistent
- Kanban doesn’t impose strict constraints or meetings; the process is more flexible

1.1.4 Usage Examples

You can customize your boards according to your business activities:

Software development

- Backlog
- Ready
- Work-in-progress
- To be validated
- Validated
- Deployed in production

Bug Tracking

- Reported
- Confirmed
- Work-in-progress
- Tested
- Fixed

Sales

- Leads
- Meeting
- Proposal
- Purchase

Lean Business Management

- Ideas
- Development
- Measure
- Analysis
- Done

Recruiting Process

- Job offers
- Candidates
- Phone screens

- Interviews
- Hires

Online Shops

- Orders
- Packaging
- Ready to send
- Shipped

Manufactory

- Customer Orders
- Assembly
- Tests
- Packaging
- Ready to ship
- Shipped

1.2 Users and Groups

1.2.1 User Types

In Kanboard there are two types of users:

Type	Description
Local User	User that stores his password in Kanboard's database
Remote User	User credentials are managed by another system (Example: LDAP server)

Examples of remote users:

- LDAP user
- Users authenticated by a reverse-proxy
- OAuth2 users

1.2.2 User Roles

Application Roles

Each Kanboard user has one of these roles:

Role	Description
Administrator	Access to everything
Manager	Can create team projects but cannot change application settings
User	Can create private projects only

Project Roles

Each individual team project can assign a different role to each user and group:

Role	Description
Project Manager	Can change project settings, access to the Gantt chart and reports
Project Member	Can create tasks and use the board
Project Viewer	Read-only access to the board and tasks

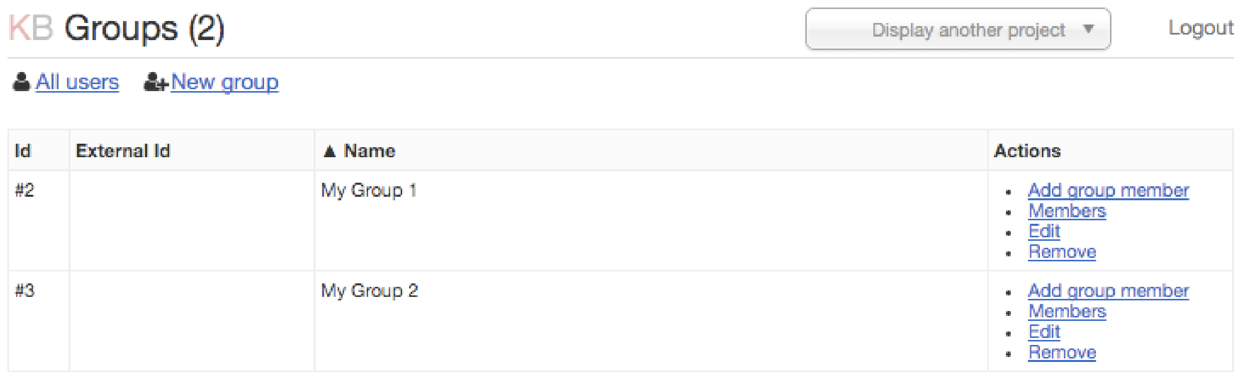
Custom project roles can be created to apply a set of restrictions to the users.

1.2.3 Groups Management

In Kanboard, each user can be a member of one or many groups. A group is like a team or an organization.

Only administrators can create new groups and assign users.

Groups can be managed from **User management > View All Groups**. From there, you can create groups and assign users.



Id	External Id	▲ Name	Actions
#2		My Group 1	Add group member, Members, Edit, Remove
#3		My Group 2	Add group member, Members, Edit, Remove

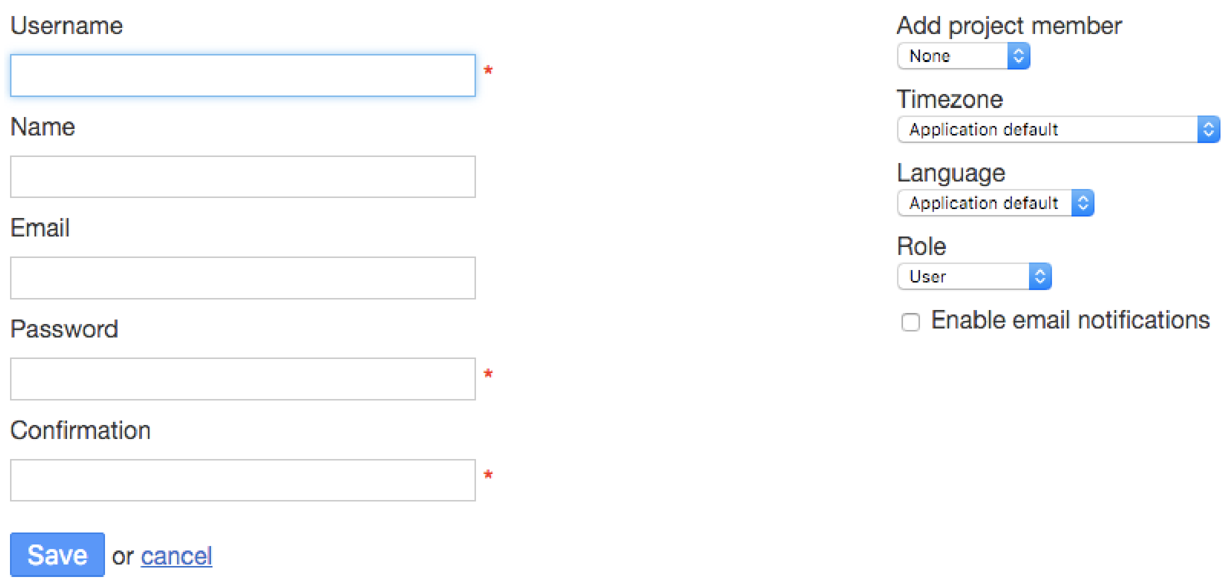
Each project manager can authorize the access to a set of groups from the project permissions page.

The external id is mainly used for external group providers. Kanboard provides a LDAP group provider to sync automatically groups from LDAP servers.

1.2.4 Add a New User

To add a new user, you must be an administrator.

1. From the dropdown menu in the top right corner, go to the menu **Users Management**
2. On the top, you have a link **New local user** or **New remote user**
3. Fill the form and save



When you create a **local user**, you have to specify at least those values:

- **username**: This is the unique identifier of your user (login)
- **password**: The password of your user must have at least 6 characters

For **remote users**, only the username is mandatory.

1.2.5 Edit Users

When you go to the **users** menu, you have the list of users, to modify a user click on the **edit link**.

- If you are a regular user, you can change only your own profile
- You have to be an administrator to be able to edit any users

1.2.6 Remove Users

From the **users** menu, click on the link **remove**. This link is visible only if you are administrators.

If you remove a specific user, **tasks assigned to this person will be unassigned** after the operation.

1.2.7 Two-Factor Authentication

Each user can enable the two-factor authentication. After a successful login, a one-time code (6 characters) is asked to the user to allow access to Kanboard.

This code has to be provided by a compatible software usually installed on your smartphone or a different device.

Kanboard use the Time-based One-time Password Algorithm defined in the RFC 6238.

There are many software compatible with the standard TOTP system. For example, you can use these applications:

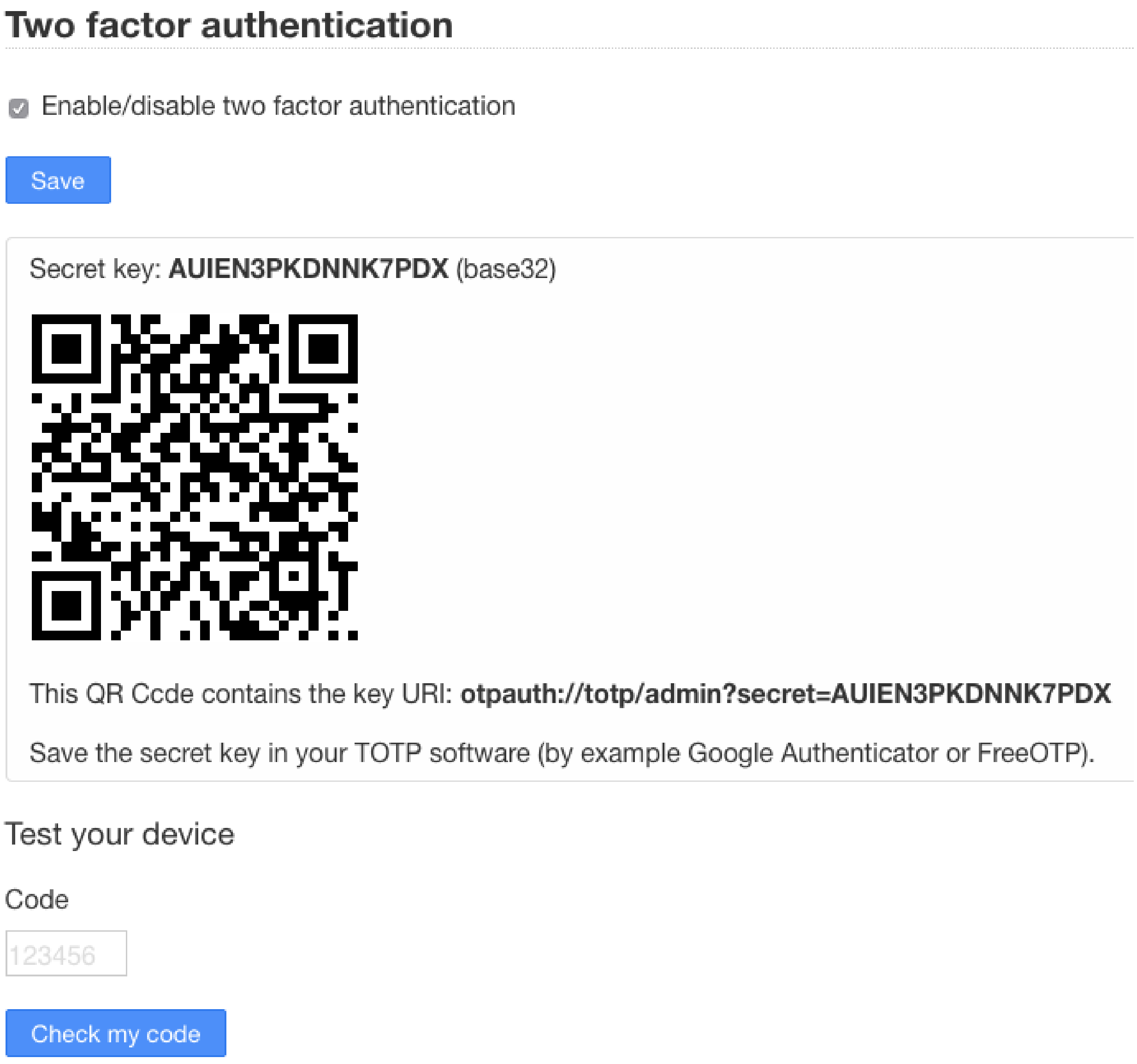
- Google Authenticator (Android, iOS, Blackberry)
- FreeOTP (Android, iOS)

- OATH Toolkit (Command line utility on Unix/Linux)

This system can work offline and you don't necessarily need to have a mobile phone.

Configuration

1. Go to your user profile
2. On the left, click on **Two Factor Authentication** and click on the button
3. A secret key is generated for you



Two factor authentication

☑ Enable/disable two factor authentication

Secret key: **AUIEN3PKDNNK7PDX** (base32)

This QR Ccde contains the key URI: **otpauth://totp/admin?secret=AUIEN3PKDNNK7PDX**

Save the secret key in your TOTP software (by example Google Authenticator or FreeOTP).

Test your device

Code

- You have to save the secret key in your TOTP software. If you use a smartphone, the easiest solution is to scan the QR code with FreeOTP or Google Authenticator.
- Each time you will open a new session, a new code will be asked
- Don't forget to test your device before closing your session

A new secret key is generated each time you enable/disable this feature.

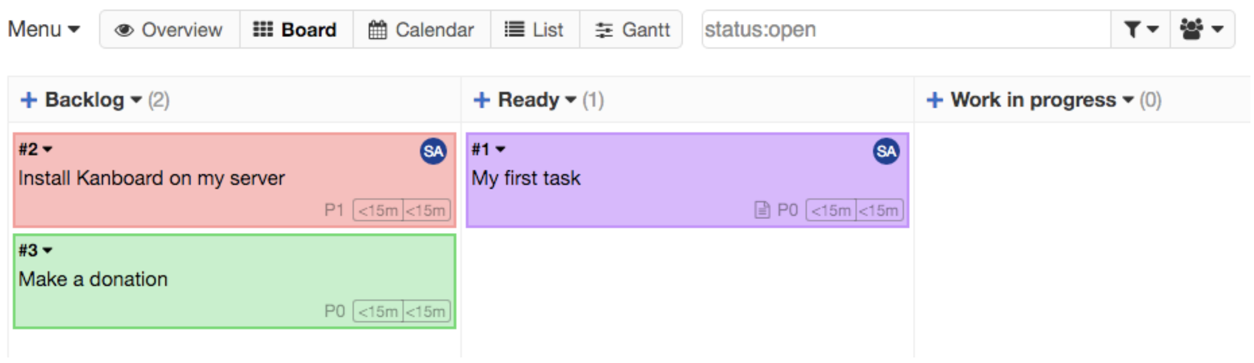
Note: Since Kanboard v1.2.8, people with two-factor authentication enabled must use API keys.

1.3 Boards

1.3.1 Project Views

For each project, tasks can be visualized with several views: **Board, Calendar, List and Gantt**. Each view shows the result of the filter box at the top.

Board View



- With this view, you can drag and drop tasks between columns easily.
- You can also use the keyboard shortcut **"v b"** to switch to the board view.
- Tasks with a shadow are recently modified.

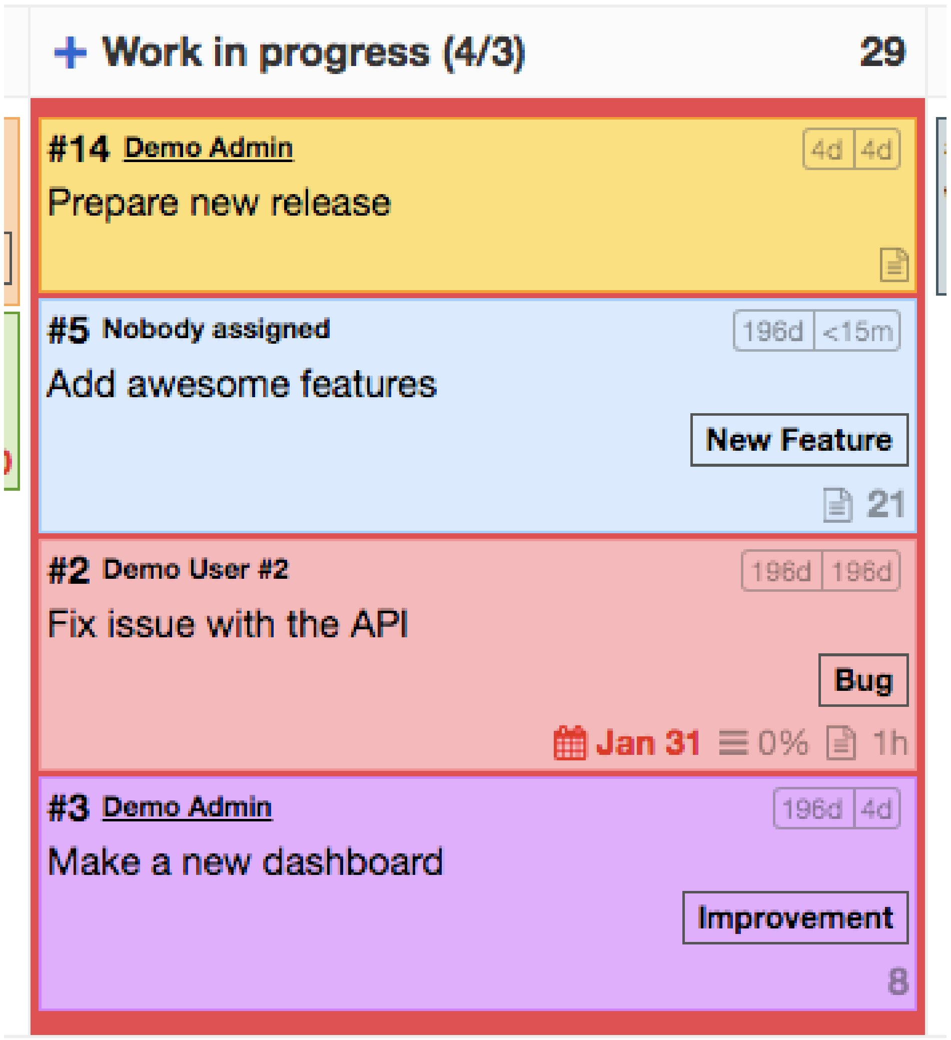
When the task limit is reached for a column, the background becomes red. That means there are too many tasks in progress at the same time.

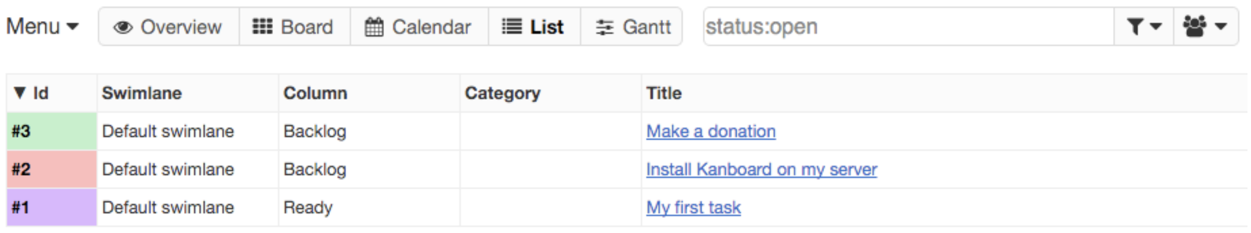
List View

- With this view, all results of your search are displayed in a table.
- You can also use the keyboard shortcut **"v l"** to switch to the list view.

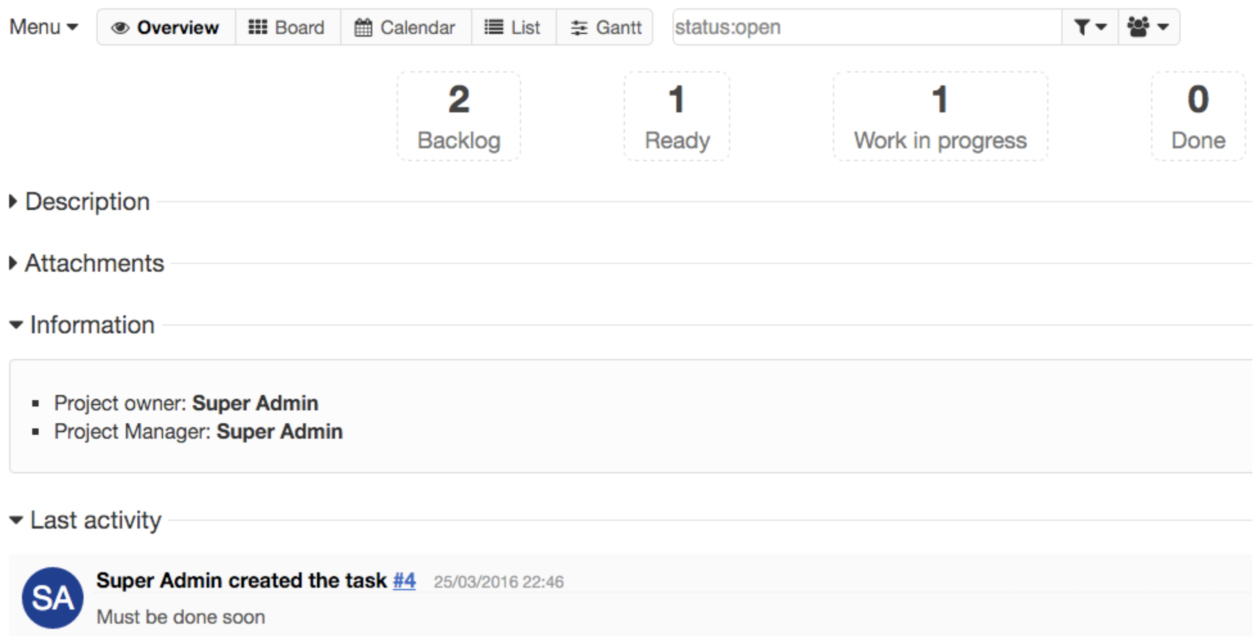
Project Overview

- View the description of the project.
- Attach and upload documents to the project.
- View list of project members.
- View the last activities of the project.





▼ Id	Swimlane	Column	Category	Title
#3	Default swimlane	Backlog		Make a donation
#2	Default swimlane	Backlog		Install Kanboard on my server
#1	Default swimlane	Ready		My first task



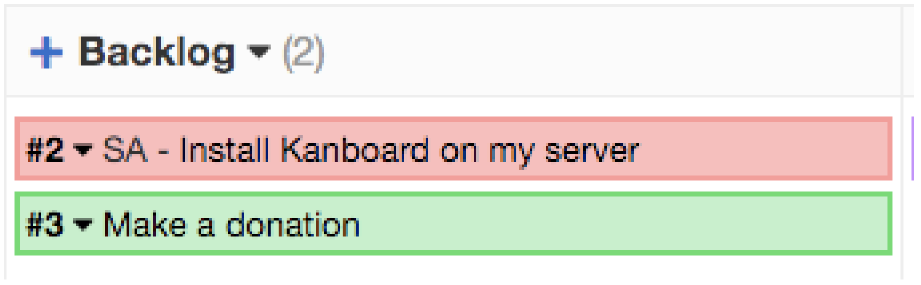
- Project owner: **Super Admin**
- Project Manager: **Super Admin**

Super Admin created the task #4 25/03/2016 22:46
Must be done soon

1.3.2 Collapsed and Expanded Cards

Tasks on the board can be displayed in collapsed or in expanded mode. Switching from one view to another can be done with the keyboard shortcut **"s"** or by using the drop-down menu on the left.

Collapsed mode:



- If the task is assigned to someone, the initials of the person are shown next to the task number.
- If the task title is too long, you can put your mouse over the task to show a tooltip with the full title.

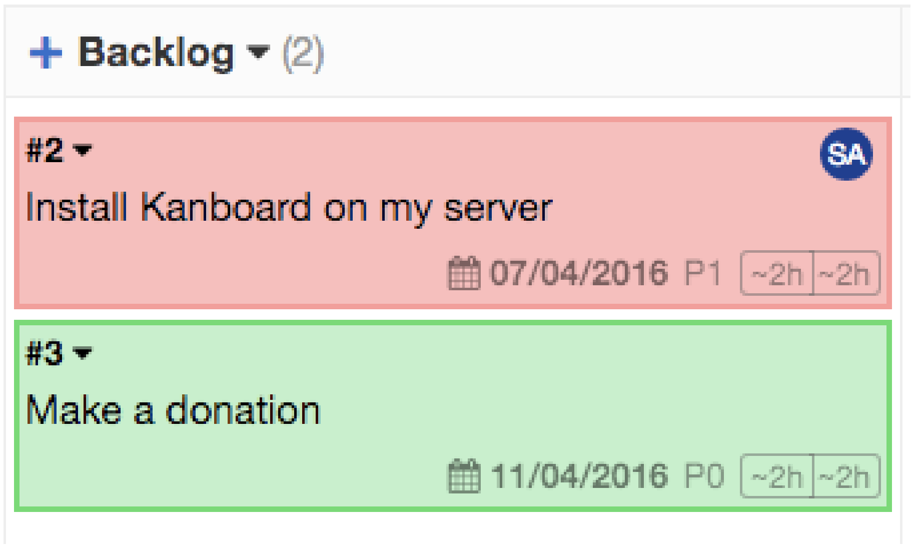
Expanded mode:

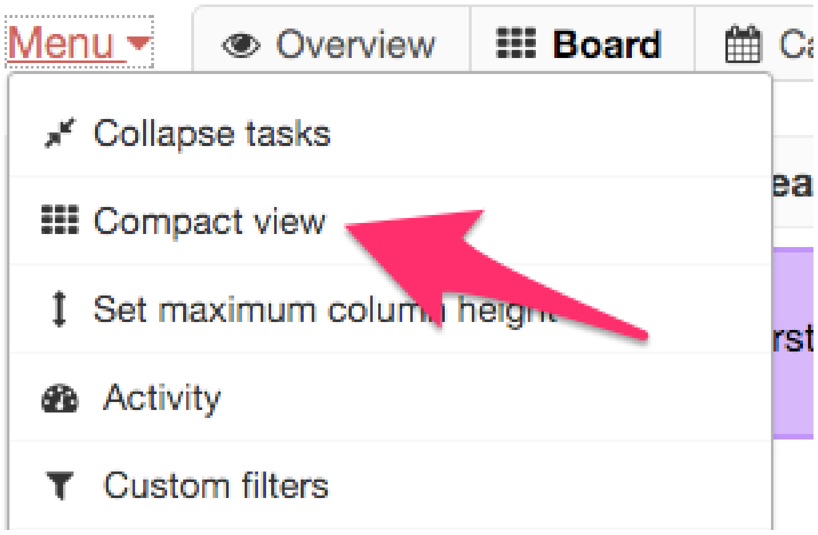
1.3.3 Horizontal Scrolling and Compact Mode

When the board cannot fit on your screen, a horizontal scroll bar will appear at the bottom.

However, it's possible to switch to the compact the view to display all columns in your screen.

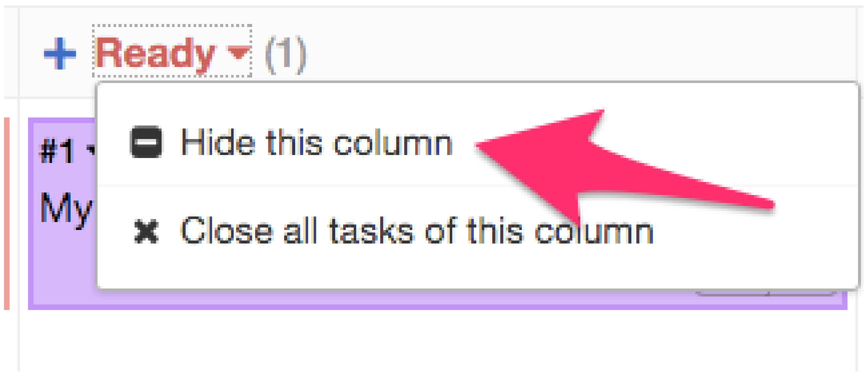
Switching between horizontal scrolling and compact view can be done with the keyboard shortcut **"c"** or by using the drop-down menu on the top left.


+ Backlog (2)
#2
SA
Install Kanboard on my server
07/04/2016 P1 ~2h ~2h
#3
Make a donation
11/04/2016 P0 ~2h ~2h

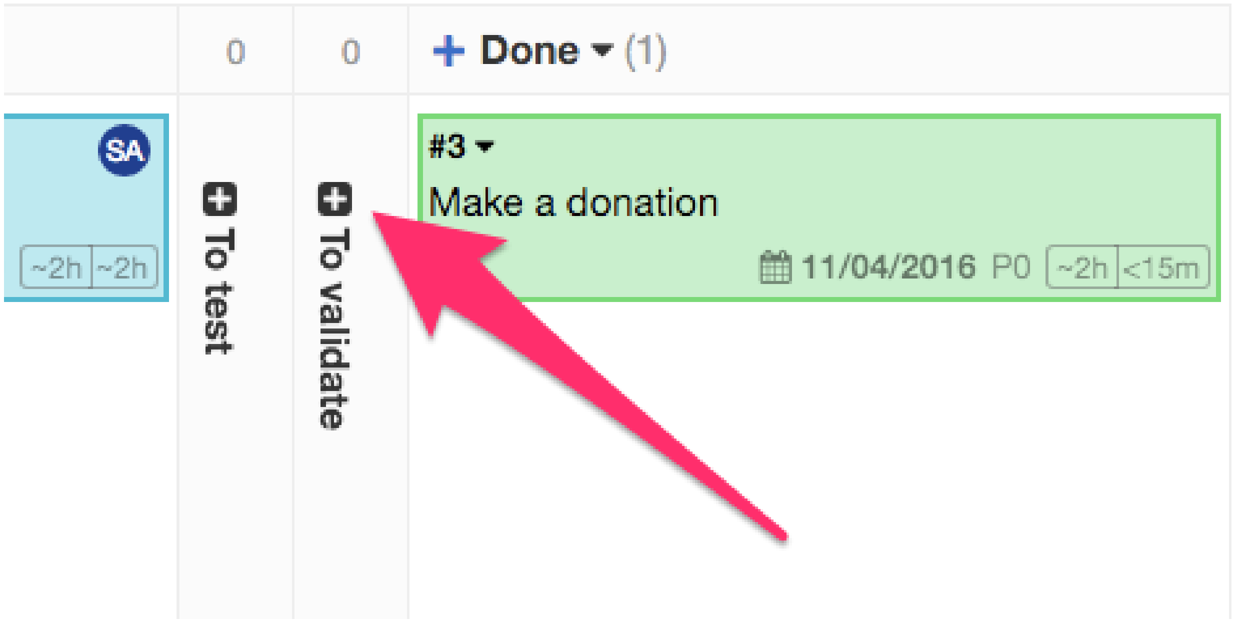

Menu
Overview
Board
Collapse tasks
Compact view
Set maximum column height
Activity
Custom filters

1.3.4 Show and Hide Columns

You can hide or display columns very easily on the board:



To hide a column, click on the column dropdown menu and choose "Hide this column":



To show the column again, click on the "plus icon".

1.4 Projects

1.4.1 Project Types

There are two types of projects:

Type	Description
Team Project	Project with user and group management
Private Project	Project that belongs to only one person, there is no user management

- Only Administrators and Application Managers can create team projects.
- Private projects can be created by anyone.

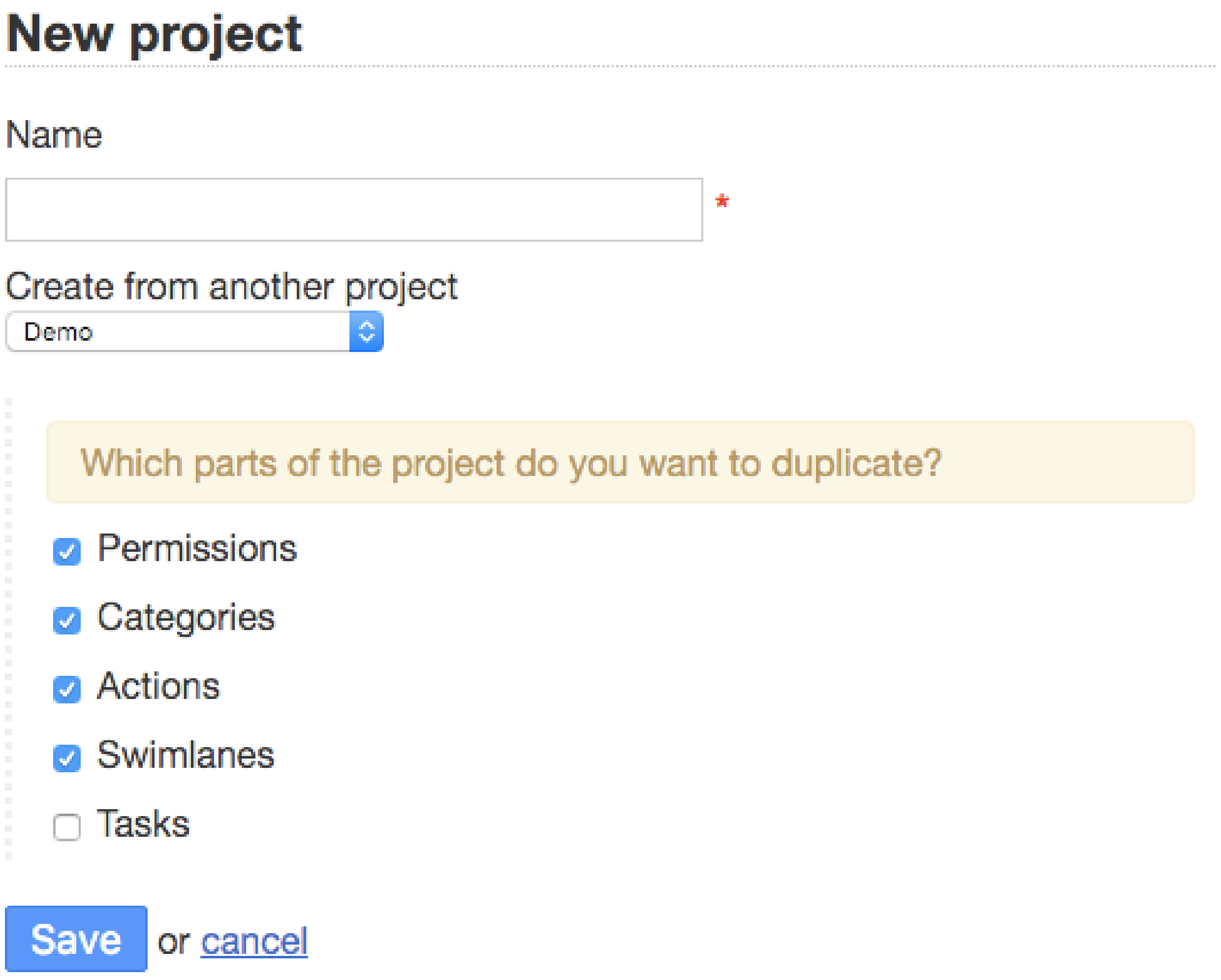
1.4.2 Creating Projects

Kanboard can handle multiple projects.

Creating Projects for Multiple Users

- Only administrators and managers can create those projects
- User management is available

From the dashboard, click on the link **New project**:


New project

Name *

Create from another project: Demo

Which parts of the project do you want to duplicate?

- [✓] Permissions
- [✓] Categories
- [✓] Actions
- [✓] Swimlanes
- [ ] Tasks

Fig. 1: Project creation form

It's very easy: you just have to find a name for your project!

Creating a Private Project

- Anybody can create a private project
- There is **NO** user management
- Only the owner and administrators can access the project

From the dashboard, click on the link **New private project**.

Creating Projects from Another Project

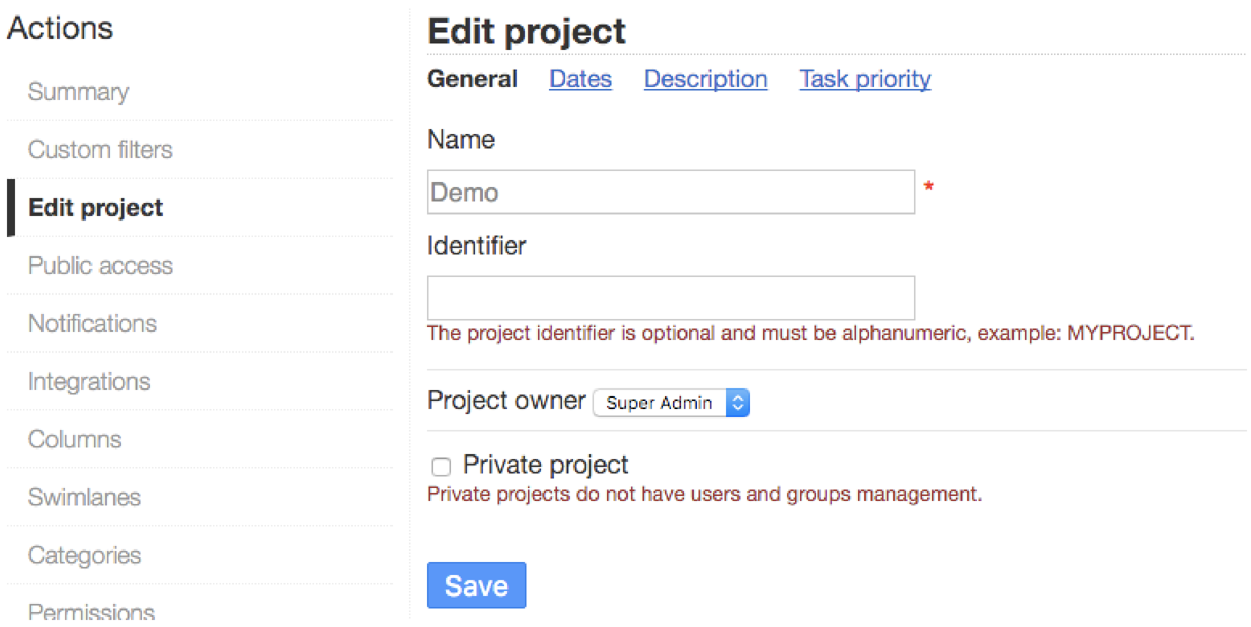
When you create a new project, you can choose to duplicate the properties of another project:

- Permissions
- Actions
- Swimlanes
- Categories
- Tasks

1.4.3 Editing Projects

Projects can be modified at any time.

To rename a project, just click on the link entitled **"Edit project"** on the left.



Name

Demo *

Identifier

The project identifier is optional and must be alphanumeric, example: MYPROJECT.

Project owner: Super Admin

☐ Private project

Private projects do not have users and groups management.

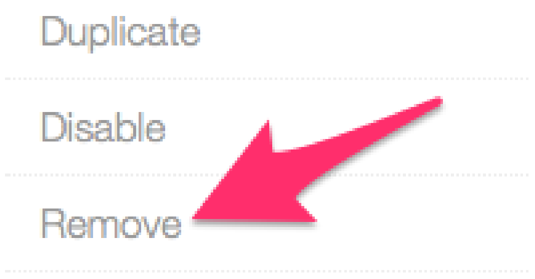
- The start date and end date are used to generate the project Gantt chart.
- The description is visible as a tooltip on the board and on the projects listing page.
- Administrators and project administrators can convert a private project to a multiple-user project by changing the checkbox "Private project".
- You can also convert a multiple-user project to a private project.

Note: When you make a project private, all existing users will still have access to the project. You must adjust the list of users according to your needs.

1.4.4 Removing Projects

To remove a project, you must be the manager of the project or an administrator.

Go to the **"Project settings"**, and from the menu on the left, at the bottom, choose **"Remove"**.



Removing a project removes all tasks that belong to this project.

1.4.5 Project Permissions

Each project is isolated from other projects. Project access must be allowed by the project owner.

Each user and each group can have a different role assigned. There are 3 types of roles for projects:

- Project Manager
- Project Member
- Project Viewer

Only administrators have access to everything.

Role assignments are visible in **Project Settings > Permissions**:

Private projects cannot define permissions.

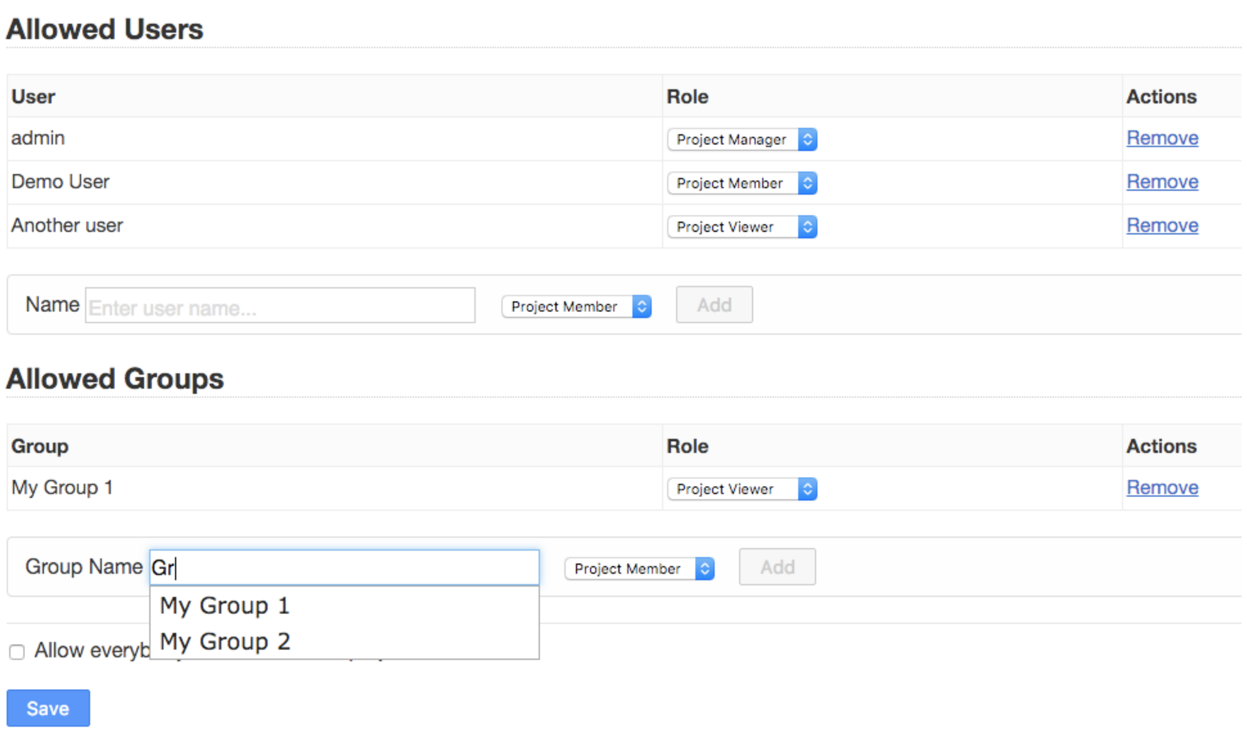
1.4.6 Custom Project Roles

You can create custom project roles to apply a set of specific restrictions on the people that belong to this role. These custom roles are defined for each project.

A custom role inherits from the project member role. For example, you may want to create a custom role to force someone to follow a process. You can have a group of people that are allowed to move tasks only from the column "Work in progress" to the column "Done".

Available Restrictions

- Project Restrictions:
  - Task creation is not permitted
  - Closing or opening a task is not permitted



Allowed Users

User	Role	Actions
admin	Project Manager	Remove
Demo User	Project Member	Remove
Another user	Project Viewer	Remove

Name Enter user name... Project Member Add

Allowed Groups

Group	Role	Actions
My Group 1	Project Viewer	Remove

Group Name Gr Project Member Add

My Group 1
My Group 2

Allow everyb

Save

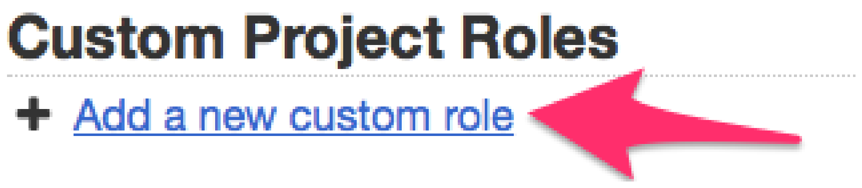
    – Moving a task is not permitted

- Columns Restrictions:
    – Task creation is **allowed** only for a specific column
    – Task creation is **blocked** only for a specific column
    – Closing or opening a task is **allowed** only for a specific column
    – Closing or opening a task is **blocked** only for a specific column
- Moving tasks only between specified columns

Configuration

1) Create a new custom role

From the project settings, click on the left on the menu **Custom Roles** and at the top of the page click on **Add a new custom role**.



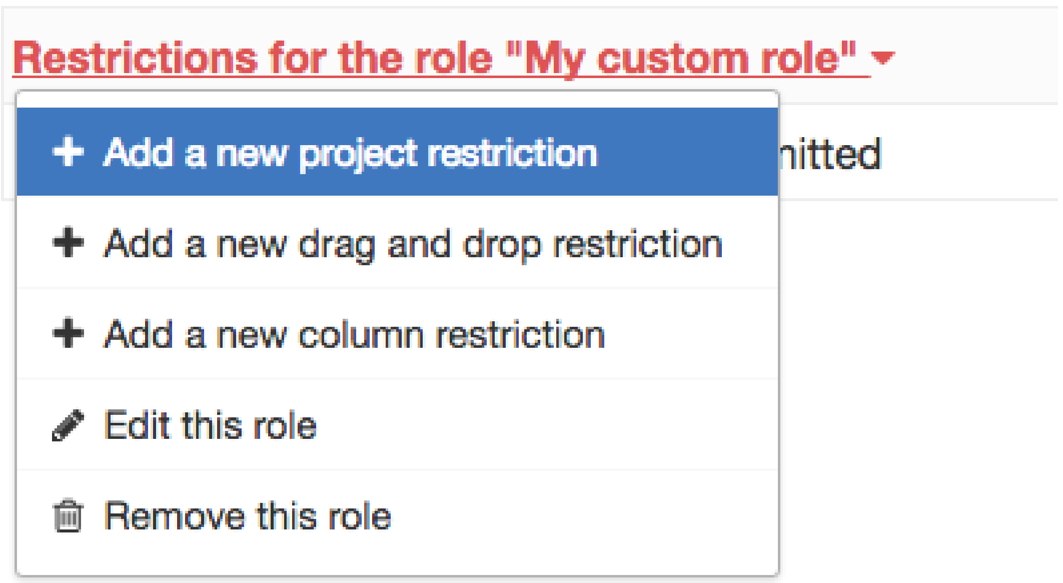
Give a name to the role and submit the form.

2) Add a restriction to the role

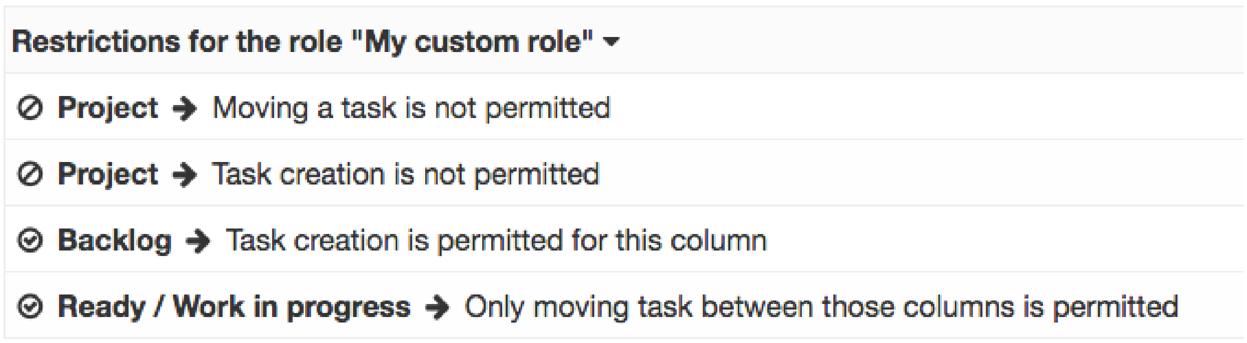
There are different kinds of restrictions:

- Project restrictions
- Drag and drop restrictions
- Column restrictions

You can click on the drop-down menu on the table to add a new restriction:


Restrictions for the role "My custom role"
Add a new project restriction
Add a new drag and drop restriction
Add a new column restriction
Edit this role
Remove this role
nitted

3) List of restrictions

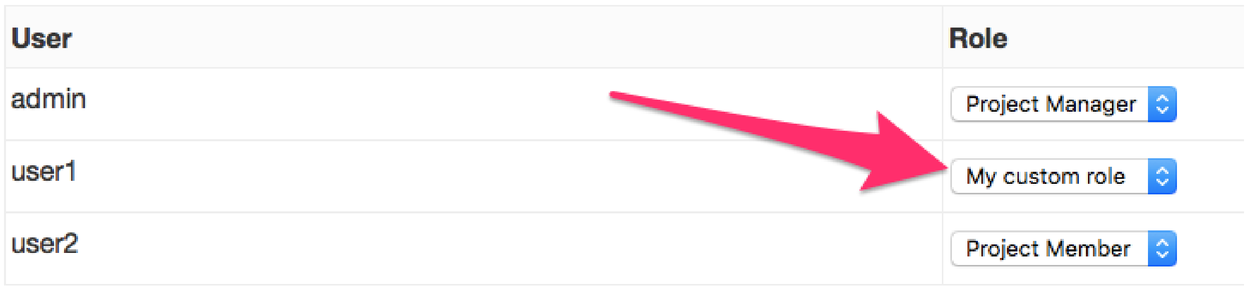


Restrictions for the role "My custom role"
Project → Moving a task is not permitted
Project → Task creation is not permitted
Backlog → Task creation is permitted for this column
Ready / Work in progress → Only moving task between those columns is permitted

For example, this role is able to create tasks only in the column "Backlog" and to move tasks between the column "Ready" and "Work in progress".

4) Assign the role to someone

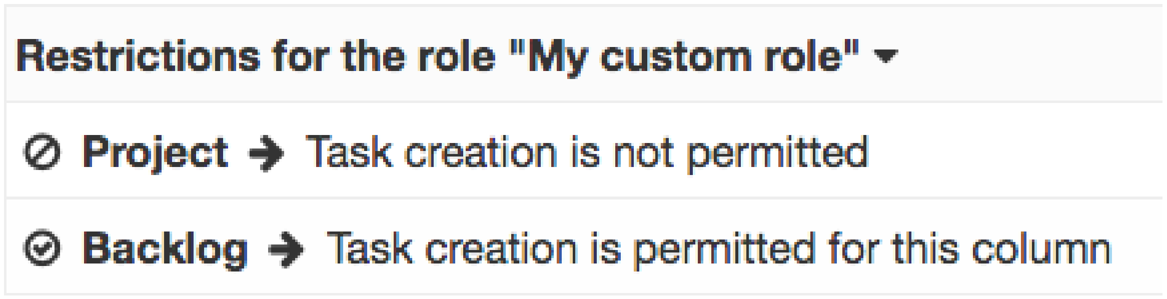
Go to the “permissions” section on the left menu and assign the desired role to the user.



User	Role
admin	Project Manager
user1	My custom role
user2	Project Member

Examples

Allow people to create tasks only in specific columns

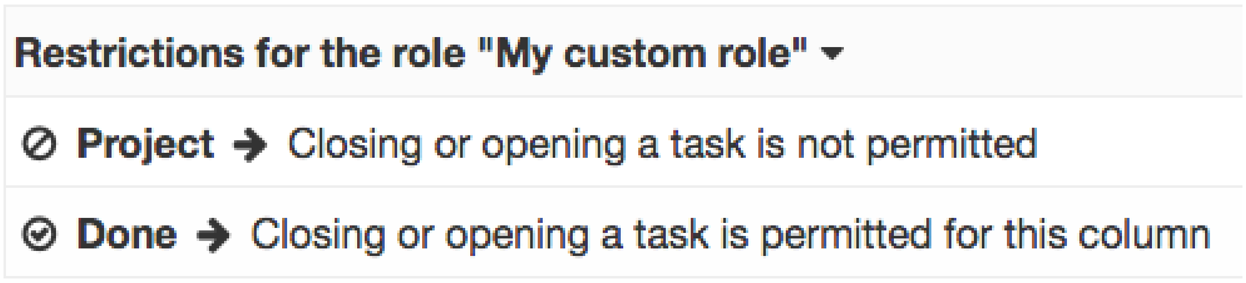


Restrictions for the role "My custom role"

- Project → Task creation is not permitted
- Backlog → Task creation is permitted for this column

- Users that belong to this role will be able to create new tasks only in the column “Backlog”.
- The combination of the 2 rules is important, otherwise that will not work.

Allow people to change the task status only in specific columns

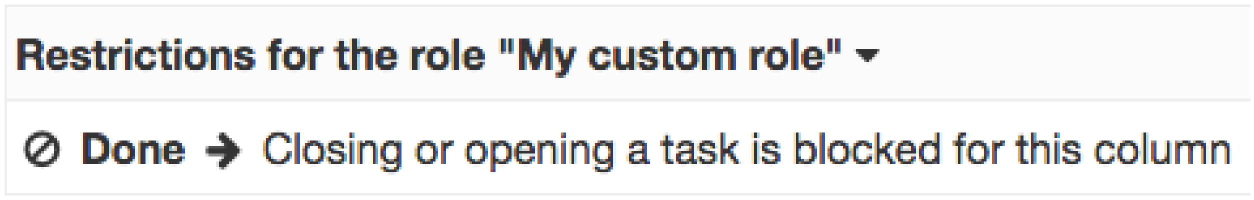


Restrictions for the role "My custom role"

- Project → Closing or opening a task is not permitted
- Done → Closing or opening a task is permitted for this column

- Users that belong to this role will be able to change the task status in the column “Backlog”.
- Tasks with the status open are visible on the board and tasks with the status closed is hidden by default on the board.

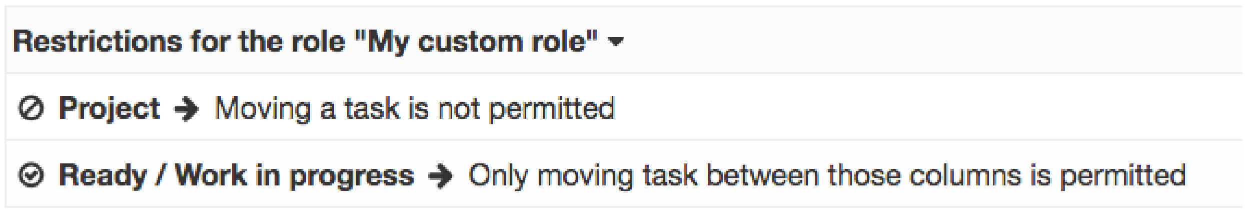
Do not allow people to change task status in a specific column


Restrictions for the role "My custom role"

Done → Closing or opening a task is blocked for this column

Users that belong to this role won't be able to change the task status in the column "Done". However, it will be possible in other columns.

Allow people to move tasks only between specific columns


Restrictions for the role "My custom role"

Project → Moving a task is not permitted

Ready / Work in progress → Only moving task between those columns is permitted

Users that belong to this role will be able to move tasks only between the column "Ready" and "Work in progress".

1.4.7 Sharing Boards and Tasks

By default, boards are private, but it's possible to make a board public.

A public board **cannot be modified: it has read-only access**. Access is protected by a random token. Only people who have the correct URL can see the board.

Public boards are automatically refreshed every 60 seconds. Task details are also available in read-only mode.

Usage examples:

- Share your board with someone outside of your organization
- Display the board on a large screen in your office

Enable Public Access

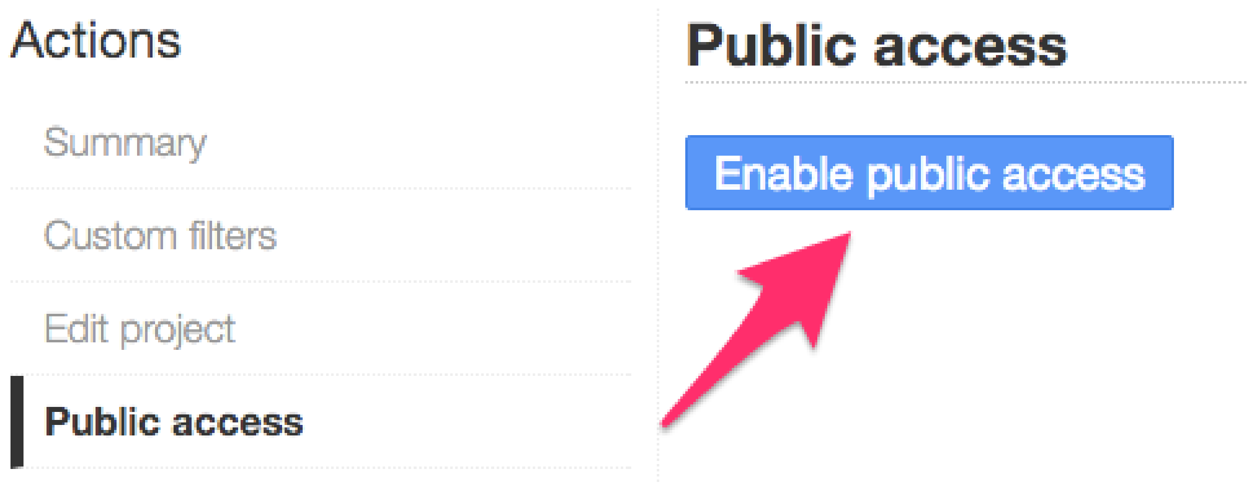
Select your project, then click on **"Public access"** and finally click on the button **"Enable public access"**.

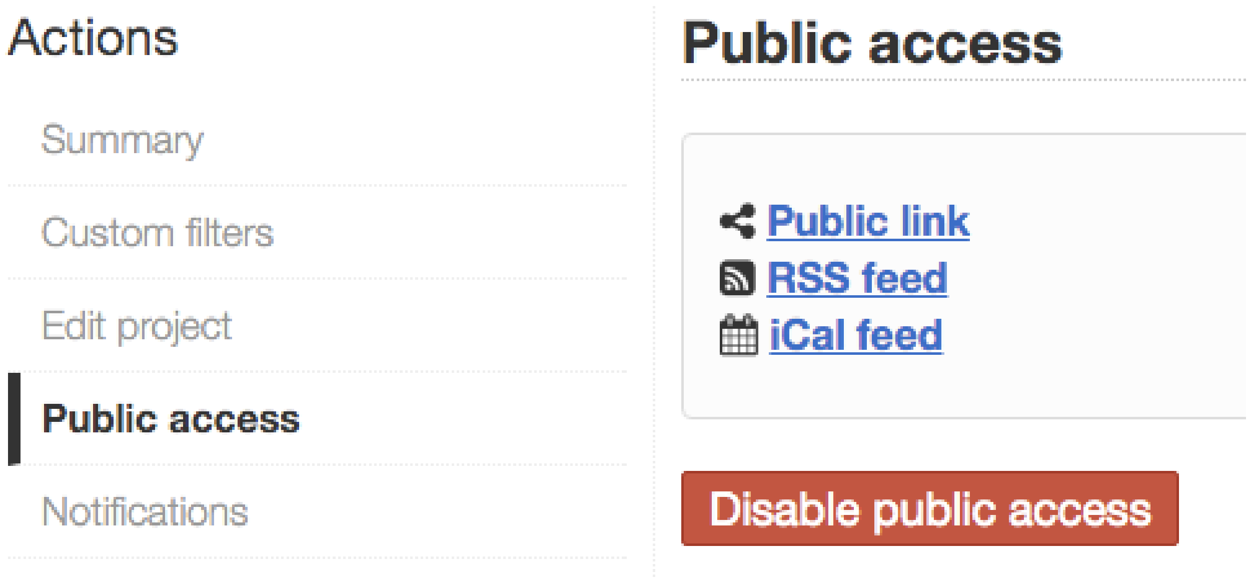
When public access is enabled, a couple of links are generated:

- Public board view
- RSS feed subscription link
- iCalendar subscription link

You can also disable public access whenever you want.

Each time you enable or disable public access, a new random token is generated. **The previous links will not work anymore**!


Actions
Summary
Custom filters
Edit project
Public access
Public access
Enable public access


Actions
Summary
Custom filters
Edit project
Public access
Notifications
Public access
Public link
RSS feed
iCal feed
Disable public access

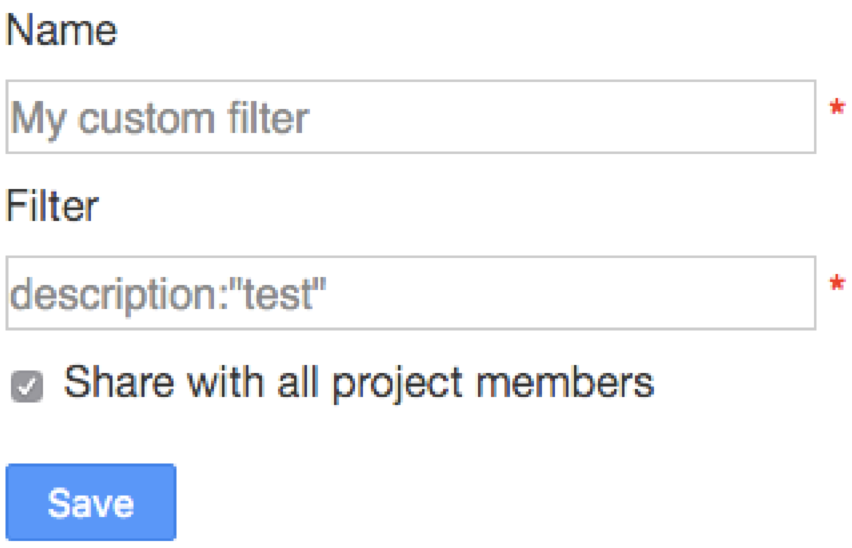
1.4.8 Custom Filters

Custom filters allow you to save any search query. In this way, you can extend the default filters easily and save most used search queries.

- Custom filters are stored by project and associated to the creator.
- If the creator is project manager, he can choose to share the filter with other project members.

Filter Creation

Go to the action drop-down or in the project settings and choose **custom filters**:


Name

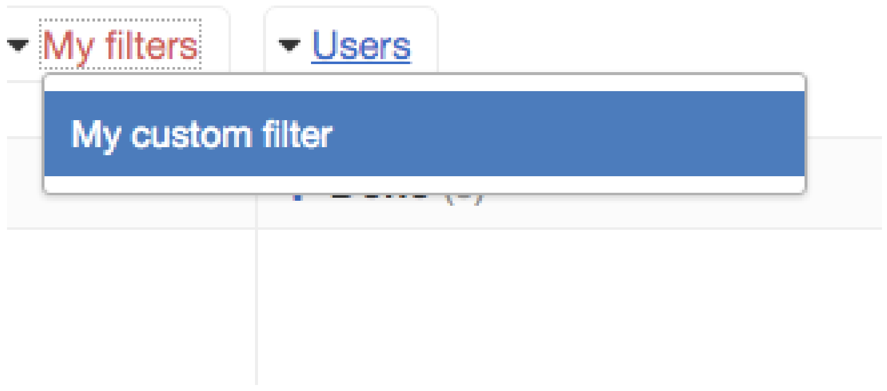
My custom filter *

Filter

description:"test" *

✓ Share with all project members

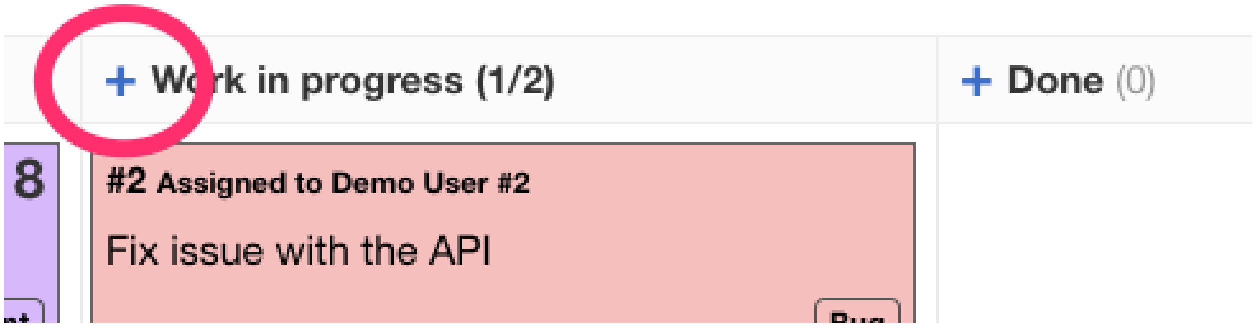
After creating your filter, it will appear on the board next to the default filters:



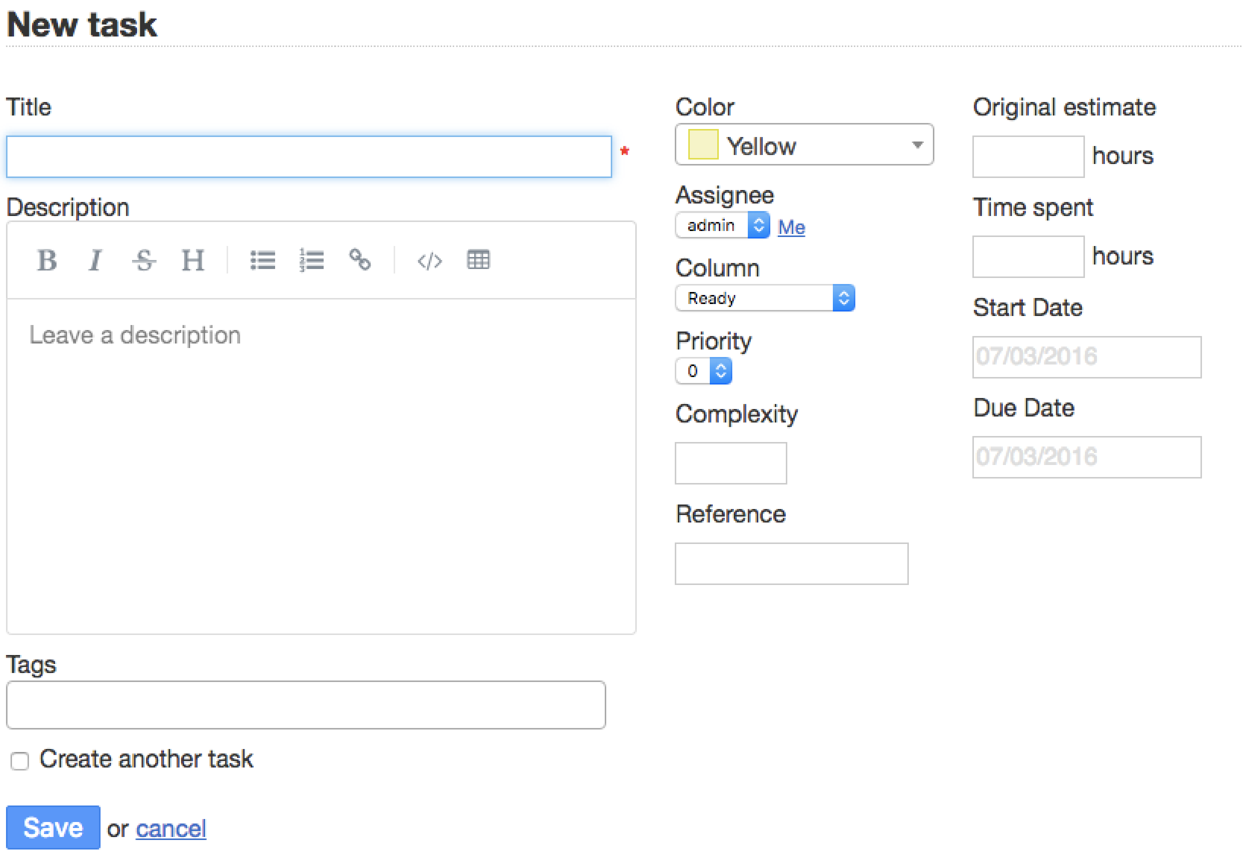
1.5 Tasks

1.5.1 Creating Tasks

From the board, click on the plus sign next to the column name:



Then the task creation form appears:



- **Title**: The title of your task, which will be displayed on the board.
- **Description**: Description that use the Markdown syntax.
- **Tags**: The list of tags associated to tasks.
- **Create another task**: Check this box if you want to create a similar task (some fields will be pre-filled).

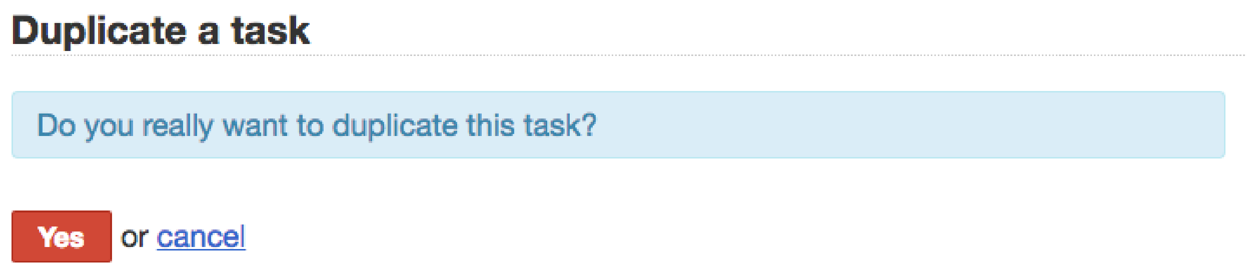
- **Color**: Choose the color of the card.
- **Assignee**: The person that will work on the task.
- **Category**: Only one category can be assigned to a task (visible only if the projects have categories).
- **Column**: The column where the task will be created, your task will be positioned at the bottom.
- **Priority**: Task priority, the range can be defined in the project settings, default values are P0 to P3.
- **Complexity**: Used in agile project management (Scrum), the complexity or story points is a number that tells the team how hard the story is. Often, people use the Fibonacci series.
- **Reference**: External ID for the task, for example it can be ticket number that come from another system
- **Original Estimate**: Estimation in hours to complete the task.
- **Time Spent**: Time spent working on the task.
- **Start Date**: This is a date time field.
- **Due Date**: Overdue tasks will have a red due date and upcoming due dates will be black on the board. Several date format are accepted in addition to the date picker.

With the preview link, you can see the task description converted from the Markdown syntax.

1.5.2 Duplicating and Moving Tasks

Duplicate a task into the same project

Go to the task view and choose **Duplicate** on the left.



Duplicate a task

Do you really want to duplicate this task?

A new task will be created with the same properties as the original.

Duplicate a task to another project

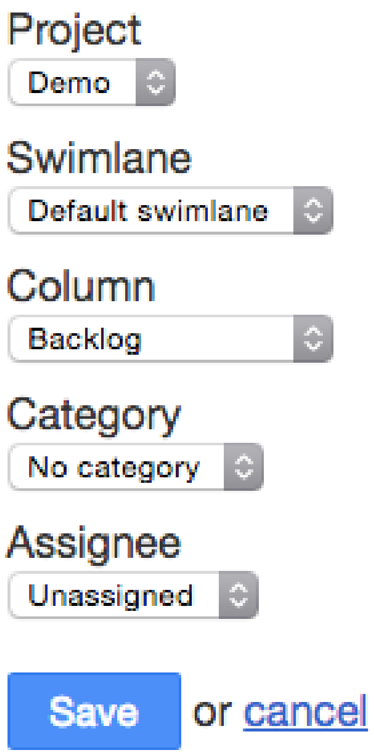
Go to the task view and choose **Duplicate to another project**.

Only projects where you are members will be shown in the drop-down.

Before to copy the tasks, Kanboard will ask you the destination properties that are not common between the source and destination project.

Basically, you need to define:

- The destination swim lane
- The column
- The category



- The assignee

Move a task to another project

Go to the task view and choose **Move to another project**.

Moving a task to another project work in the same way as the duplication, you have to choose the new properties of the task.

List of duplicated properties

- title
- description
- date_due
- color_id
- project_id
- column_id
- owner_id
- score
- category_id
- time_estimated

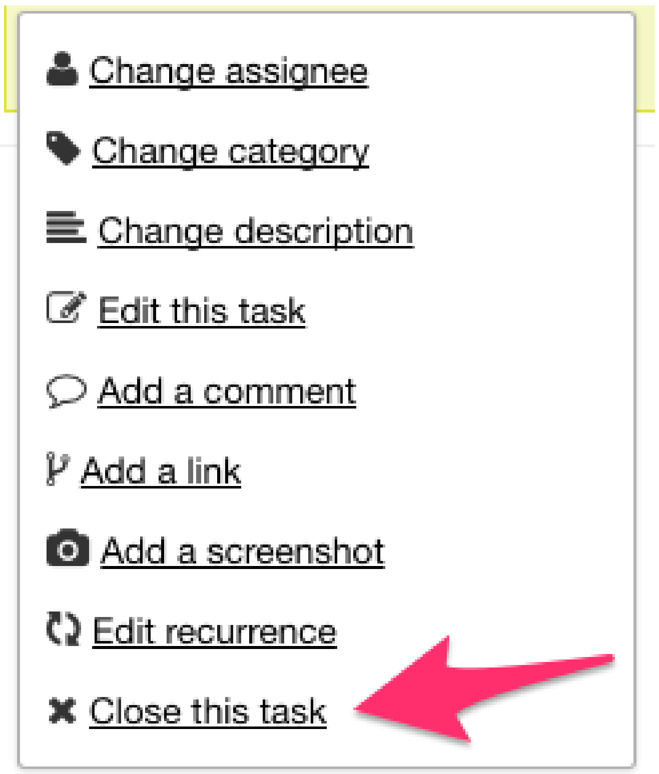
- swimlane_id
- recurrence_status
- recurrence_trigger
- recurrence_factor
- recurrence_timeframe
- recurrence_basedate

1.5.3 Closing Tasks

When a task is closed, it is hidden from the board.

However, you can always access to the list of closed tasks by using the query **status:closed** in any search form or simply choose **Closed tasks** from the filter drop-down.

There are two different ways to close a task, from the task drop-down menu on the board:



Or from the task sidebar menu in the task detail view:

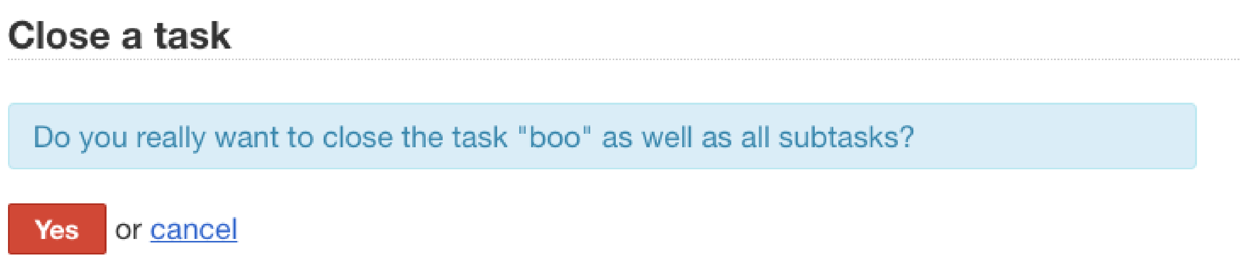
Note: When you close a task, all sub-tasks not completed will be changed to the status “Done”.

1.5.4 Internal Task Links

Tasks can be linked together with pre-defined relationships:

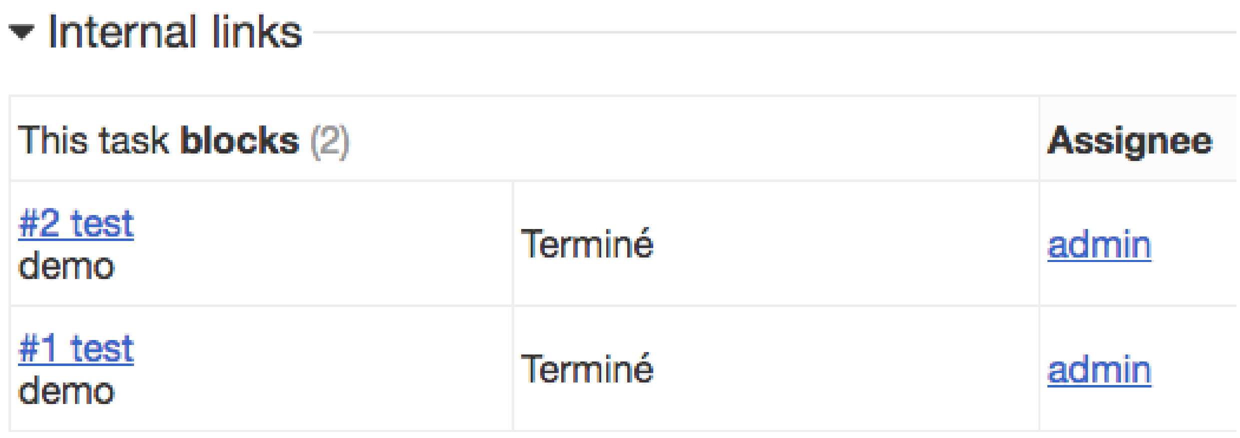
This is also possible to link tasks across projects.

The default relationships are:


Close a task

Do you really want to close the task "boo" as well as all subtasks?

Yes or cancel


Internal links

This task **blocks** (2)		**Assignee**
#2 test demo	Terminé	admin
#1 test demo	Terminé	admin

- **relates to**
- **blocks** | is blocked by
- **is blocked by** | blocks
- **duplicates** | is duplicated by
- **is duplicated by** | duplicates
- **is a child of** | is a parent of
- **is a parent of** | is a child of
- **targets milestone** | is a milestone of
- **is a milestone of** | targets milestone
- **fixes** | is fixed by
- **is fixed by** | fixes

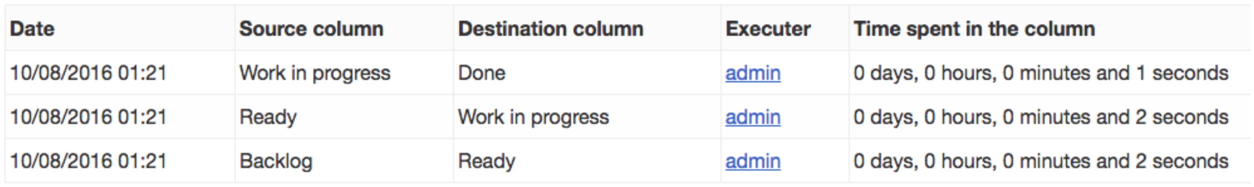
Those labels can be changed in the application settings.

1.5.5 Task Transitions

Each movement of a task between columns is recorded in the database.

Available from the task view, you can see that information:

- Date of the action
- Source column



Date	Source column	Destination column	Executer	Time spent in the column
10/08/2016 01:21	Work in progress	Done	admin	0 days, 0 hours, 0 minutes and 1 seconds
10/08/2016 01:21	Ready	Work in progress	admin	0 days, 0 hours, 0 minutes and 2 seconds
10/08/2016 01:21	Backlog	Ready	admin	0 days, 0 hours, 0 minutes and 2 seconds

- Destination column
- Executor (users that moves the task)
- Time spent in the origin column

1.5.6 Recurring Tasks

To fit with the Kanban methodology, the recurring tasks are not based on a date but on board events.

- Recurring tasks are duplicated to the first column of the board when the selected events occur
- The due date can be recalculated automatically
- Each task records the task id of the parent task that created it and the child task created

Configuration

Go to the task view page or use the drop-down menu on the board, then select **Edit recurrence**.

There are 3 triggers that currently create a new recurring task:

- Moving a task from the first column
- Moving a task to the last column
- Closing the task

Due dates, if set on the current task, can be recalculated by a given factor of days, months or years. The base date for the calculation of the new due date can be either the existing due date, or the action date.

1.5.7 Adding Screenshots

You can copy and paste images directly in Kanboard to save time. These images are uploaded as attachments to the task.

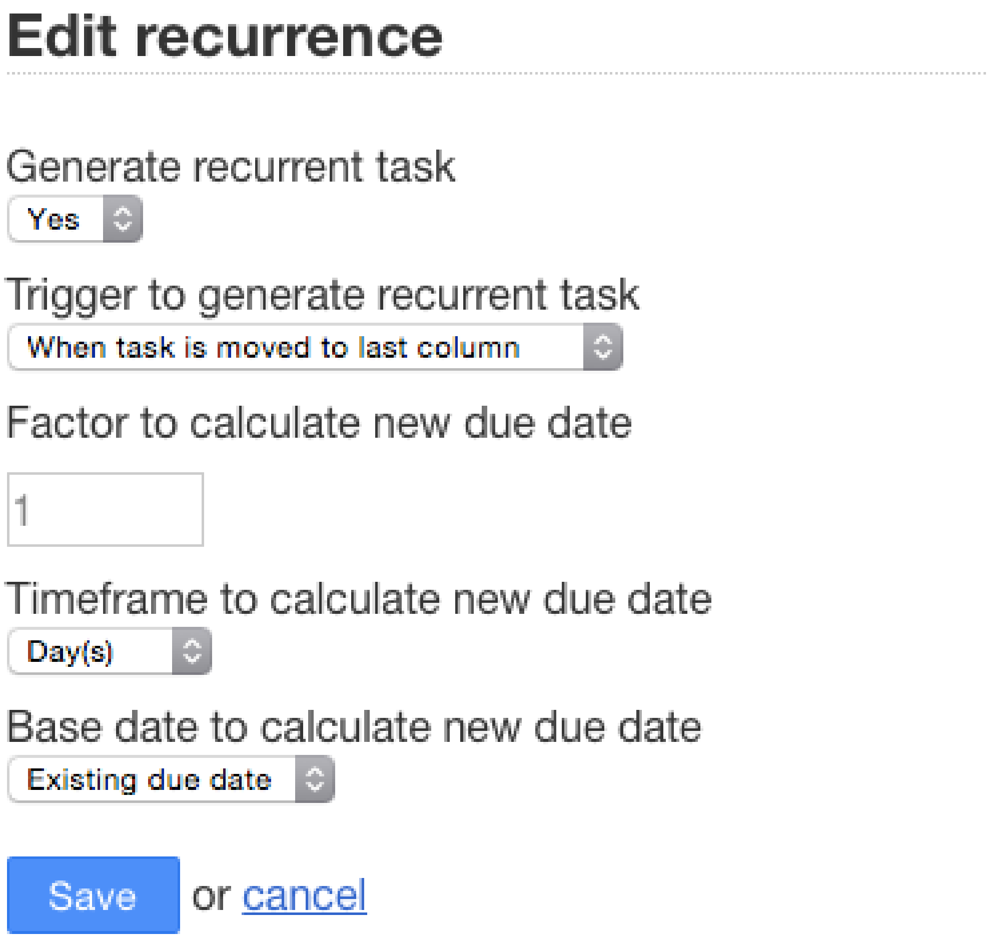
This is especially useful for taking screenshots to describe an issue for example.

You can add screenshots directly from the board by clicking on the dropdown menu or in the task view page.

To add a new image, take your screenshot and paste with CTRL+V or Command+V:

On Mac OS X, you can use those shortcuts to take screenshots:

- Command-Control-Shift-3: Take a screenshot of the screen, and save it to the clipboard
- Command-Control-Shift-4, then select an area: Take a screenshot of the area and save it to the clipboard
- Command-Control-Shift-4, then space, then click a window: Take a screenshot of a window and save it to the clipboard



Edit recurrence

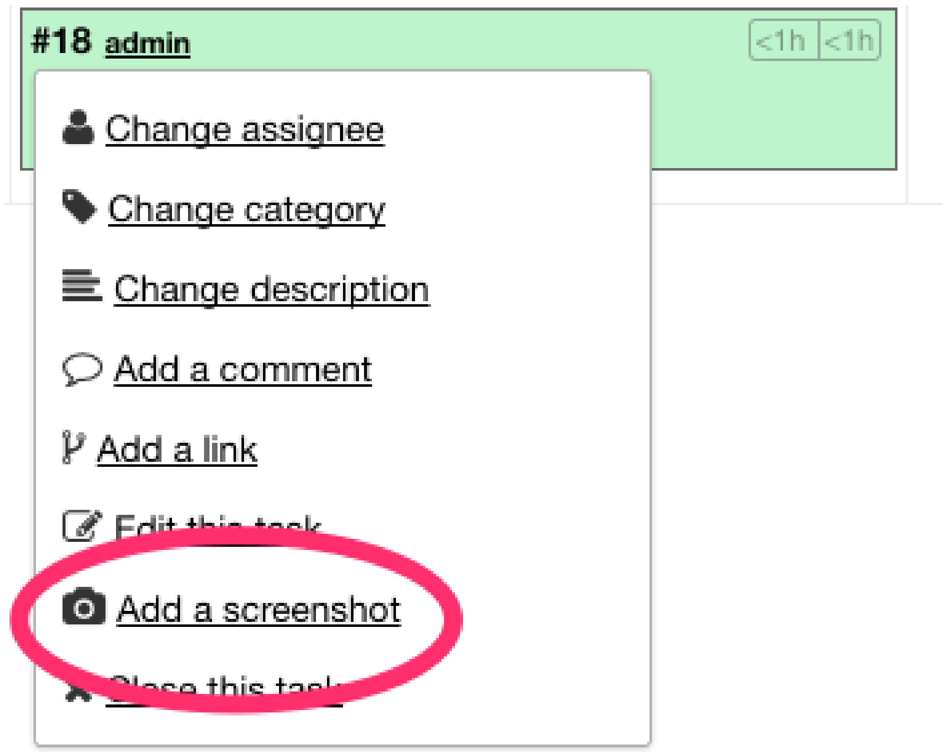
Generate recurrent task: Yes

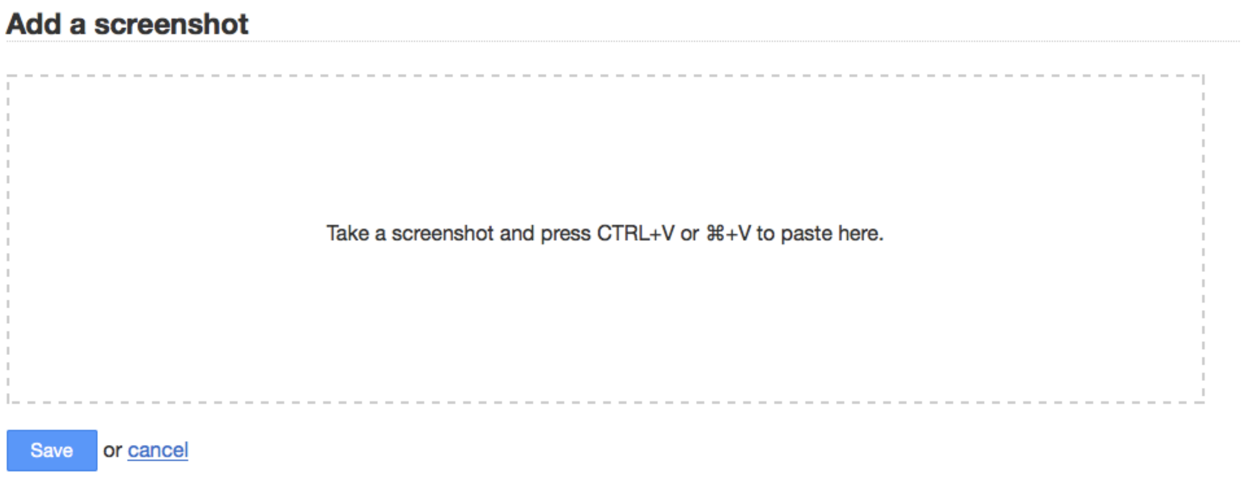
Trigger to generate recurrent task: When task is moved to last column

Factor to calculate new due date: 1

Timeframe to calculate new due date: Day(s)

Base date to calculate new due date: Existing due date





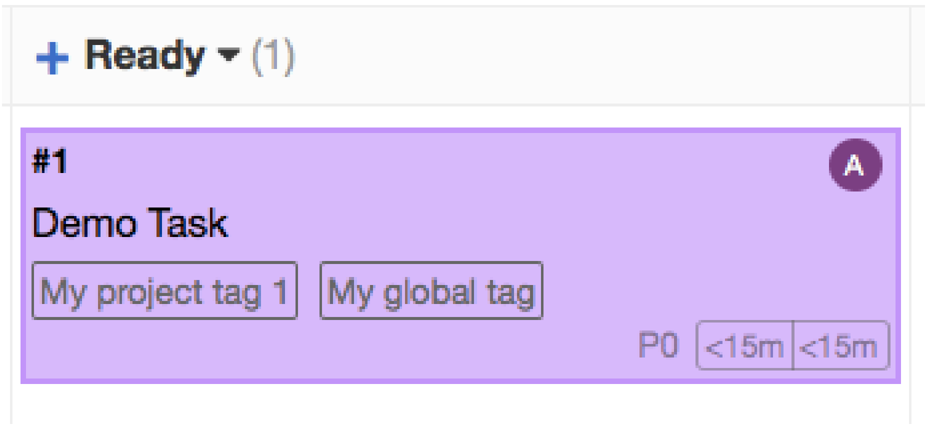
Take a screenshot and press CTRL+V or ⌘+V to paste here.

There are also several third-party applications that can be used to take screenshots with annotations and shapes.

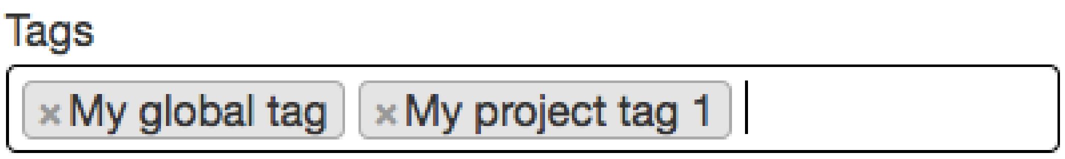
Warning: This feature doesn't work with all browsers. It does not work with Safari due to this bug: https://bugs.webkit.org/show_bug.cgi?id=49141

1.5.8 Tags

With Kanboard, you can associate one or many tags to a task. You can define tags globally for all projects or only for a specific project.

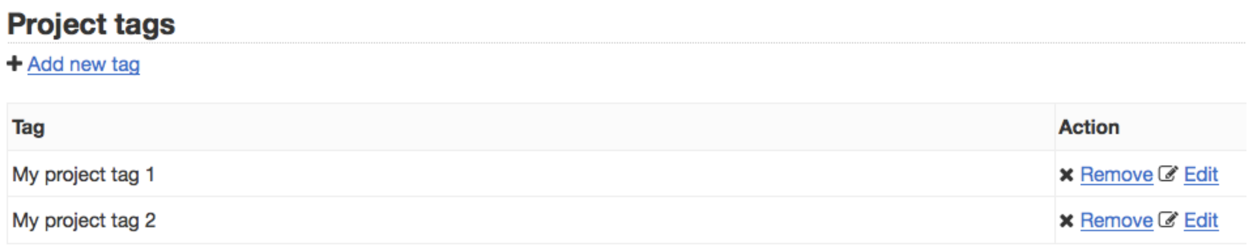


From the task form, you can enter the desired tags:



The auto-completion form will show up to suggest available tags.

You can also create tags directly from the task form. By default, when you create tags from a task form they are associated to the current project:



Project tags

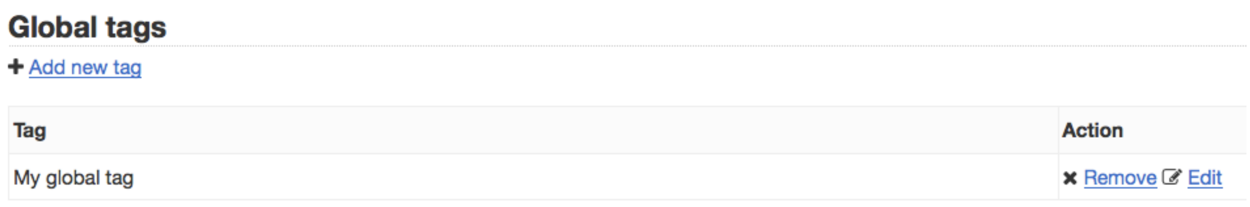
+ Add new tag

Tag	Action
My project tag 1	✖ Remove Edit
My project tag 2	✖ Remove Edit

All tags can be managed in the project settings.

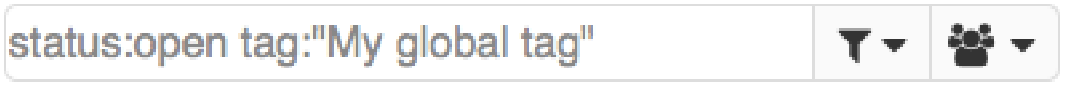
To define tags globally for all projects, go to the application settings:

To search tasks based on tags, just use the attribute "tag":


Global tags

+ Add new tag

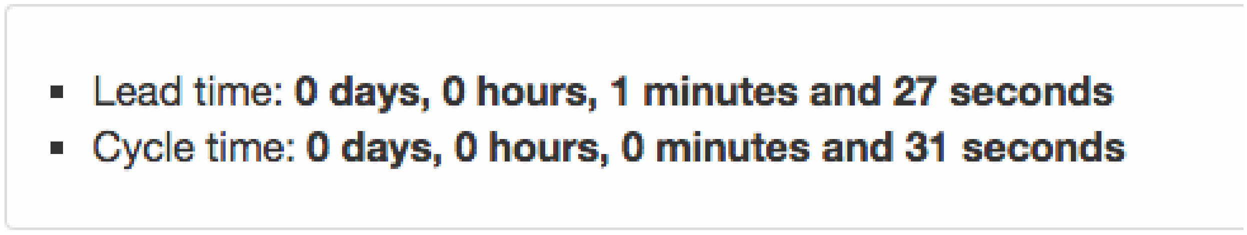
Tag	Action
My global tag	✖ Remove Edit


status:open tag:"My global tag"

1.5.9 Analytics

Each task has an analytics section available from the left menu in the task view.

Lead and Cycle Time


- Lead time: **0 days, 0 hours, 1 minutes and 27 seconds**
- Cycle time: **0 days, 0 hours, 0 minutes and 31 seconds**

- The lead time is the time between the task creation and the date of completion (task closed).
- The cycle time is the time between the start date and the date of completion.
- If the task is not closed the current time is used instead of the date of completion.
- If the start date is not specified, the cycle time is not calculated.

Note: You can configure an automatic action to define the start date automatically when you move a task to the column of your choice.

Time Spent Into Each Column

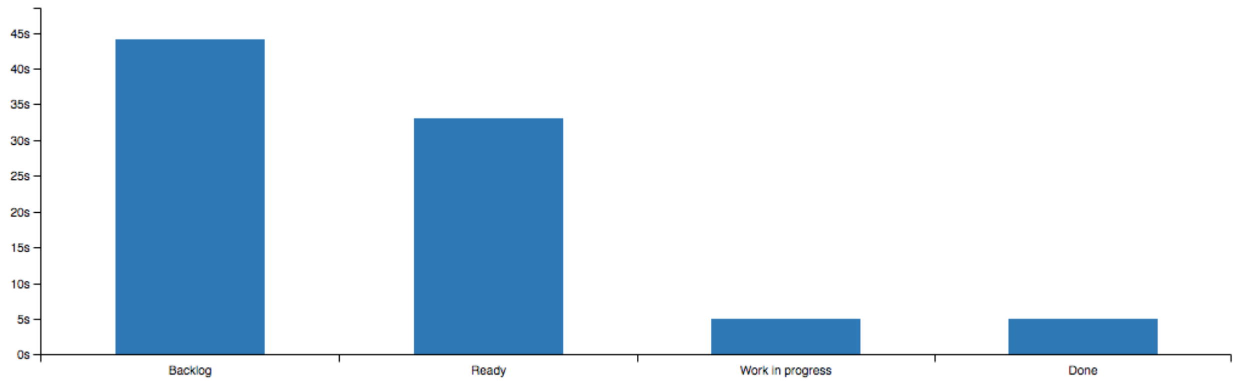
- This chart shows the total time spent into each column for the task.
- The time spent is calculated until the task is closed.

1.6 Subtasks

Subtasks are useful to split the work of a task.

Each subtask:

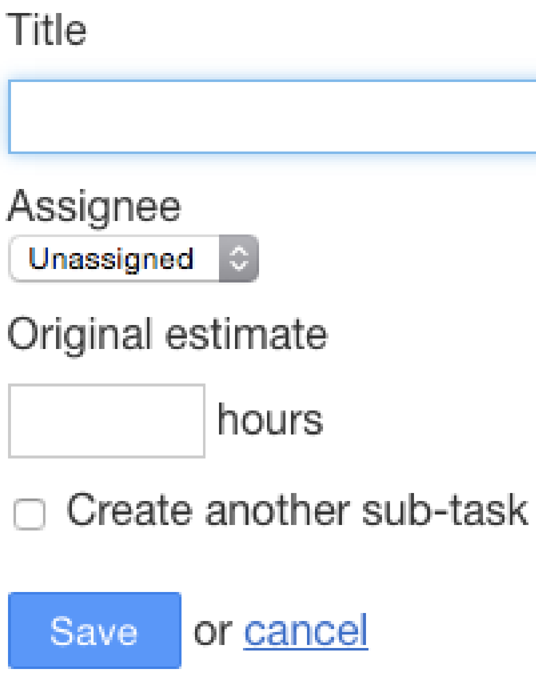
- Can be assigned to a project member
- Have 3 different statuses: **Todo**, **In progress**, **Done**



- Have time tracking information: **time spent** and **time estimated**
- Be ordered by position

1.6.1 Creating Subtasks

From the task view, on left sidebar click on **Add a subtask**:

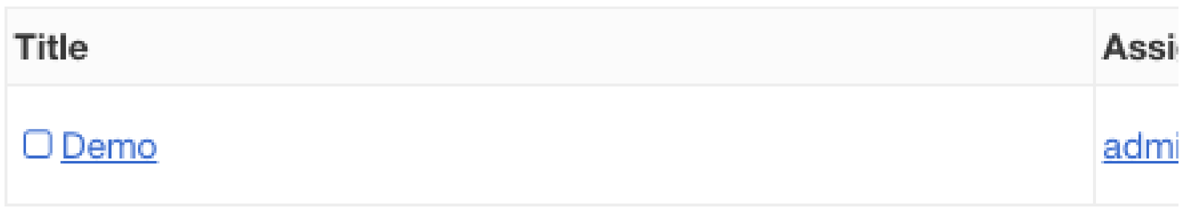


You can also add a subtask quickly by entering only the title:

1.6.2 Changing Subtask Status

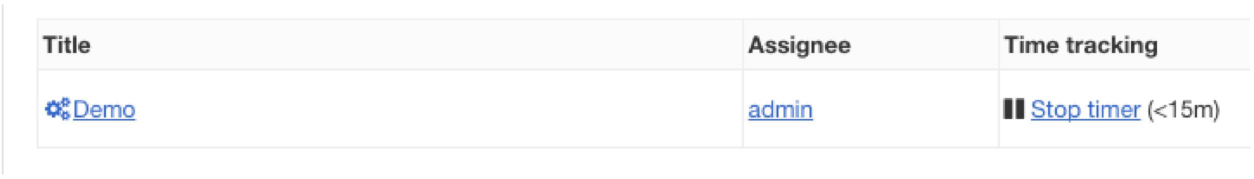
When you click on the subtask title the status change:

The icon before the title is updated according to the status.



Title	Assi
▢ Demo	admi

Type here to create a new sub-task * Add



Title	Assignee	Time tracking
Demo	admin	Stop timer (<15m)

Note: When the task is closed, all subtasks are changed to the status **Done**.

1.6.3 Subtask Timer

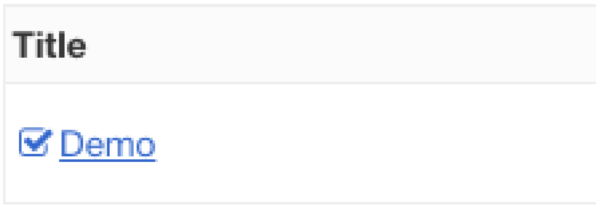
- Each time a subtask is in progress, the timer is also started. The timer can be started and stopped at any time.
- The timer records the time spent on the subtask automatically. You can also change manually the value of the time spent field when you edit a subtask.
- The time calculated is rounded to the nearest quarter. This information is recorded in a separate table.
- The task time spent and time estimated is updated automatically according to the sum of all subtasks.

1.7 Swimlanes

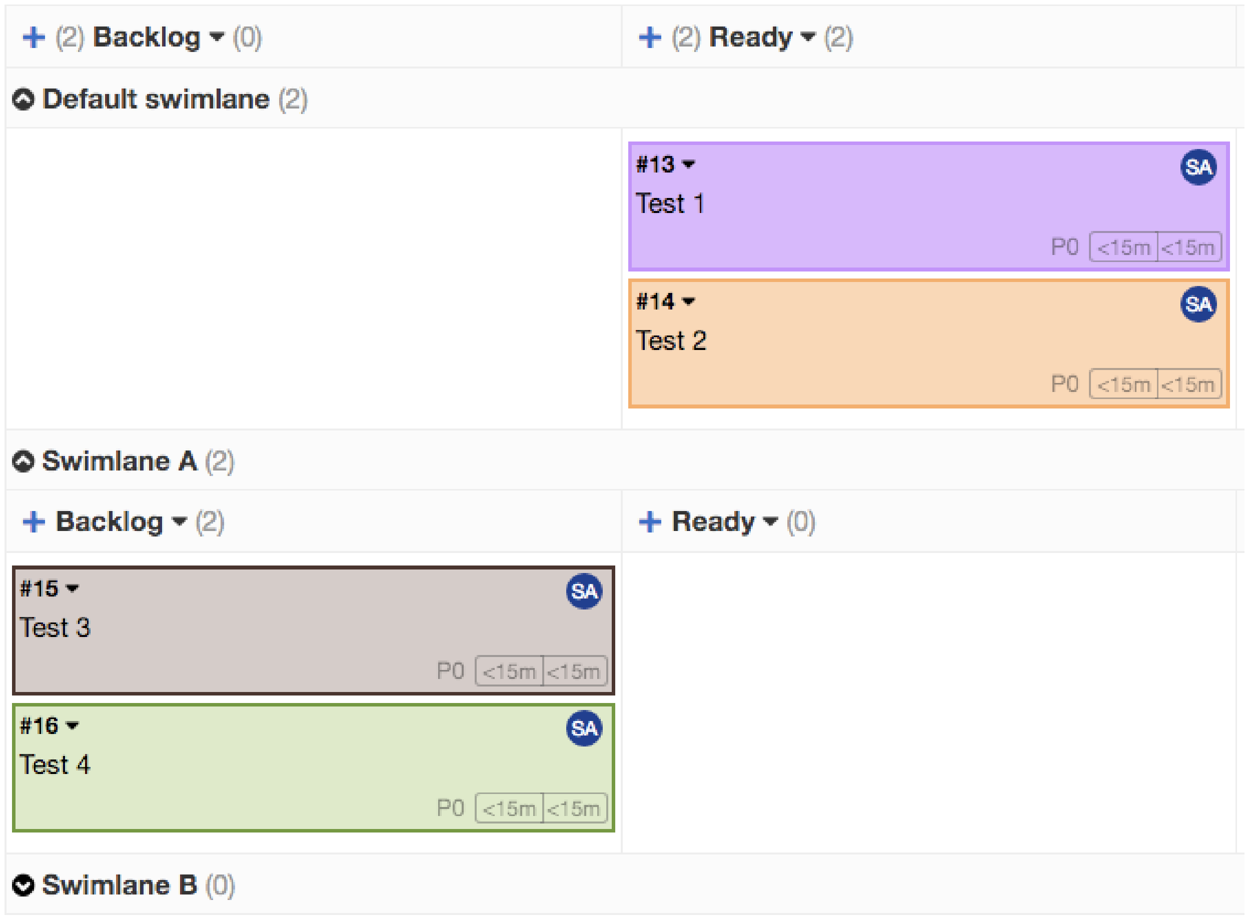
Swimlanes are horizontal separations in your board. For example, it's useful to separate software releases, divide your tasks in different products, teams or whatever you want.

1.7.1 Board with Swimlanes

- You can collapse swimlanes by clicking on the icon on the left
- The default swimlane is always shown at the top



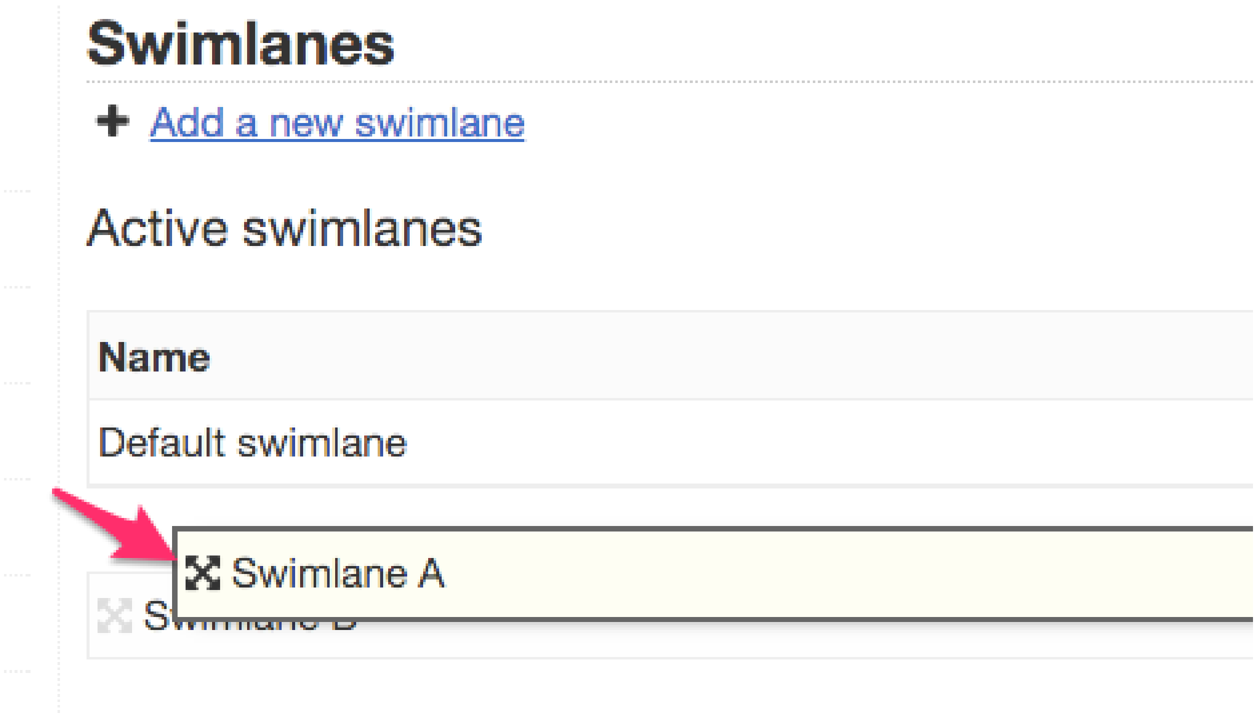
Title
✓ Demo


+ (2) Backlog (0)
+ (2) Ready (2)
Default swimlane (2)
#13
Test 1
SA
P0 <15m <15m
#14
Test 2
SA
P0 <15m <15m
Swimlane A (2)
+ Backlog (2)
+ Ready (0)
#15
Test 3
SA
P0 <15m <15m
#16
Test 4
SA
P0 <15m <15m
Swimlane B (0)

1.7.2 Managing Swimlanes

- All projects have a default swimlane.
- If there is more than one swimlane, the board will show all swimlanes.
- You can drag and drop tasks between swimlanes.

To configure swimlanes go to the **project configuration page** and choose the section **Swimlanes**.



From there, you can add a new swimlane or rename the default one. You can also disable and change the position of the different swimlanes.

- The default swimlane is always on the top but you can hide it.
- Inactive swimlanes are not shown on the board.
- **Removing a swimlane doesn’t remove tasks assigned to it**, those tasks will be moved to the default swimlane.

1.8 Automatic Actions

To minimize user interaction, Kanboard support automated actions.

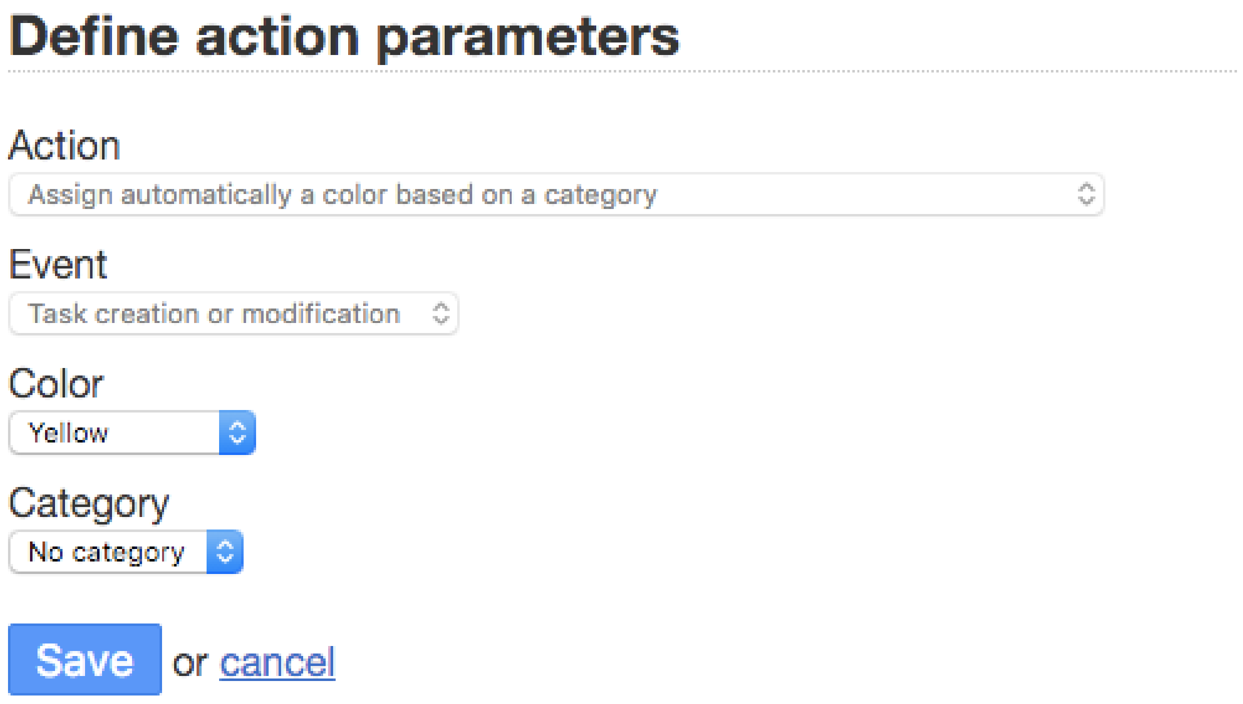
Each automatic action is defined with these properties:

- An event to listen
- Action linked to the event
- Additional parameters

Each project has a different set of automatic actions. The configuration panel is located on the project listing page - just click on the link **Automatic actions**.

1.8.1 Add a new action

Click on the link **Add a new automatic action**.


Define action parameters

Action
Assign automatically a color based on a category

Event
Task creation or modification

Color
Yellow

Category
No category

1. Choose an action
2. Select an event
3. Define the parameters

1.8.2 Available actions

- Create a comment from an external provider
- Add a comment log when moving the task between columns
- Automatically assign a category based on a color
- Change the category based on an external label
- Automatically assign a category based on a link
- Automatically assign a color based on a category
- Assign a color when the task is moved to a specific column
- Change task color when using a specific task link
- Assign a color to a specific user
- Assign the task to the person who does the action
- Assign the task to the person who does the action when the column is changed
- Assign the task to a specific user

- Change the assignee based on an external username
- Close the task
- Close a task in a specific column
- Create a task from an external provider
- Duplicate the task to another project
- Send a task by email to someone
- Move the task to another project
- Move the task to another column when assigned to a user
- Move the task to another column when the category is changed
- Move the task to another column when assignee is cleared
- Open a task
- Automatically update the start date

1.8.3 Examples

Here are some examples used in real life:

When I move a task to the column "Done", automatically close this task

- Choose action: **Close a task in a specific column**
- Choose the event: **Move a task to another column**
- Define action parameter: **Column = Done** (this is the destination column)

When I move a task to the column "To be validated", assign this task to a specific user

- Choose the action: **Assign the task to a specific user**
- Choose the event: **Move a task to another column**
- Define the action parameters: **Column = To be validated** and **User = Bob** (Bob is our tester)

When I move a task to the column "Work in progress", assign this task to the current user

- Choose action: **Assign the task to the person who does the action when the column is changed**
- Choose the event: **Move a task to another column**
- Define action parameter: **Column = Work in progress**

When a task is completed, duplicate this task to another project

Let's say we have two projects: "Customer orders" and "Production". Once the order is validated, swap it to the "Production" project.

- Choose action: **Duplicate the task to another project**
- Choose the event: **Closing a task**

- Define action parameters: **Column = Validated** and **Project = Production**

When a task is moved to the last column, move the exact same task to another project

Let’s say we have two projects: “Ideas” and “Development”. Once the idea is validated, swap it to the “Development” project.

- Choose action: **Move the task to another project**
- Choose the event: **Move a task to another column**
- Define action parameters: **Column = Validated** and **Project = Development**

I want to assign automatically a color to the user Bob

- Choose action: **Assign a color to a specific user**
- Choose the event: **Task assignee change**
- Define action parameters: **Color = Green** and **Assignee = Bob**

I want to assign a color automatically to the defined category “Feature Request”

- Choose action: **Assign automatically a color based on a category**
- Choose the event: **Task creation or modification**
- Define action parameters: **Color = Blue** and **Category = Feature Request**

I want to set the start date automatically when the task is moved to the column “Work in progress”

- Choose action: **Automatically update the start date**
- Choose the event: **Move a task to another column**
- Define action parameters: **Column = Work in progress**

1.9 Project Analytics

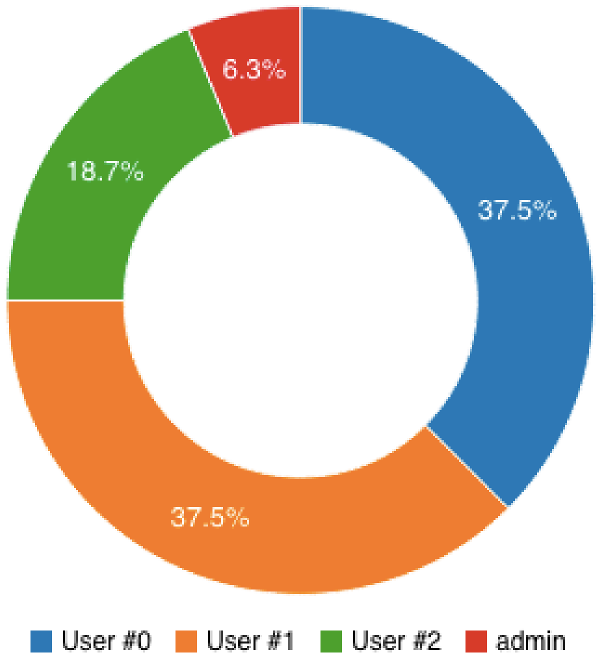
Each project have an analytics section. Depending on how you are using Kanboard, you can see those reports:

1.9.1 User Repartition

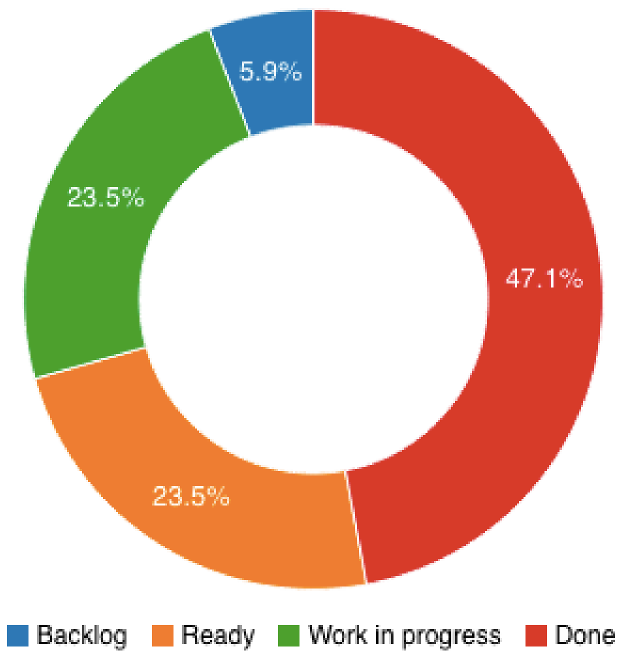
This pie chart show the number of open tasks assigned per user.

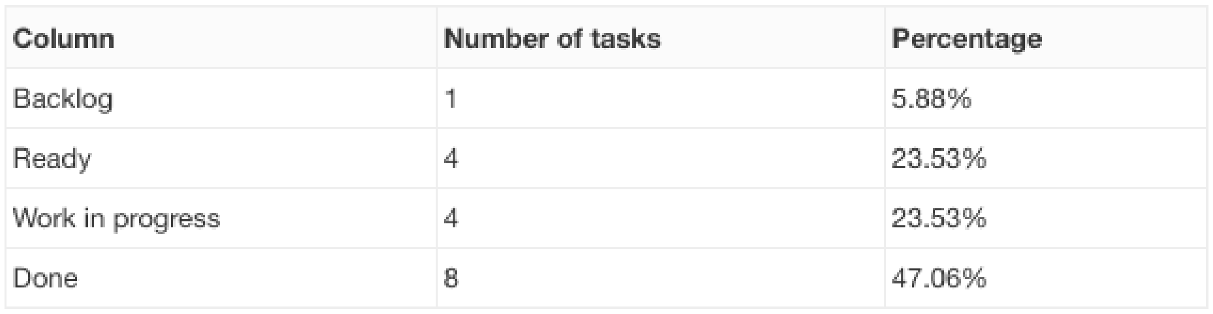
1.9.2 Task Distribution

This pie chart gives an overview of the number of open tasks per column.

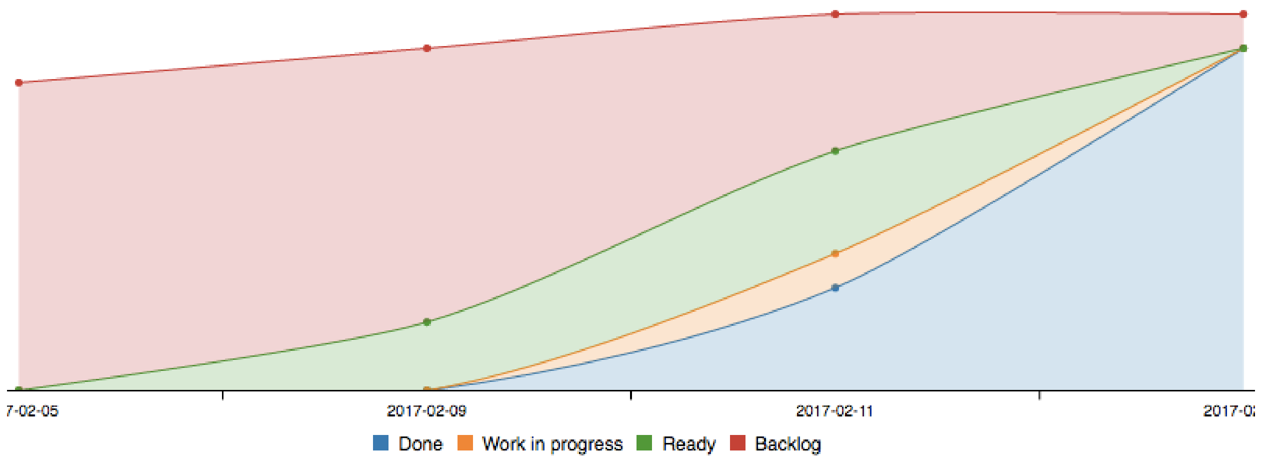


User	Number of tasks	Percentage
User #0	6	37.50%
User #1	6	37.50%
User #2	3	18.75%
admin	1	6.25%





Column	Number of tasks	Percentage
Backlog	1	5.88%
Ready	4	23.53%
Work in progress	4	23.53%
Done	8	47.06%

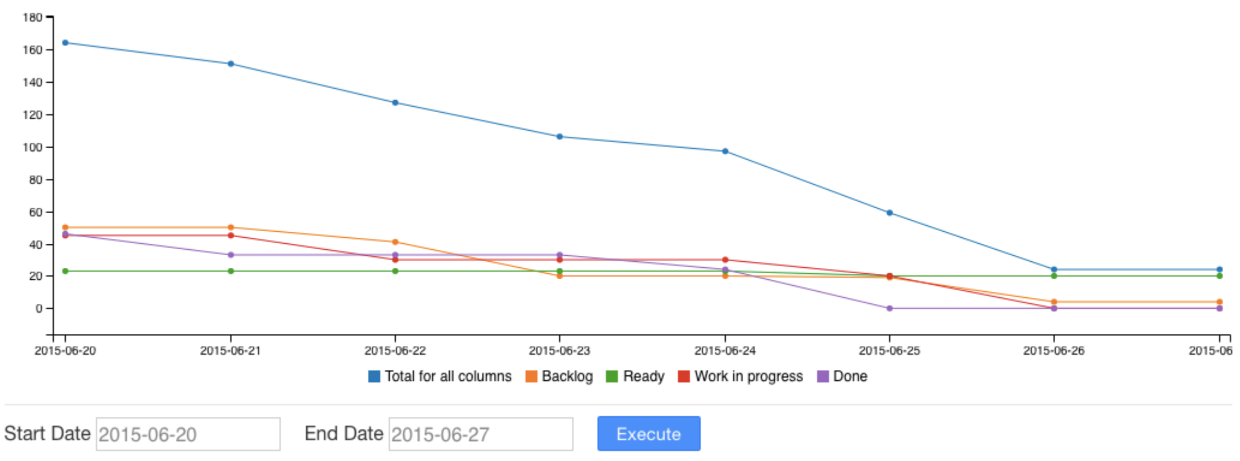


1.9.3 Cumulative Flow Diagram

- This chart shows the number of tasks cumulatively for each column over the time.
- The legend order is the same as the stack in the chart.
- The color of each column is determined automatically.
- Every day, the number of tasks is recorded for each column.
- If you would like to exclude closed tasks, change the global project settings.

Note: You need to have at least two days of data to see the graph.

1.9.4 Burn Down Chart



Start Date 2015-06-20 End Date 2015-06-27

The burn down chart is available for each project.

- This chart is a graphical representation of work left to do versus time.
- Kanboard use the complexity or story point to generate this diagram.
- Everyday, the sum of the story points for each column is calculated.

1.9.5 Average Time Spent Into Each Column

This chart shows the average time spent into each column for the last 1000 tasks.

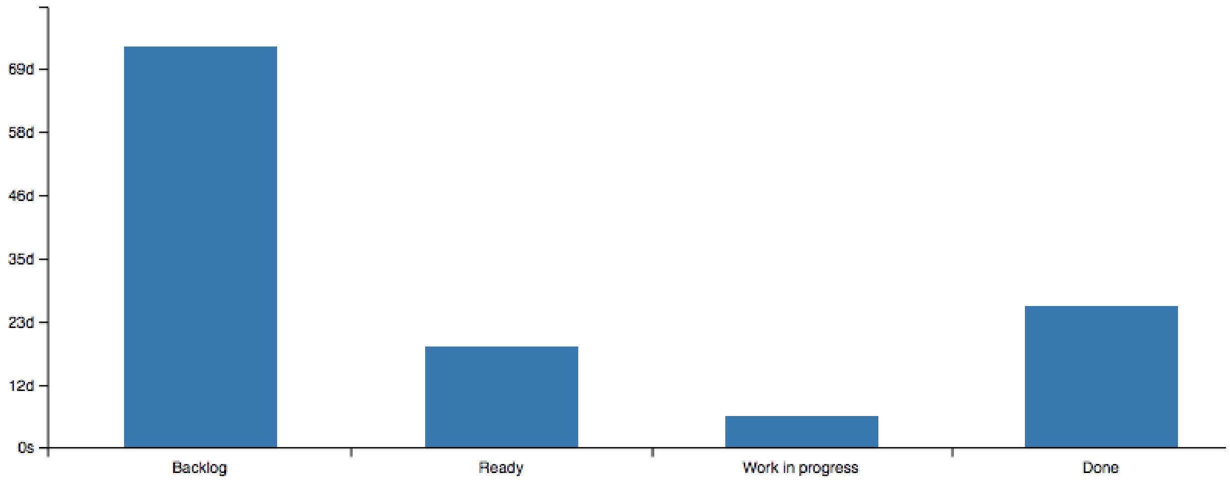
- Kanboard uses the task transitions to calculate the data.
- The time spent is calculated until the task is closed.

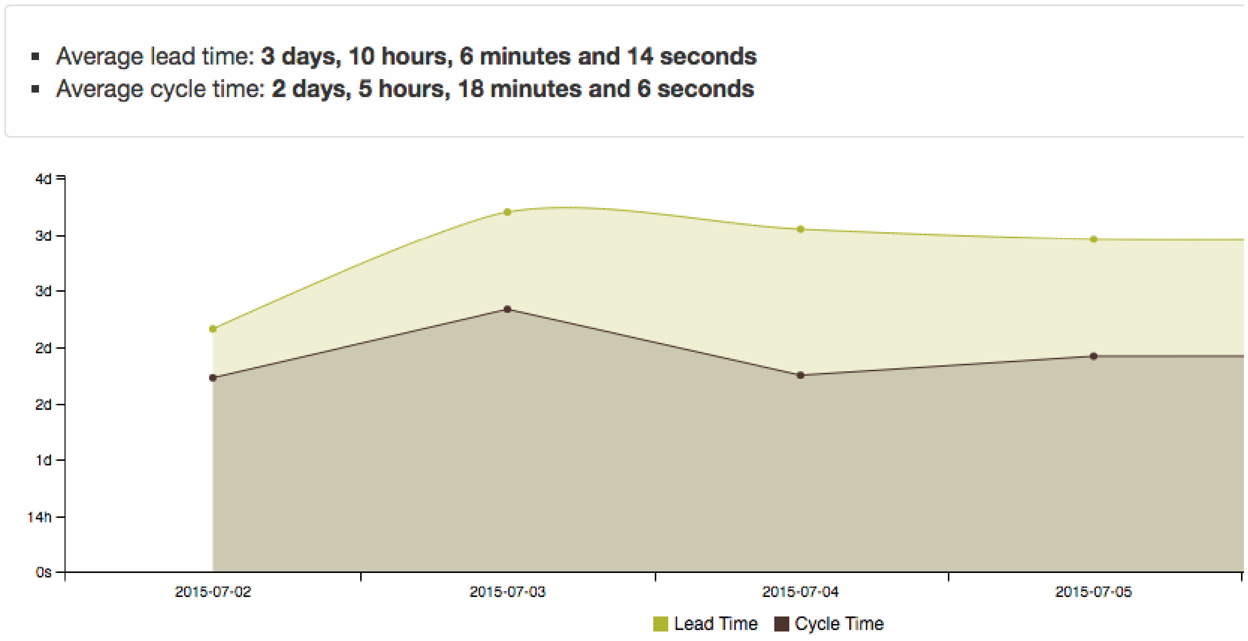
1.9.6 Average Lead and Cycle time

This chart show the average lead and cycle time for the last 1000 tasks over time.

- The lead time is the time between the task creation and the date of completion.
- The cycle time is time between the specified start date of the task to the completion date.
- If the task is not closed, the current time is used instead of the date of completion.

Those metrics are calculated and recorded every day for the whole project.



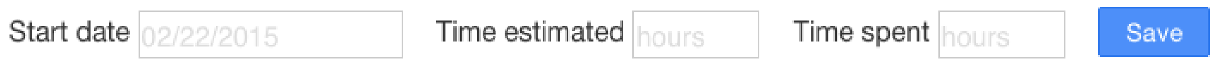


- Average lead time: **3 days, 10 hours, 6 minutes and 14 seconds**
- Average cycle time: **2 days, 5 hours, 18 minutes and 6 seconds**

1.10 Time Tracking

Time tracking information can be defined at the task level or at the subtask level.

1.10.1 Task Time Tracking



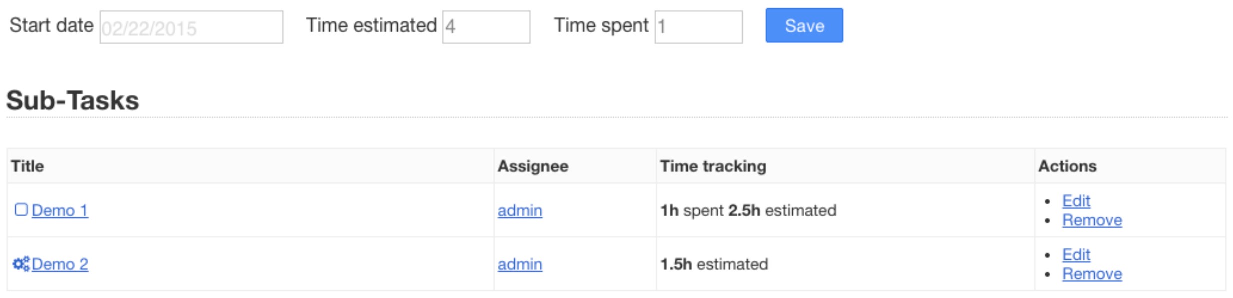
Start date	02/22/2015	Time estimated	hours	Time spent	hours	Save

Tasks have two fields:

- Time estimated
- Time spent

These values represent hours of work and have to be set manually.

1.10.2 Subtask Time Tracking



Start date	02/22/2015	Time estimated	4	Time spent	1	Save

Sub-Tasks

Title	Assignee	Time tracking	Actions
Demo 1	admin	**1h** spent **2.5h** estimated	• Edit • Remove
Demo 2	admin	**1.5h** estimated	• Edit • Remove

Subtasks also have the fields “time spent” and “time estimated”.

When you change the value of these fields, **the task time tracking values are updated automatically and becomes the sum of all subtask values**.

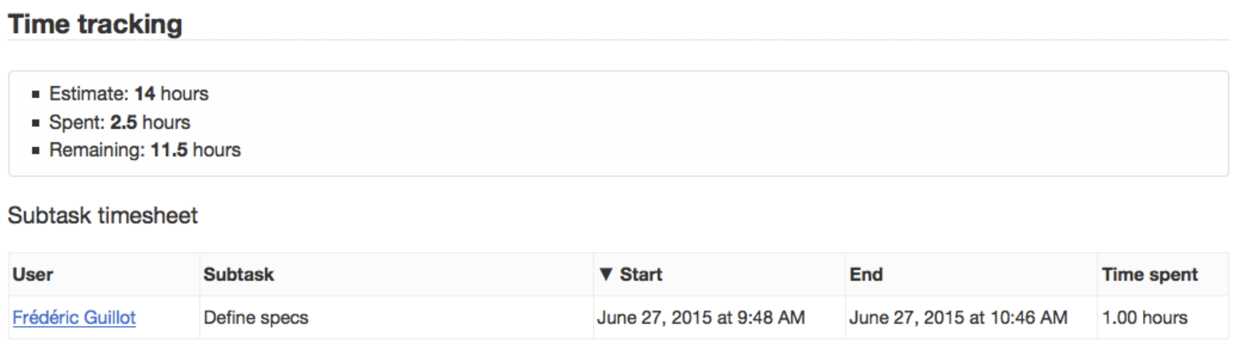
Kanboard records the time between each subtask status change in a separate table.

- Changing subtask status from **todo** to **in progress** logs the start time
- Changing subtask status from **in progress** to **done** logs the end time but also update the time spent of the subtask and the task

The breakdown of all records is visible in the task view page:

For each subtask, the timer can be stopped/started at any time:

- The timer doesn’t depend of the subtask status
- Each time you start the timer a new record is created in the time tracking table
- Each time you stop the clock the end date is recorded in the time tracking table
- The calculated time spent is rounded to the nearest quarter (only for Kanboard < 1.0.32)

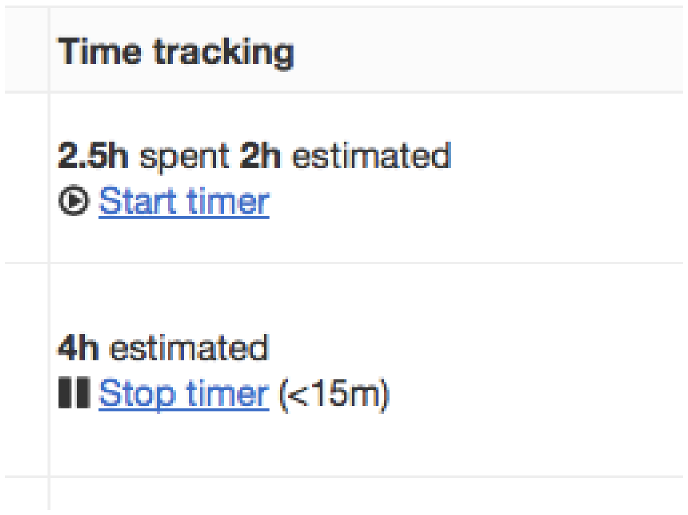


Time tracking

- Estimate: **14** hours
- Spent: **2.5** hours
- Remaining: **11.5** hours

Subtask timesheet

User	Subtask	▼ Start	End	Time spent
Frédéric Guillot	Define specs	June 27, 2015 at 9:48 AM	June 27, 2015 at 10:46 AM	1.00 hours



Time tracking

2.5h spent **2h** estimated
Start timer

4h estimated
Stop timer (<15m)

1.11 Notifications

Kanboard is able to send notifications through several channels:

- Email
- Web (List of unread messages)

External plugins allow you to send notifications to Slack, Hipchat, Jabber or any chat system.

1.11.1 Configuration

Each user must enable the notifications in their profile: **User Profile > Notifications**. It's disabled by default.

To receive email notifications you need a valid email address in your profile and the application must be configured to send emails.

You can choose your favorite notification method:

- Emails
- Web (see below)

For each project you are a member, you can choose to receive notifications for:

- All tasks
- Only for tasks assigned to you
- Only for tasks created by you
- Only for tasks created by you and assigned to you

You can also select only some projects, by default it's all projects where you are a member.

1.11.2 Web notifications

Web notifications are available from the dashboard or from the icon at the top:

Notifications are shown in a list, so you can mark individual notification as read or everything.

In this way you can still get notified without having to receive emails.

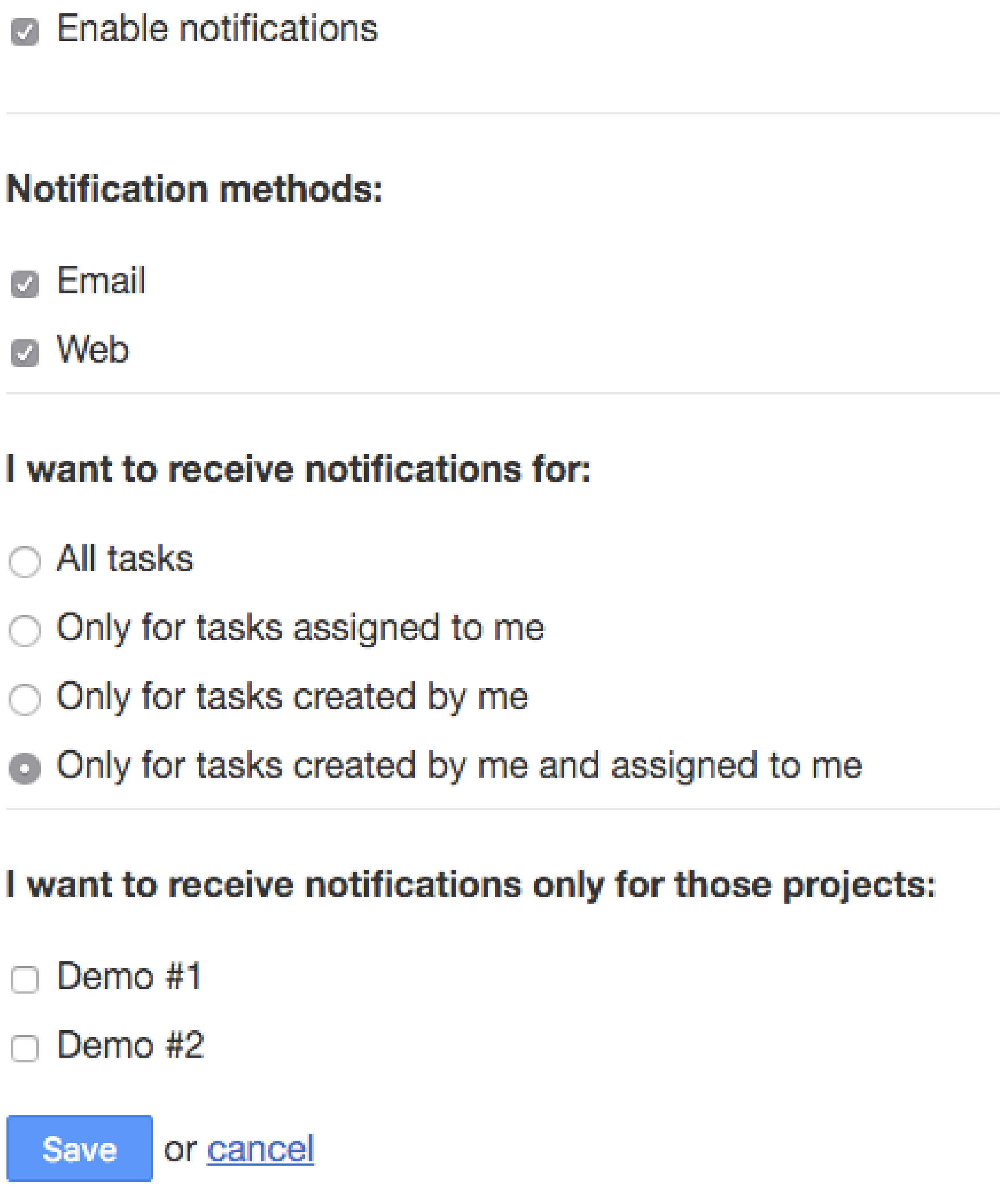
1.11.3 User Mentions

Kanboard offers the possibility to send notifications when someone is mentioned.

If you need to get the attention of someone in a comment or in a task, use the @ symbol followed by their username. Kanboard will automatically suggest a list of users:

- At the moment, only the task description and the comment text area have this feature enabled.
- The user mentions works only during tasks and comments creation.
- To be notified, mentioned users need to be a member of the project.
- When someone is mentioned, this user will receive a notification.
- The @username mention is linked to the public user profile.

The notification is sent according to the user settings, it can be an email, a web notification or even a message on Slack/Hipchat/Jabber if you have installed the right plugins.



[✓] Enable notifications

Notification methods:

[✓] Email

[✓] Web

I want to receive notifications for:

○ All tasks

○ Only for tasks assigned to me

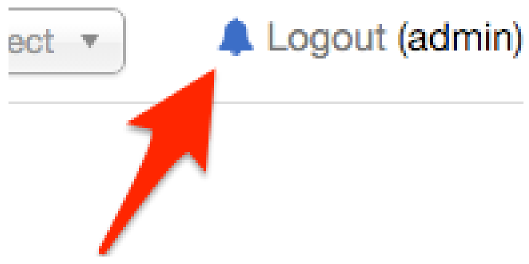
○ Only for tasks created by me

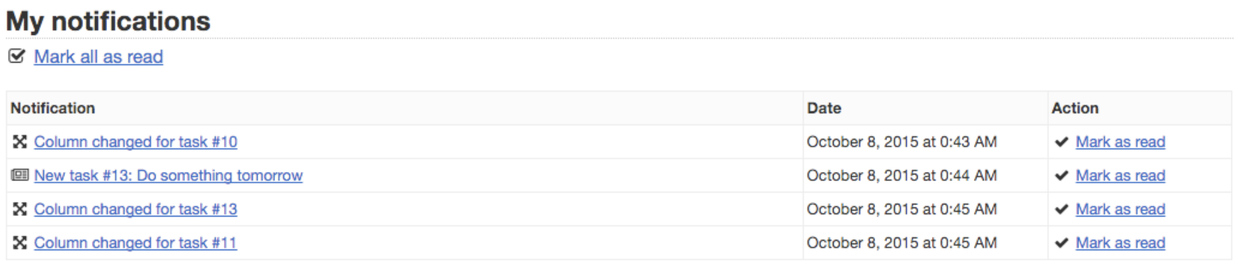
● Only for tasks created by me and assigned to me

I want to receive notifications only for those projects:

☐ Demo #1

☐ Demo #2

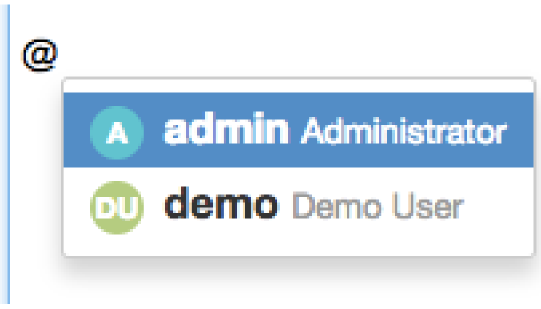




My notifications

Mark all as read

Notification	Date	Action
Column changed for task #10	October 8, 2015 at 0:43 AM	✓ Mark as read
New task #13: Do something tomorrow	October 8, 2015 at 0:44 AM	✓ Mark as read
Column changed for task #13	October 8, 2015 at 0:45 AM	✓ Mark as read
Column changed for task #11	October 8, 2015 at 0:45 AM	✓ Mark as read



1.12 RSS/Atom Subscriptions

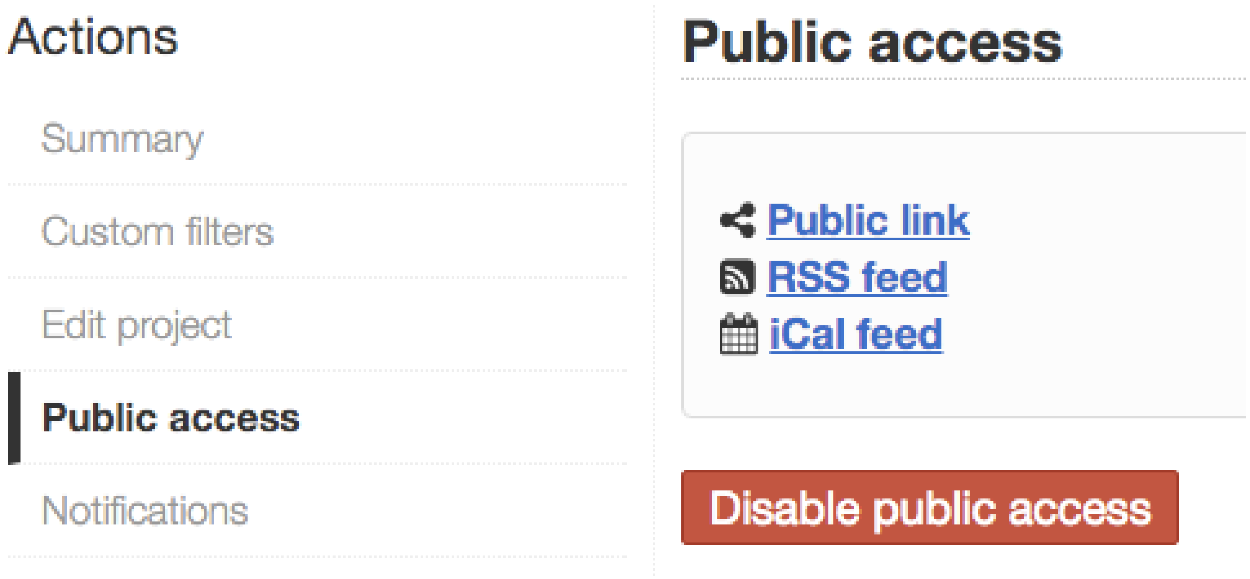
Kanboard supports RSS feeds for projects and users.

- Project feeds contains only the activity the project
- User feeds contains the activity stream of all projects the user is a member

Those subscriptions are only activated when the public access is enabled in the user profile or in the project settings.

1.12.1 Enable/Disable Project RSS Feeds

Go to **Project settings > Public access**.



1.12.2 Enable/Disable User RSS Feeds

Go to **User profile > Public access**.

The RSS link is protected by a random token, only people who know the URL can access to the feed.

1.13 iCalendar Subscriptions

Kanboard supports iCal feeds for projects and users. This feature allows you to import Kanboard tasks in almost any calendar program (Microsoft Outlook, Apple Calendar, Mozilla Thunderbird and Google Calendar).

Calendar subscriptions are **read-only** access, you cannot create tasks from external calendar software. The Calendar feed export follows the iCal standard.

Note: Only tasks within the date range of -2 months to +6 months are exported to the iCalendar feed.

1.13.1 Project Calendars

- Each project has its own calendar.
- The subscription link is unique per project, the link is activated when you enable the public access of your project: **Project settings > Public access**.
- This calendar shows only tasks for the selected project.

1.13.2 User Calendars

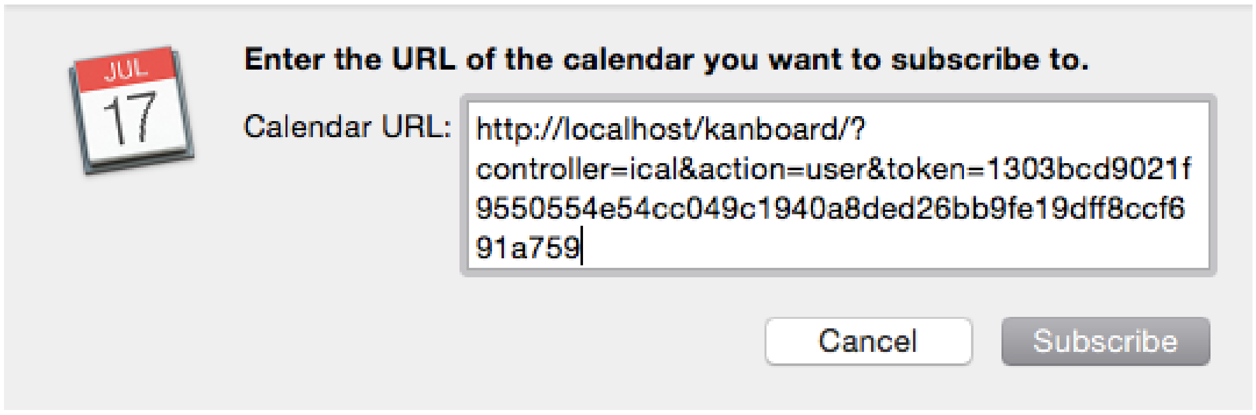
- Each user has its own calendar.
- The subscription link is unique per user, the link is activated when you enable the public access of your user: **User profile > Public access**.
- This calendar show tasks assigned to the user for all projects.

1.13.3 Adding your Kanboard Calendar to Apple Calendar

- Open Calendar
- Select **File > New Calendar Subscription**
- Copy and paste the iCal feed URL from Kanboard
- You can choose to synchronize the calendar with iCloud to be available across all your devices
- Don’t forget to select the refresh frequency

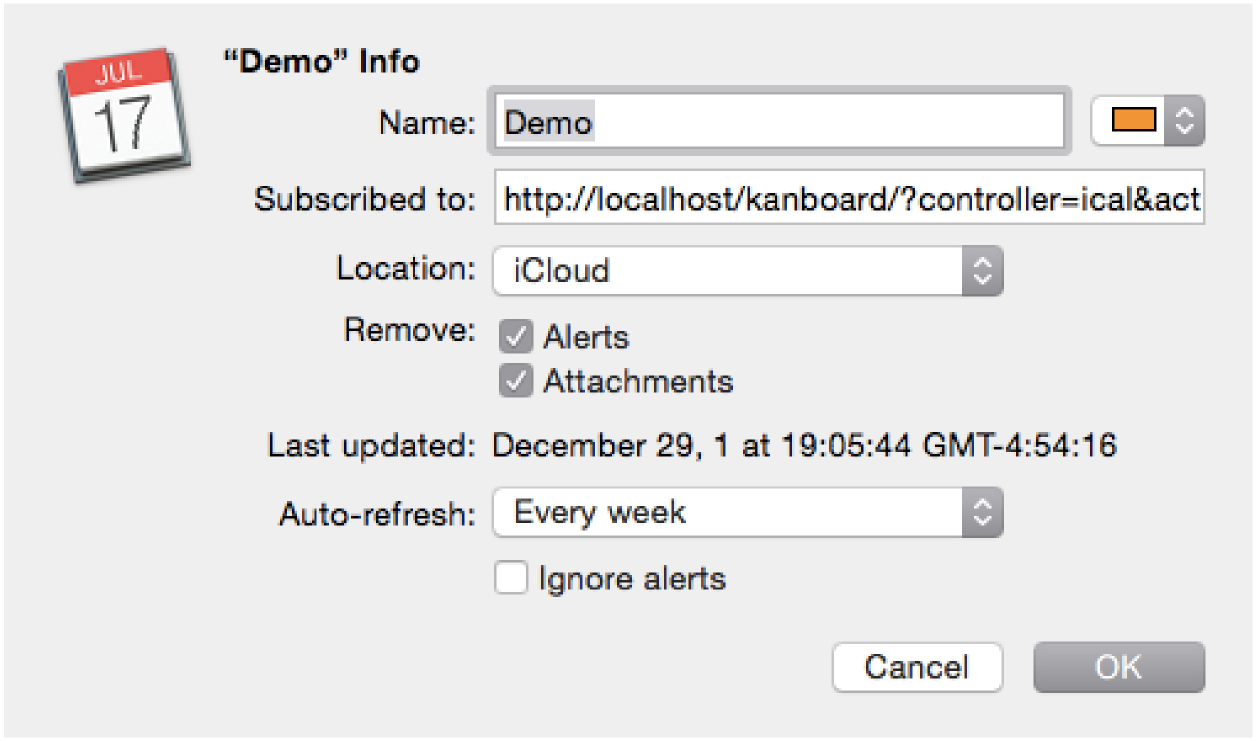
1.13.4 Adding your Kanboard Calendar to Mozilla Thunderbird

- Install the Add-on **Lightning** to add calendar support to Thunderbird
- Click on **File > New Calendar**
- In the dialog box, choose **On the Network**
- Choose the format iCalendar



Enter the URL of the calendar you want to subscribe to.

Calendar URL: http://localhost/kanboard/?controller=ical&action=user&token=1303bcd9021f9550554e54cc049c1940a8ded26bb9fe19dff8ccf691a759



"Demo" Info

Name: Demo

Subscribed to: http://localhost/kanboard/?controller=ical&act

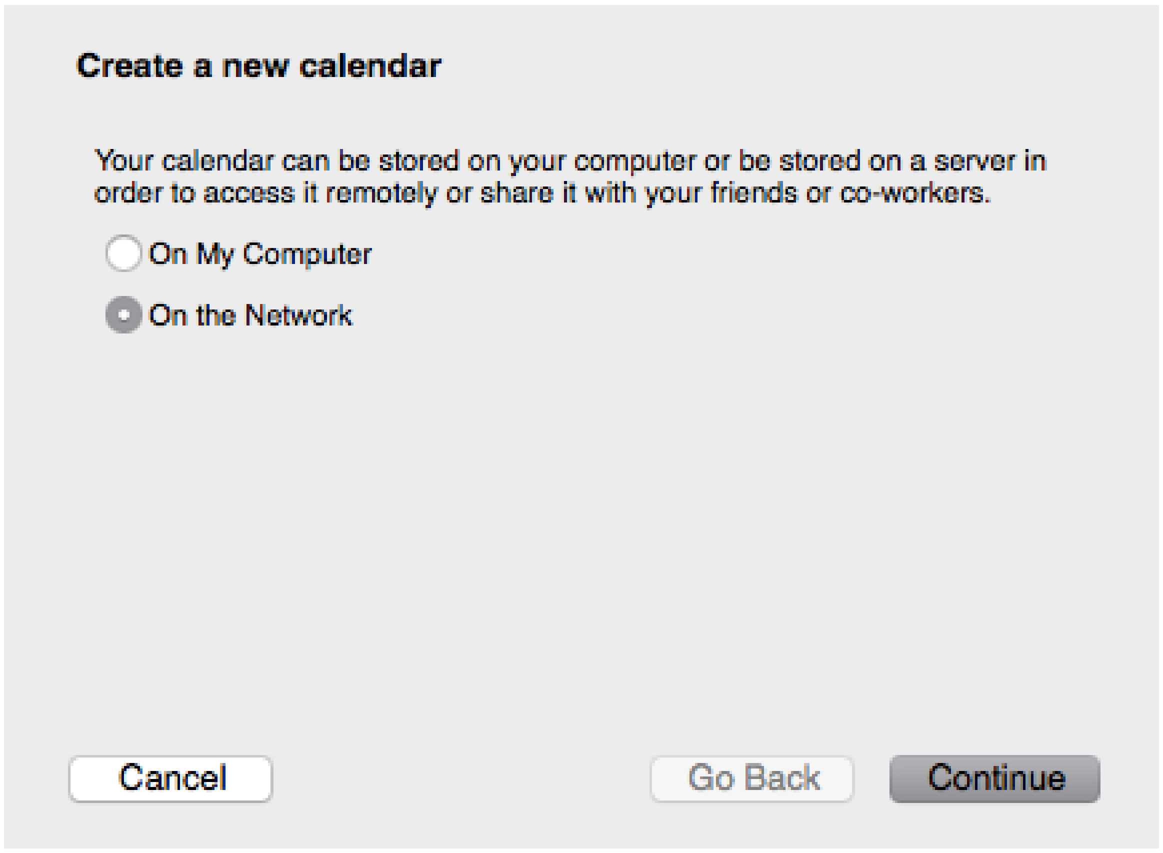
Location: iCloud

Remove: ✓ Alerts ✓ Attachments

Last updated: December 29, 1 at 19:05:44 GMT-4:54:16

Auto-refresh: Every week

☐ Ignore alerts



Create a new calendar

Your calendar can be stored on your computer or be stored on a server in order to access it remotely or share it with your friends or co-workers.

[ ] On My Computer [✓ On the Network]

- Copy and paste the iCal feed URL from Kanboard
- Choose the colors and other settings and finally save

1.13.5 Adding your Kanboard calendar to Google Calendar

- Click the down-arrow next to **Other calendars**.
- Select **Add by URL** from the menu.
- Copy and paste the iCal feed URL from Kanboard

Your Kanboard calendar can also be available from your Android device if you enable the synchronization.

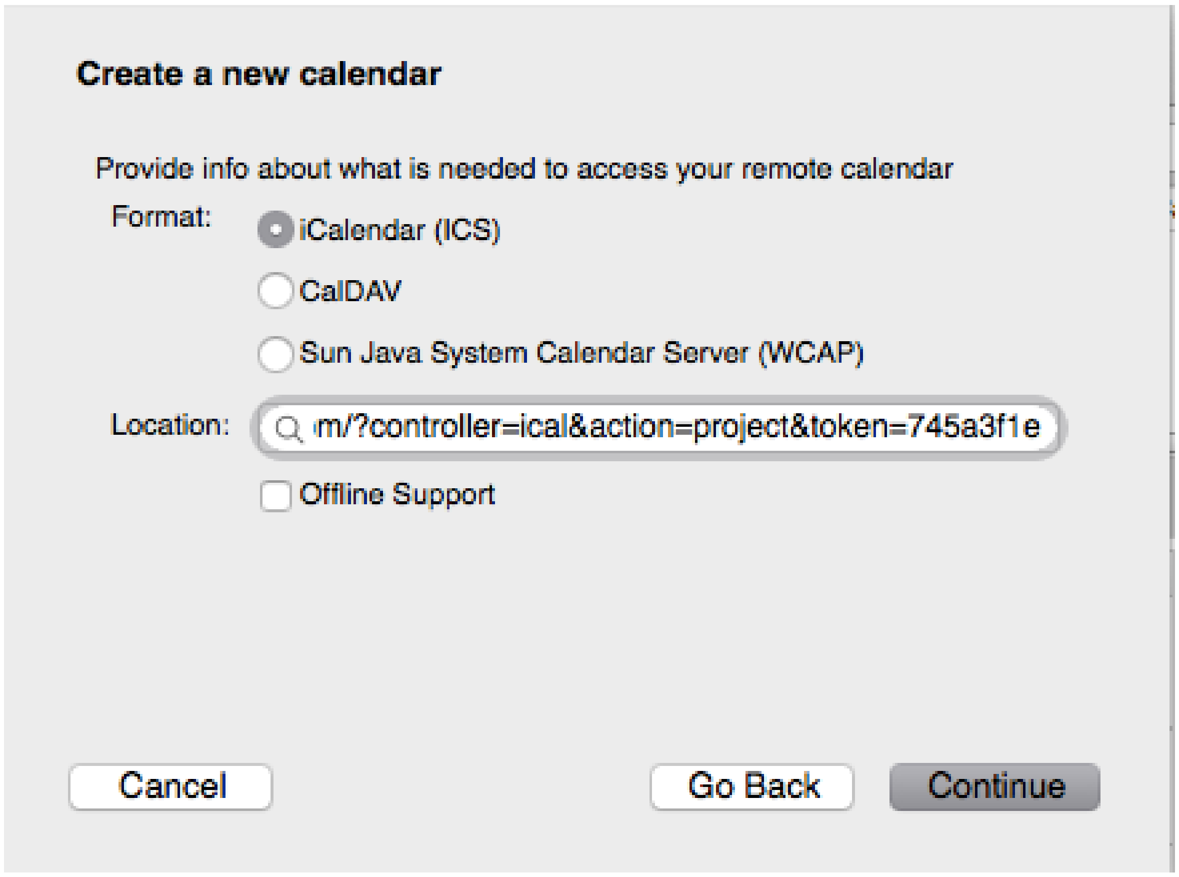
Note: According to the Google Support, external calendars are not refreshed very often, read the documentation.

1.14 Settings

Some parameters for the application can be changed on the settings page. Only administrators can change those settings.

1.14.1 Application Settings

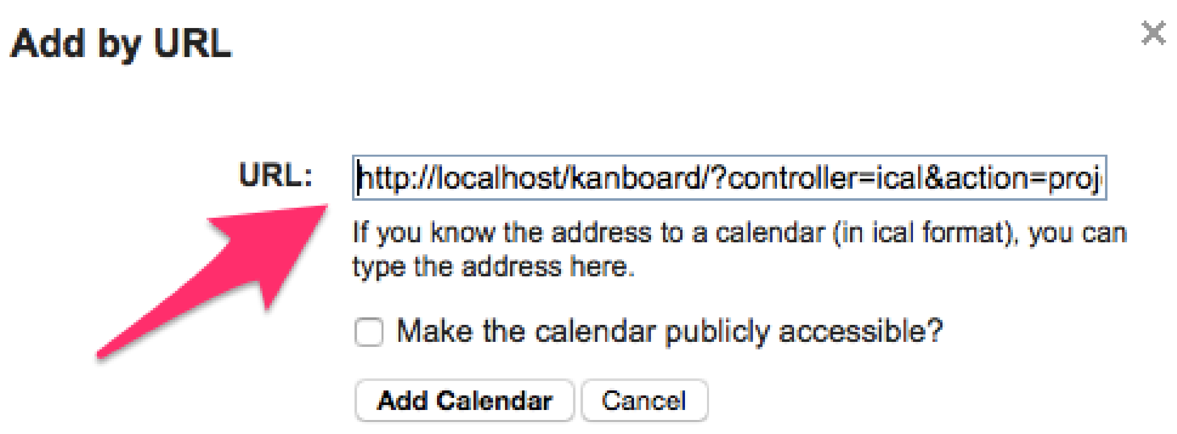
Go to the menu **Settings**, then choose **Application settings** on the left.


Create a new calendar
Provide info about what is needed to access your remote calendar
Cancel
Go Back
Continue

Format: [✓ iCalendar (ICS)] CalDAV Sun Java System Calendar Server (WCAP)

Location: m/?controller=ical&action=project&token=745a3f1e

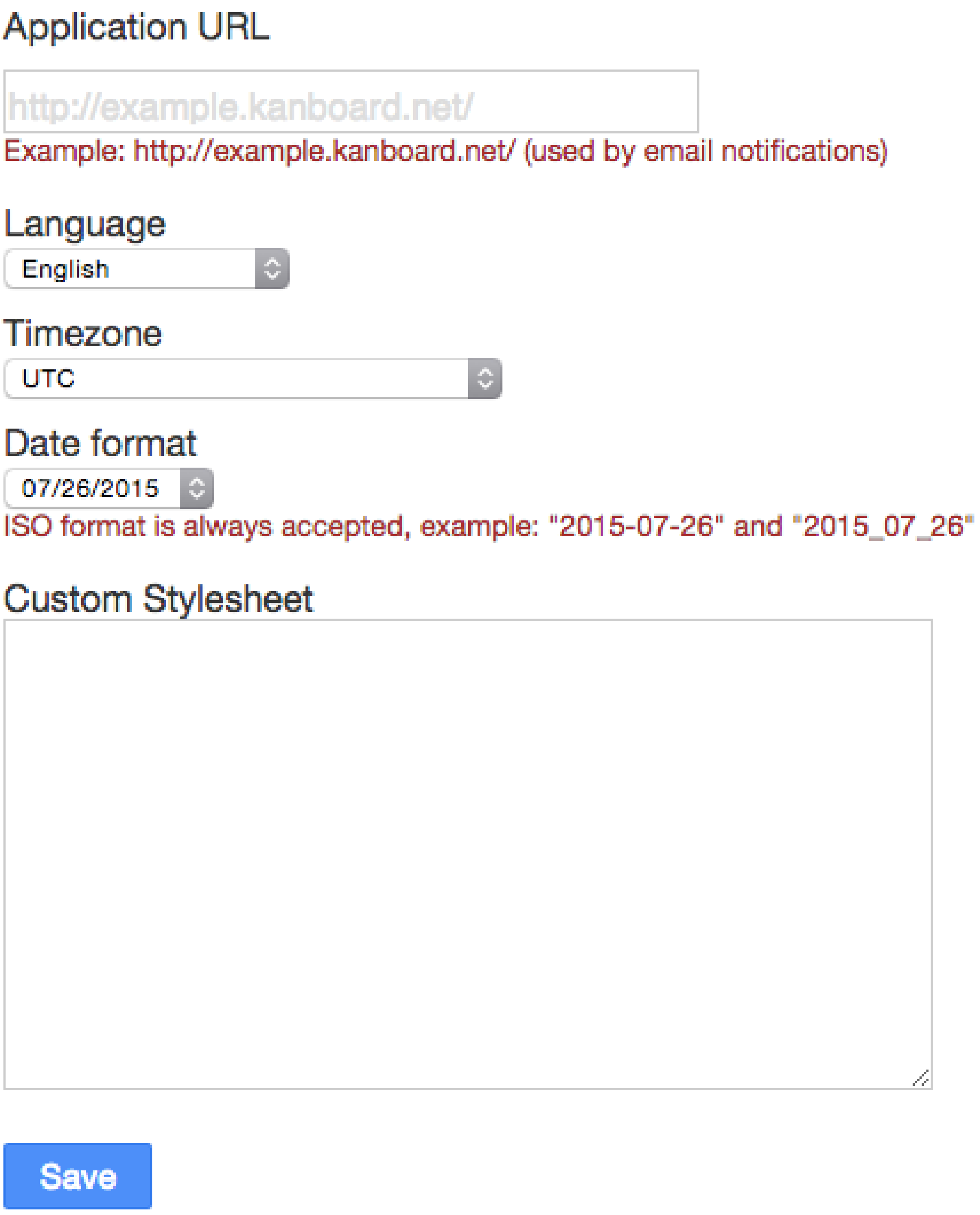
Offline Support ☐



URL: http://localhost/kanboard/?controller=ical&action=proj

If you know the address to a calendar (in ical format), you can type the address here.

☐ Make the calendar publicly accessible?



Application URL

http://example.kanboard.net/

Example: http://example.kanboard.net/ (used by email notifications)

Language

English

Timezone

UTC

Date format

07/26/2015

ISO format is always accepted, example: "2015-07-26" and "2015_07_26"

Custom Stylesheet

Application URL

This parameter is used for email notifications. The email footer will contain a link to the Kanboard task.

Language

The application language can be changed at anytime. The language will be set for all users.

Time zone

By default, Kanboard use UTC as time zone, but you can define your own time zone. The list contains all supported time zones by your web server.

Date format

Input format used for date fields, for examples the due date for tasks.

Kanboard offers 4 different formats:

- DD/MM/YYYY
- MM/DD/YYYY (default)
- YYYY/MM/DD
- MM.DD.YYYY

The ISO 8601 format is always accepted (YYYY-MM-DD or YYYY_MM_DD).

Custom Stylesheet

Write your own CSS to override or improve Kanboard default style.

Here an example to change color of category labels:

For the category container:

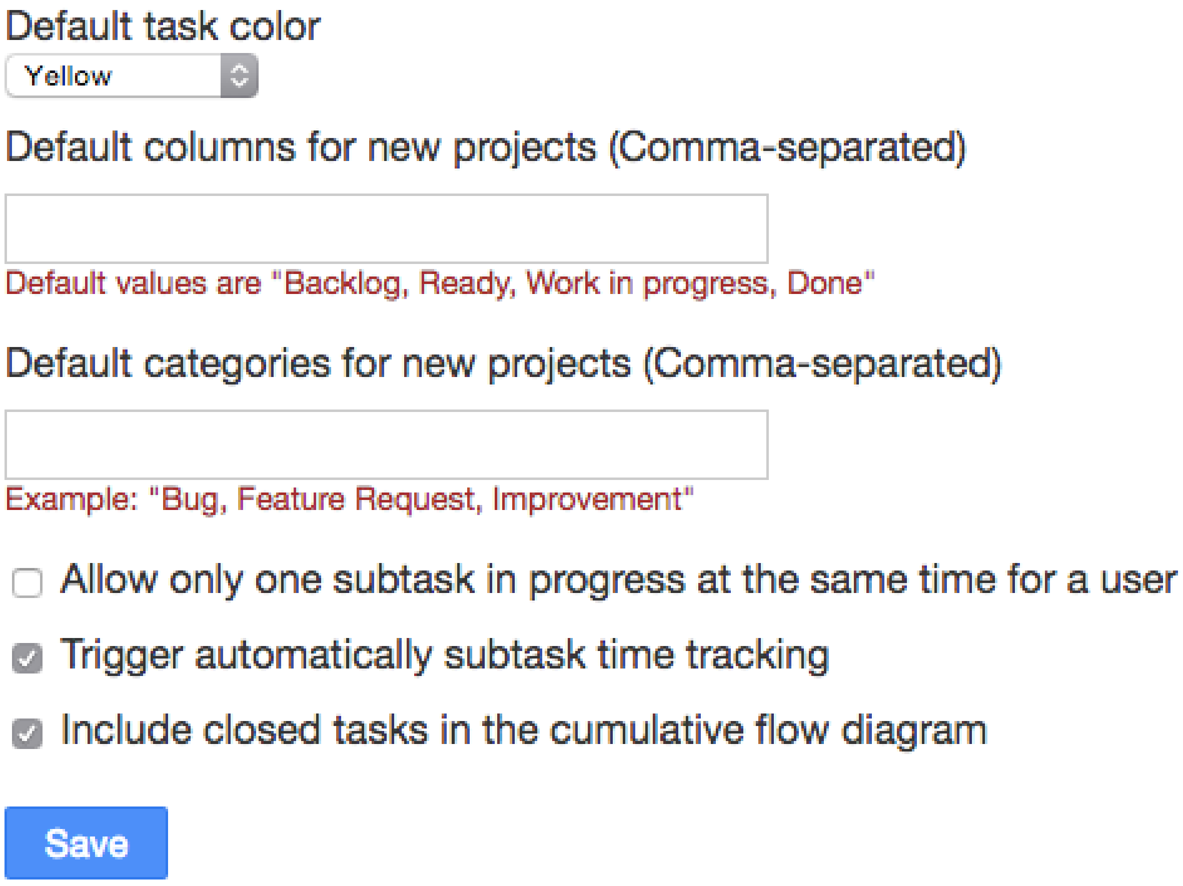
```css
.task-board-category-container-color span {
  border: solid 0.5px grey;
  color: black;
}
```

Custom css values for one category - this is an example for displaying the text:

```css
[class*="category-MyLabel"] {
  background-color: rgba(255, 0, 0, 0.50);
  border: none!important;
  font-weight: bold;
  font-style: italic;
  box-shadow: 0 1px 1px rgba(186, 186, 186, 0.55);
  color: white!important;
  font-size:11px;
}
```

1.14.2 Project Settings

Go to the menu **Settings**, then choose **Project settings** on the left.


Default task color

Yellow

Default columns for new projects (Comma-separated)

Default values are "Backlog, Ready, Work in progress, Done"

Default categories for new projects (Comma-separated)

Example: "Bug, Feature Request, Improvement"

- [ ] Allow only one subtask in progress at the same time for a user
- [x] Trigger automatically subtask time tracking
- [x] Include closed tasks in the cumulative flow diagram

Default columns for new projects

You can change the default column names here. It's useful if you always create projects with the same columns.

Each column name must be separated by a comma.

By default, Kanboard use those column names: Backlog, Ready, Work in progress and Done.

Default categories for new projects

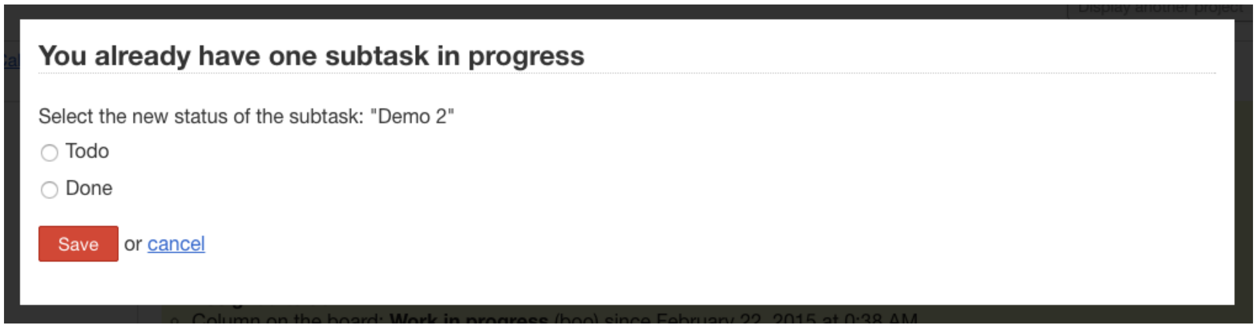
Categories are not global to the application but attached to a project. Each project can have different categories.

However, if you always create the same categories for all your projects, you can define here the list of categories to create automatically.

Allow only one subtask in progress at the same time for a user

When this option is enabled, a user can work with only one subtask at the time.

If another subtask have the status "in progress", the user will see this dialog box:


You already have one subtask in progress

Select the new status of the subtask: "Demo 2"

Trigger automatically subtask time tracking

- If enabled, when a subtask status is changed to “in progress”, the timer will start automatically.
- Disable this option if you don’t use time tracking.

Include closed tasks in the cumulative flow diagram

- If enabled, closed tasks will be included in the cumulative flow diagram.
- If disabled, only open tasks will be included.
- This option affects the column “total” of the table “project_daily_column_stats”

1.14.3 Board Settings

Go to the menu **Settings**, then choose **Board settings** on the left.

Task highlighting

This feature displays a shadow around the task when a task is moved recently.

Set the value 0 to disable this feature, 2 days by default (172800 seconds).

Everything moved since 2 days will have shadow around the task.

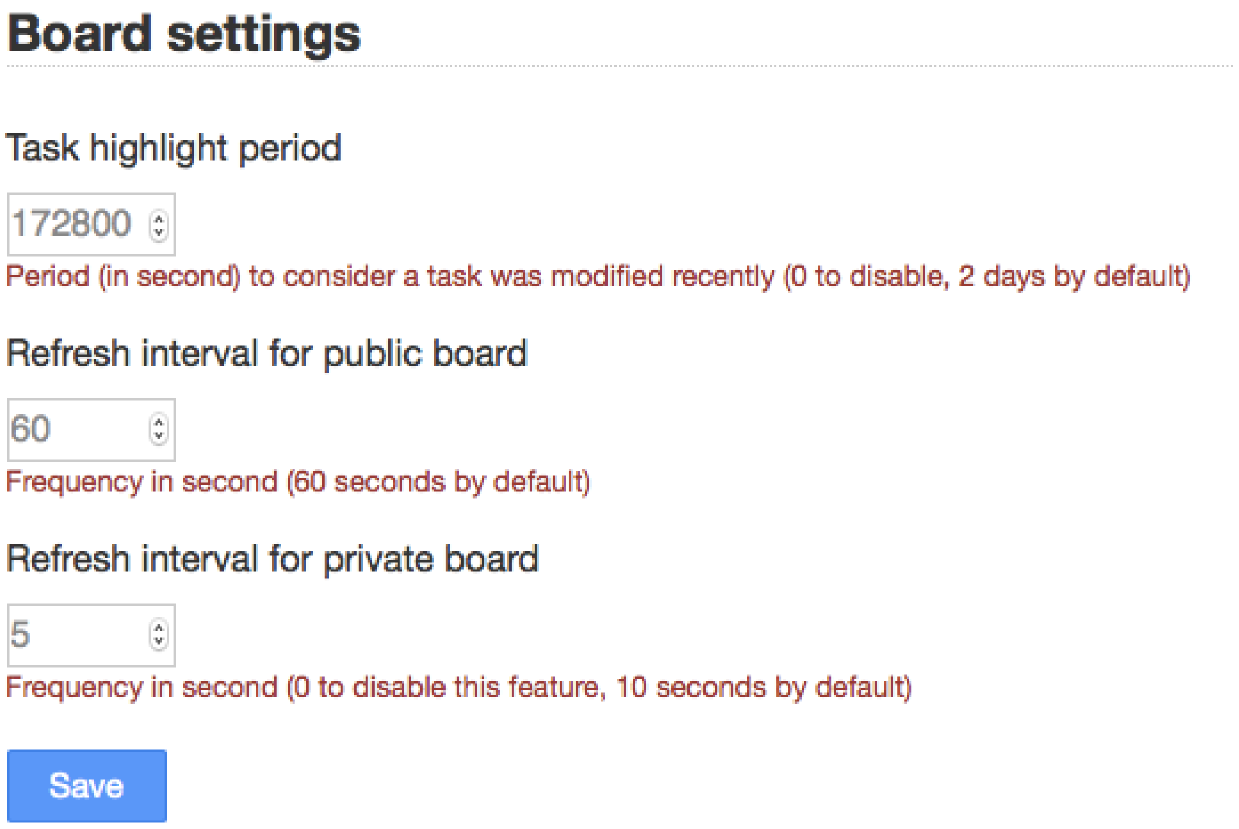
Refresh interval for public board

When you share a board, the page will refresh every 60 seconds automatically by default.

Refresh interval for private board

When your web browser is open on a board, Kanboard checks every 10 seconds if something has been changed by someone else.

Technically this process is done by Ajax polling.



Board settings

Task highlight period

172800

Period (in second) to consider a task was modified recently (0 to disable, 2 days by default)

Refresh interval for public board

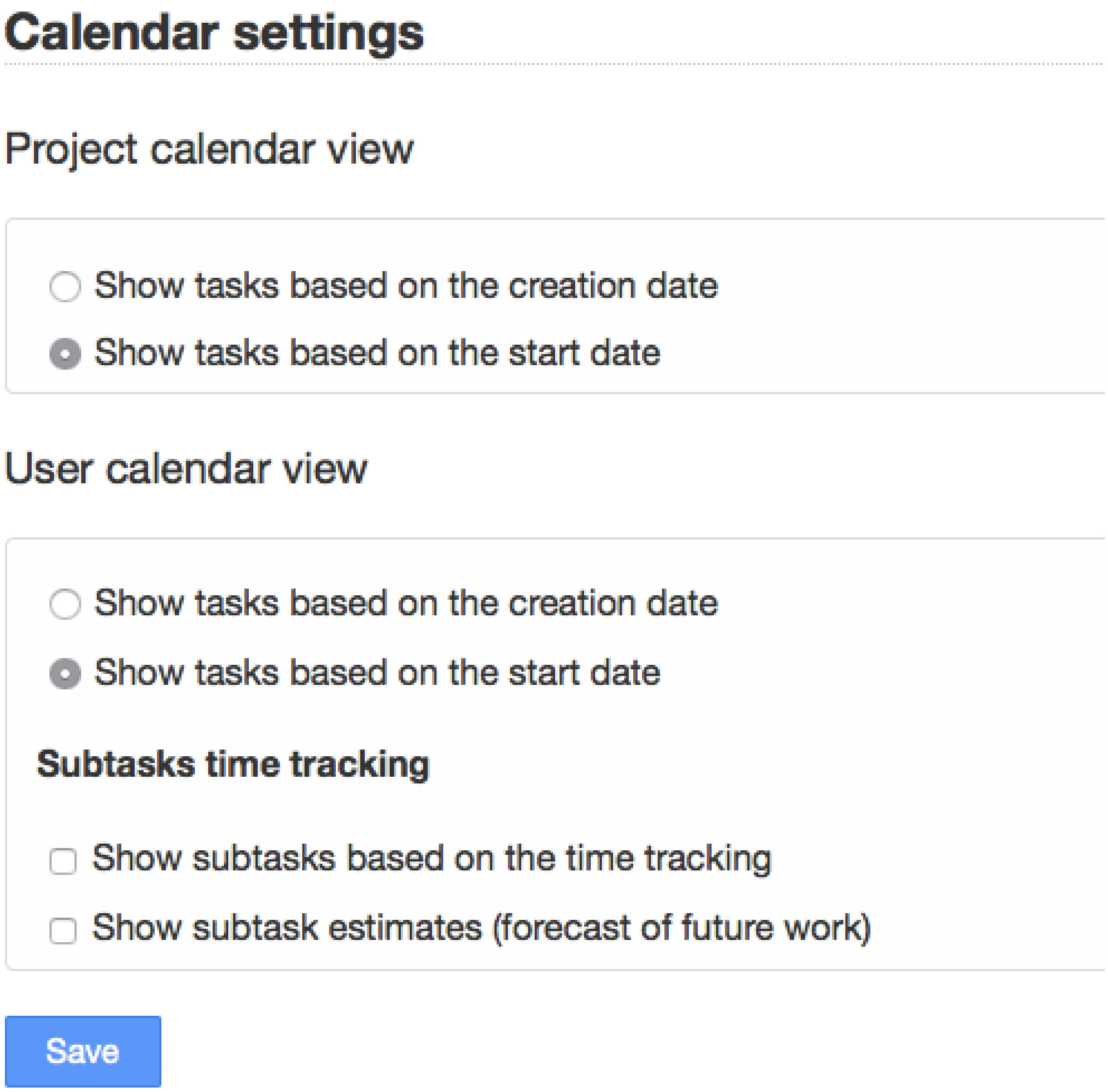
60

Frequency in second (60 seconds by default)

Refresh interval for private board

5

Frequency in second (0 to disable this feature, 10 seconds by default)



Calendar settings

Project calendar view

- ◯ Show tasks based on the creation date
- ● Show tasks based on the start date

User calendar view

- ◯ Show tasks based on the creation date
- ● Show tasks based on the start date

Subtasks time tracking

- [ ] Show subtasks based on the time tracking
- [ ] Show subtask estimates (forecast of future work)

1.14.4 Calendar Settings

Go to the menu **Settings**, then choose **Calendar settings** on the left.

There are two different calendars in Kanboard:

- Project calendar
- User calendar (available from the dashboard)

Project Calendar

This calendar shows tasks with defined due date and tasks based on the creation date or the start date.

Show tasks based on the creation date:

- The start date of the calendar event is the creation date of the task.
- The end date of the event is the date of completion.

Show tasks based on the start date:

- The start date of the calendar event is the start date of the task.
- This date can be defined manually.
- The end date of the event is the date of completion.
- If there is no start date the task will not appear on the calendar.

User Calendar

This calendar shows only tasks assigned to the user and optionally sub-tasks information.

Show sub-tasks based on the time tracking:

- Display sub-tasks in the calendar from the information recorded in the time tracking table.
- The intersection with the user timetable is also calculated.

Show sub-task estimates (forecast of future work):

- Display the estimate of future work for sub-tasks in status “todo” and with a defined “estimate” value.

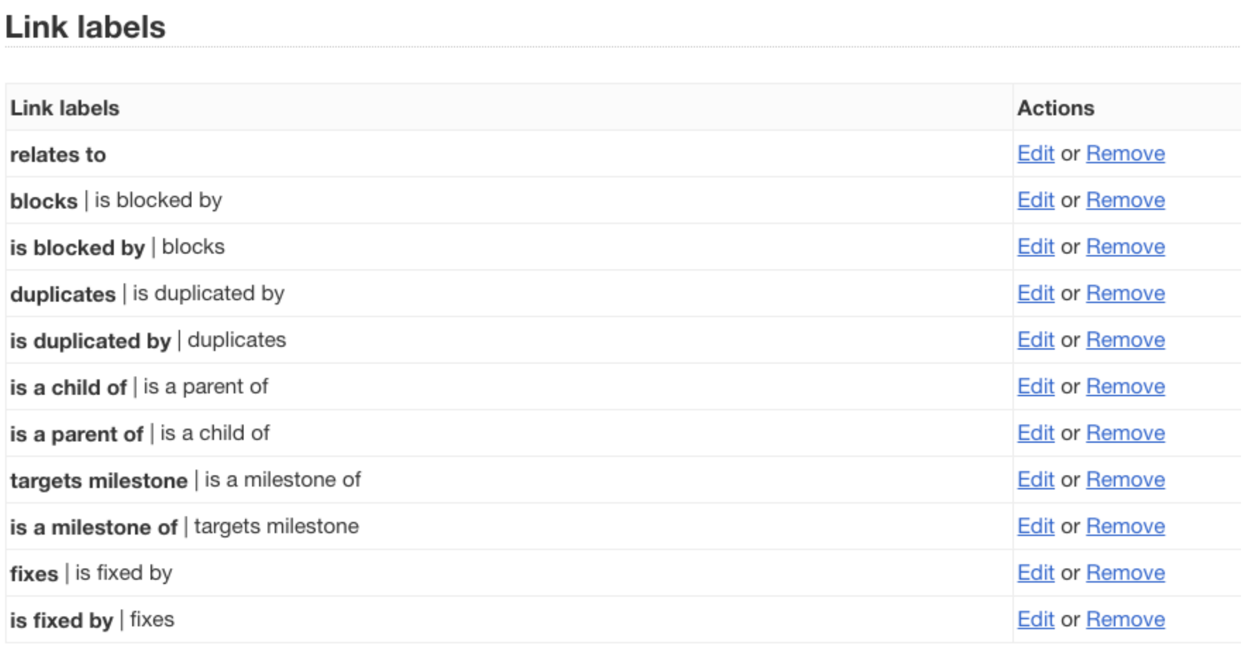
1.14.5 Link Settings

Task relations can be changed from the application settings (**Settings > Link settings**)

Each label may have an opposite label defined. If there is no opposite, the label is considered bidirectionnal.

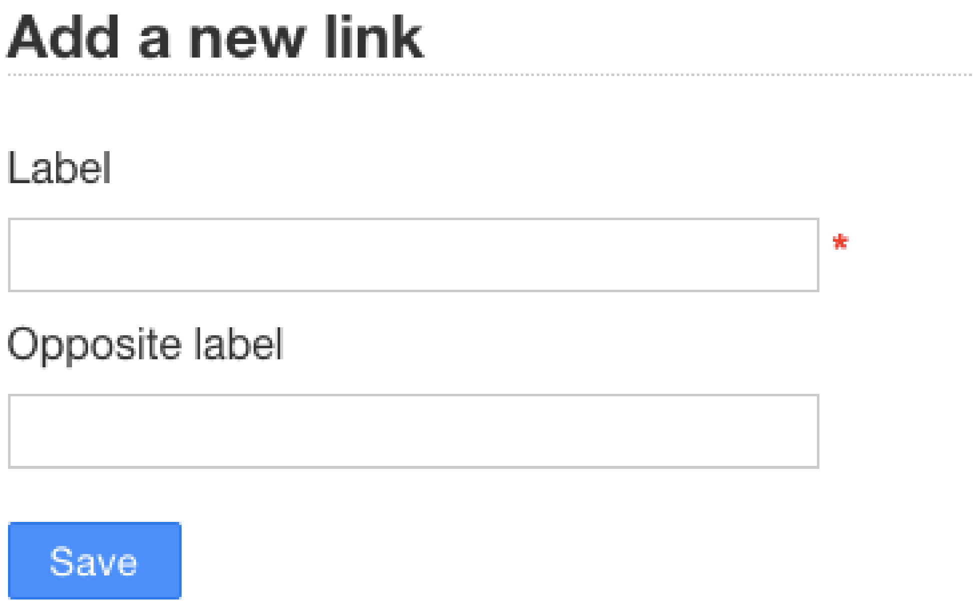
1.15 Advanced Search Syntax

Kanboard uses a simple query language for advanced search. You can search in tasks, comments, subtasks, links but also in the activity stream.



Link labels

Link labels	Actions
relates to	Edit or Remove
blocks | is blocked by	Edit or Remove
is blocked by | blocks	Edit or Remove
duplicates | is duplicated by	Edit or Remove
is duplicated by | duplicates	Edit or Remove
is a child of | is a parent of	Edit or Remove
is a parent of | is a child of	Edit or Remove
targets milestone | is a milestone of	Edit or Remove
is a milestone of | targets milestone	Edit or Remove
fixes | is fixed by	Edit or Remove
is fixed by | fixes	Edit or Remove



Add a new link

Label

*

Opposite label

1.15.1 Example of Query

This example will return all tasks assigned to me with a due date for tomorrow and a title that contains "my title":

```
assigne:me due:tomorrow my title
```

1.15.2 Project Search

Search by task id or title

- Search by task id: `#123`
- Search by task id and task title: `123`
- Search by task title: anything that doesn't match any search attributes

Search by status

Attribute: **status**

- Query to find open tasks: `status:open`
- Query to find closed tasks: `status:closed`

Search by assignee

Attribute: **assignee**

- Query with the full name: `assignee:"Frederic Guillot"`
- Query with the username: `assignee:fguillot`
- Multiple assignee lookup: `assignee:user1 assignee:"John Doe"`
- Query for unassigned tasks: `assignee:nobody`
- Query for my assigned tasks: `assignee:me`

Search by task creator

Attribute: **creator**

- Tasks created by myself: `creator:me`
- Tasks created by John Doe: `creator:"John Doe"`
- Tasks created by the user id #1: `creator:1`

Search by subtask assignee

Attribute: **subtask:assignee**

- Example: `subtask:assignee:"John Doe"`

Search by color

Attribute: **color**

- Query to search by color id: `color:blue`
- Query to search by color name: `color:"Deep Orange"`

Search by the due date

Attribute: **due**

- Search tasks due today: `due:today`
- Search tasks due tomorrow: `due:tomorrow`
- Search tasks due yesterday: `due:yesterday`
- Search tasks due with the exact date: `due:2015-06-29`
- Search tasks without a due date: `due:none`

The date must use the ISO 8601 format: **YYYY-MM-DD**.

All string formats supported by the `strtotime()` function are supported, for example `next Thursday`, `-2 days`, `+2 months`, `tomorrow`, etc.

Operators supported with a date:

- Greater than: **due:>2015-06-29**
- Lower than: **due:<2015-06-29**
- Greater than or equal: **due:>=2015-06-29**
- Lower than or equal: **due:<=2015-06-29**

Search by modification date

Attribute: **modified** or **updated**

The date formats are the same as the due date.

There is also a filter by recently modified tasks: `modified:recently`.

This query will use the same value as the board highlight period configured in settings.

Search by creation date

Attribute: **created**

Works in the same way as the modification date queries.

Search by creation date with range

Attribute: **createdRange**

Date separator `..` (two dots)

Example: `createdRange:2018/01/21..2018/01/31` or `createdRange:"2018-01-21..2018-01-31"`

Search by completion date with range

Attribute: **completedRange**

Date separator `..` (two dots)

Example: `completedRange:2018/01/21..2018/01/31` or `completedRange:"2018-01-21..2018-01-31"`

Search by modification date with range

Attribute: **updatedRange**, **modifiedRange**

Date separator `..` (two dots)

Example: `updatedRange:2018/01/21..2018/01/31` or `updatedRange:"2018-01-21..2018-01-31"`

Search by moved date with range

Attribute: **movedRange**

Date separator `..` (two dots)

Example: `movedRange:2018/01/21..2018/01/31` or `movedRange:"2018-01-21..2018-01-31"`

Search by start date

Attribute: **started**

Search by description

Attribute: **description** or **desc**

Example: `description:"text search"`

Search by completion

Attribute: **completed**

Search by external reference

The task reference is an external ID of your task, for example a ticket number from another software.

- Find tasks with a reference: `ref:1234` or `reference:TICKET-1234`
- Find tasks with no reference: `reference:none`
- Wildcard search: `ref:TICKET-*`

Search by category

Attribute: **category**

- Find tasks with a specific category: `category:"Feature Request"`
- Find all tasks that have those categories: `category:"Bug" category:"Improvements"`
- Find tasks with no category assigned: `category:none`

Search by project

Attribute: **project**

- Find tasks by project name: `project:"My project name"`
- Find tasks by project id: `project:23`
- Find tasks for several projects: `project:"My project A" project:"My project B"`

Search by columns

Attribute: **column**

- Find tasks by column name: `column:"Work in progress"`
- Find tasks for several columns: `column:"Backlog" column:ready`

Search by swimlane

Attribute: **swimlane**

- Find tasks by swim-lane: `swimlane:"Version 42"`
- Find tasks into several swim-lanes: `swimlane:"Version 1.2" swimlane:"Version 1.3"`

Search by task link

Attribute: **link**

- Find tasks by link name: `link:"is a milestone of"`
- Find tasks into several links: `link:"is a milestone of" link:"relates to"`

Search by comment

Attribute: **comment**

- Find comments that contains this title: `comment:"My comment message"`

Search by tags

Attribute: **tag**

- Example: `tag:"My tag"`

Search by score/complexity

Attribute: **score** or **complexity**

- `score:>=21`
- `complexity:8`

1.15.3 Activity Stream Search

Search events by task title

Attribute: **title** or none (default)

- Example: `title:"My task"`
- Search by task id: `#123`

Search events by task status

Attribute: **status**

Search by event creator

Attribute: **creator**

Search by event creation date

Attribute: **created**

Search events by project

Attribute: **project**

1.16 Markdown Syntax

Kanboard use the Markdown syntax for comments or task descriptions.

1.16.1 Bold and italic

- Bold text: Use 2 asterisks or 2 underscores
- Italic text: Use 1 asterisk or 1 underscore

```
This **word** is very __important__.

And here, an *italic* word with one _underscore_.
```

1.16.2 Unordered Lists

Unordered list can use asterisks, minuses or pluses.

```
- Item 1
- Item 2
- Item 3

or

* Item 1
* Item 2
* Item 3
```

1.16.3 Ordered lists

Ordered lists are prefixed by a number like that:

```
1. Do that first
2. Do this
3. And that
```

1.16.4 Links

```
[My link title](https://kanboard.org/)

<https://kanboard.org>
```

1.16.5 Source code

Inline code

Use a backtick.

```
Execute this command: `tail -f /var/log/messages`.
```

Code blocks

Use 3 backticks with eventually the language name.

~~~
```php
<?php

phpinfo();

?>
```
~~~

1.16.6 Titles

```
# Title level 1

## Title level 2

### Title level 3
```

1.16.7 Tables

```
| Header 1  | Header 2 |
| --------- |--------- |
| a         | b        |
| c         | d        |
```

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily.

1.17 Keyboard Shortcuts

Keyboard shortcuts availability depends of the page you are presently.

1.17.1 Project views (Board, Calendar, List, Gantt)

- Switch to the project overview = **v o**
- Switch to the board view = **v b** (press on **v** then **b**)
- Switch to the calendar view = **v c**
- Switch to the list view = **v l**
- Switch to the Gantt view = **v g**

1.17.2 Board view

- New task = **n**
- Expand/collapse tasks = **s**
- Compact/wide view = **c**

1.17.3 Task view

- Edit task = **e**
- New subtask = **s**
- New comment = **c**
- New internal link = **l**

1.17.4 Application

- Display list of keyboard shortcuts = **?**
- Open board switcher = **b**
- Go to the search box = **f**
- Reset the search box = **r**
- Close dialog box = **ESC**
- Submit form = **CTRL+ENTER** or **+ENTER**

CHAPTER 2

Administrator's Guide

Installation, deployment and administration guides.

2.1 Requirements

2.1.1 Server Side

Compatible Operating Systems

Operating System
Alpine Linux
Linux Ubuntu
Linux Centos
Linux Redhat
Linux Debian
FreeBSD
Microsoft Windows

Note: The recommended operating system is GNU/Linux (Debian/Ubuntu/RHEL/Alpine Linux).

Compatible Databases

Database
Sqlite >= 3.7
Mysql >= 5.6
MariaDB >= 10
Postgresql >= 9.4

Which database to choose?

Type	Usage
Sqlite	Single user or small team (almost no concurrency)
Mysql/Postgres	Larger team, high-availability configuration

Note:

- The recommended database is Postgresql.
- Do not use Sqlite with NFS.

Compatible Web Servers

Web Server
Apache HTTP Server
Nginx
Microsoft IIS
Caddy Server

Kanboard is pre-configured to work with Apache (URL rewriting).

Warning:

- Kanboard is NOT compatible with Apache `mod_security`.
- If you use Apache, you must have the module `mod_version`.

PHP Versions

PHP Version
PHP >= 7.2

Note:

- Since the version 1.2.13, Kanboard requires at least PHP 7.2
- The latest version of PHP is recommended.

PHP Extensions Required

PHP Extension	Note
pdo_sqlite	Only if you use Sqlite
pdo_mysql	Only if you use Mysql/MariaDB
pdo_pgsql	Only if you use Postgres
gd	
mbstring	
openssl	
json	
hash	
ctype	
session	
filter	
xml	
SimpleXML	
dom	

Optional PHP extensions

PHP Extension	Note
zip	Used to install plugins from web ui
ldap	Only for LDAP integration
curl	Use cURL as HTTP client

Recommendations

- Modern Linux or Unix operating system with the latest version of PHP.
- Best performances are obtained with the latest version of PHP with OpCode caching activated.

2.1.2 Client side

Browsers

Always use a modern browser with the latest version if possible:

Browser
Safari
Google Chrome
Mozilla Firefox
Microsoft Edge

Note: The recommended browsers are Mozilla Firefox or Google Chrome.

Warning: Microsoft Internet Explorer is not supported since version 1.2.11

Devices

Device	Screen resolution
Laptop or desktop	>= 1366 x 768
Tablet	>= 1024 x 768

2.2 Installation Instructions

Note: Check the *requirements* before going further.

2.2.1 From the archive (stable version)

1. You must have a web server with PHP installed
2. Download the source code and copy the directory `kanboard` where you want
3. Check if the directory `data` is writeable by the web server user
4. With your browser go to http://yourpersonalserver/kanboard
5. The default login and password is **admin/admin**
6. Start using the software
7. Don't forget to change your password!

The `data` folder is used to store:

- Sqlite database: `db.sqlite`
- Debug file: `debug.log` (if debug mode is enabled with the `file` driver)
- Uploaded files: `files/*`
- Image thumbnails: `files/thumbnails/*`

People who are using a remote database (Mysql/Postgresql) and object storage (AWS S3 or similar) don't necessarily need to have a persistent local data folder or to change the permissions for the folder.

2.2.2 From the git repository (development version)

1. `git clone https://github.com/kanboard/kanboard.git`
2. Go to the third step just above

Note: This method will install the **current development version**, use at your own risk.

Warning: **Security measures:**

- Do not forget to change the default user/password

- Do not allow everybody to access the directory `data` from the URL. A `.htaccess` file and a `web.config` file are included for Apache/IIS but other web servers have to be configured manually.

2.3 Upgrading to a New Version

Warning: Please, do not update the software blindly without reading the ChangeLog. Always check for breaking changes if any.

Most of the time, upgrading Kanboard to a newer version is seamless. The process could be summarized to simply copy your data folder to the new Kanboard folder. Kanboard will run database migrations automatically for you.

2.3.1 Important things to do before updating

- **Always make a backup of your data before upgrading**
- **Check that your backup is valid!**
- Always read the ChangeLog and check for breaking changes
- Stop the worker if you use it
- Put the web server in maintenance mode to avoid people to use the software while upgrading

2.3.2 From the archive (stable version)

1. Decompress the new archive
2. Copy the `data` folder into the newly uncompressed directory
3. Copy your custom `config.php` if you have one
4. If you have installed some plugins, use the latest version
5. Make sure the directory `data` is writeable by your web server user
6. Modify `.htaccess` if needed (f.e. forcing SSL)
7. Test
8. Remove your old Kanboard directory

2.3.3 From the repository (development version)

1. `git pull`
2. Login and check if everything is ok

- This method will install the **current development version**, use at your own risk.
- Do not update the software blindly without checking the ChangeLog.

2.3.4 Running SQL migrations manually

By default, SQL migrations are executed automatically. The schema version is checked at each request. In this way, when you upgrade Kanboard to another version, the database schema is updated for you. This method **is not perfect**.

- **When you run the migrations, make sure only one process is accessing to the database**
- Put your Kanboard instance in "maintenance mode" to avoid people using the software while you are altering the database schema

To run database migrations manually, set the parameter `DB_RUN_MIGRATIONS` at `false` in your config file.

When you will have to upgrade Kanboard, run this command:

```
./cli db:migrate
```

2.4 Run Kanboard with Docker

Warning: Please, do not update the software blindly without reading the ChangeLog. Always check for breaking changes if any.

2.4.1 Docker Registries

Registry	Description
Docker Hub	`docker.io/kanboard/kanboard`
GitHub Container Registry	`ghcr.io/kanboard/kanboard`

2.4.2 Docker Tags

Tag	Description
`latest`	Latest stable release
`nightly`	Nightly build (latest development changes)
`v1.2.x`	Specific version of Kanboard

2.4.3 Environment Variables

All Kanboard configuration options can be used as environment variable.

2.4.4 Volumes

Path	Description
`/var/www/app/data`	Application data (Sqlite database, attachments, etc.)
`/var/www/app/plugins`	Plugins
`/etc/nginx/ssl`	SSL certificates

2.4.5 Custom Config File

- The container already include a custom config file located at `/var/www/app/config.php`.
- You can store your own config file on the data volume: `/var/www/app/data/config.php`.
- You must restart the container to take into account the new parameters of your custom config file.

2.4.6 Running the Container

Basic Usage

```
docker run -d --name kanboard -p 80:80 -t kanboard/kanboard:v1.2.8
```

Docker Compose

There is a `docker-compose.yml` file in Kanboard repository. Here an example with Sqlite:

```
version: '2'
services:
  kanboard:
    image: kanboard/kanboard:latest
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - kanboard_data:/var/www/app/data
      - kanboard_plugins:/var/www/app/plugins
      - kanboard_ssl:/etc/nginx/ssl
volumes:
  kanboard_data:
  kanboard_plugins:
  kanboard_ssl:
```

Another example with MariaDB:

```
version: '2'
services:
  kanboard:
    image: kanboard/kanboard:latest
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - kanboard_data:/var/www/app/data
      - kanboard_plugins:/var/www/app/plugins
      - kanboard_ssl:/etc/nginx/ssl
    environment:
      DATABASE_URL: mysql://kanboard:kanboard-secret@db/kanboard
  db:
    image: mariadb:latest
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: kanboard
```

(continues on next page)

(continued from previous page)

```
      MYSQL_USER: kanboard
      MYSQL_PASSWORD: kanboard-secret
volumes:
  kanboard_data:
  kanboard_plugins:
  kanboard_ssl:
```

Starting the container with Docker Compose:

```
docker-compose up
```

2.4.7 Build Your Own Docker Image

Clone the Kanboard repository and run the following command:

```
make docker-image
```

Note: You must use the SMTP method or a plugin like Mailgun/Sendgrid/Postmark to send emails.

2.5 Installation on RedHat/Centos/Oracle Linux Enterprise

Warning: This page hasn't been updated for a while, and it's probably obsolete.

2.5.1 Centos 7

Install PHP and Apache:

```
yum install -y php php-xml php-mbstring php-pdo php-gd unzip wget
```

By default, Centos 7 use PHP 5.4.16 and Apache 2.4.6.

Restart Apache:

```
systemctl restart httpd.service
```

Install Kanboard:

```
cd /var/www/html

# Download the latest release from https://github.com/kanboard/kanboard/releases
wget https://github.com/kanboard/kanboard/archive/v<version>.zip

unzip kanboard-<version>.zip
chown -R apache:apache kanboard-<version>/data
rm kanboard-<version>.zip
```

2.5.2 SELinux restrictions

If SELinux is enabled, be sure that the Apache user can write to the directory data:

```
chcon -R -t httpd_sys_content_rw_t /var/www/html/kanboard/data
```

Be sure to configure your server to allow Kanboard to send emails and make external network requests, for example with SELinux:

```
setsebool -P httpd_can_network_connect=1
```

Allowing external connections is necessary if you use LDAP, SMTP, Web hooks or any third-party integration.

2.6 Installation on Ubuntu

2.6.1 Ubuntu Focal Fossa 20.04 LTS

Install Apache and PHP:

```
sudo apt update
sudo apt install -y apache2 libapache2-mod-php php-cli php-mbstring php-sqlite3 php-
↪opcache php-json php-mysql php-pgsql php-ldap php-gd php-xml
```

(or) Install Nginx and PHP:

```
sudo apt update
sudo apt install nginx php-fpm php-mysql php-pgsql php-gd php-mbstring php-sqlite3␣
↪php-xml
```

Install Kanboard:

```
#adapt version number
version=1.2.24

# Download the latest release from https://github.com/kanboard/kanboard/releases
wget https://github.com/kanboard/kanboard/archive/v$version.tar.gz
tar xzvf v$version.tar.gz -C /var/www/html/
chown -R www-data:www-data /var/www/html/kanboard-$version/data

rm v$version.tar.gz
```

Note:

- You might need to enable PHP extensions with the command `phpenmod`.
- Some features of Kanboard requires that you run a daily background job.

2.6.2 Ubuntu Bionic 18.04 LTS

Install Apache and PHP:

```
sudo apt-get update
sudo apt-get install -y apache2 libapache2-mod-php7.2 php7.2-cli php7.2-mbstring php7.
↪2-sqlite3 \
    php7.2-opcache php7.2-json php7.2-mysql php7.2-pgsql php7.2-ldap php7.2-gd php7.2-
↪xml
```

(or) Install Nginx and PHP:

```
sudo apt-get update
sudo apt-get install nginx php7.2-fpm php7.2-mysql php7.2-pgsql php7.2-gd php7.2-
↪mbstring php7.2-sqlite3 \
    php7.2-xml
```

Install Kanboard:

```
cd /var/www/html

# Download the latest release from https://github.com/kanboard/kanboard/releases
wget https://github.com/kanboard/kanboard/archive/v<version>.zip

unzip kanboard-<version>.zip
chown -R www-data:www-data kanboard-<version>/data
rm kanboard-<version>.zip
```

Note:

- You might need to enable PHP extensions with the command `phpenmod`.
- Some features of Kanboard requires that you run a daily background job.

2.7 Installation on Debian

2.7.1 Debian 10 (Buster)

Install Apache and PHP:

```
apt update
apt install -y apache2 libapache2-mod-php php-cli php-mbstring \
    php-sqlite3 php-opcache php-json php-ldap php-gd php-xml  \
    php-mysql php-pgsql php-curl php-zip

systemctl enable apache2
```

Install Kanboard:

```
version=1.2.13
wget https://github.com/kanboard/kanboard/archive/v$version.tar.gz
tar xzvf v$version.tar.gz -C /var/www/html/
chown -R www-data:www-data /var/www/html/kanboard-$version/data

rm v$version.tar.gz
```

Note: Some features of Kanboard requires that you run a daily background job.

2.8 Installation on OpenSuse

Warning: This page hasn't been updated for a while, and it's probably obsolete.

2.8.1 OpenSuse Leap 42.3

```
# install required packages
sudo zypper install apache2-mod_php7 php7-openssl php7-gd php7-mbstring php7-mcrypt␣
↪php7-mysql php7-xmlrpc php7-ctype php7-json

# enable php7
sudo a2enmod php7

cd /srv/www/htdocs

# Download the latest release from https://github.com/kanboard/kanboard/releases

sudo wget https://github.com/kanboard/kanboard/archive/v<version>.zip
sudo unzip kanboard-<version>.zip

# Add permissions
sudo chown -R wwwrun /srv/www/htdocs/kanboard-<version>/data

# restart apache
sudo rcapache2 restart

# cleanup
sudo rm kanboard-<version>.zip
```

2.9 FreeBSD Installation

Warning: This page hasn't been updated for a while, and it's probably obsolete.

2.9.1 Package Installation

```
$ pkg upgrade -y
$ pkg install -y apache24 mod_php56 kanboard
```

Enable Apache in your `/etc/rc.conf`:

```
$ sysrc apache24_enable=yes
```

Set up PHP for Apache:

```
$ echo "AddType application/x-httpd-php .php" >> /usr/local/etc/apache24/Includes/php.
↪conf
$ echo "DirectoryIndex index.php index.html" >> /usr/local/etc/apache24/Includes/php.
↪conf
```

Then start Apache:

```
$ service apache24 start
```

Add symlink to Kanboard folder into your Apache docroot:

```
cd /usr/local/www/apache24/data
ln -s /usr/local/www/kanboard
```

Go to http://your.server.domain.tld/kanboard and enjoy!

Note: If you want to use additional features like LDAP integration etc. please install proper PHP module using pkg. - You may have to adjust the permissions of the folder data

2.9.2 Installing from ports

Generally 3 elements have to be installed:

- Apache
- mod_php for Apache
- Kanboard

Fetch and extract ports...

```
$ portsnap fetch
$ portsnap extract
```

or update already existing:

```
$ portsnap fetch
$ portsnap update
```

More details regarding portsnap can be found in the FreeBSD Handbook.

Install Apache:

```
$ cd /usr/ports/www/apache24
$ make install clean
```

Enable Apache in your `/etc/rc.conf`:

```
$ sysrc apache24_enable=yes
```

Install mod_php for Apache:

```
$ cd /usr/ports/www/mod_php5
$ make install clean
```

Install Kanboard form ports:

```
$ cd /usr/ports/www/kanboard
$ make install clean
```

Set up PHP for Apache:

```
$ echo "AddType application/x-httpd-php .php" >> /usr/local/etc/apache24/Includes/php.
↪conf
$ echo "DirectoryIndex index.php index.html" >> /usr/local/etc/apache24/Includes/php.
↪conf
```

Then start Apache:

```
$ service apache24 start
```

Go to http://your.server.domain.tld/kanboard and enjoy!

Note: If you want to use additional features like LDAP integration etc. please install proper PHP module from `lang/php5-extensions`.

2.9.3 Manual installation

As of version 1.0.16 Kanboard can be found in FreeBSD ports there is no need to install it manually.

Note: Port is being hosted on bitbucket. Feel free to comment, fork and suggest updates!

2.10 Installation on Microsoft Windows Server

Warning: This page hasn't been updated for a while, and it's probably obsolete.

2.10.1 Windows Server and Apache

This guide will help you to setup step by step Kanboard on a Windows Server with Apache and PHP.

Note: If you have a 64 bits platform choose "x64" otherwise choose "x86" for 32-bit systems.

Visual C++ Redistributable Installation

PHP and Apache are compiled with Visual Studio so you need to install this library if it's not already done.

1. Download the library from the official Microsoft website
2. Run the installer `vcredist_x64.exe` or `vcredist_x86.exe` according to your platform

Apache Installation

1. Download Apache binary from Apache Lounge
2. Unzip the Apache24 folder to `C:\Apache24`

Define the server name:

Open the file `C:\Apache24\conf\httpd.conf` and add the directive:

```
ServerName localhost
```

Install the Apache service

Open a command prompt (`cmd.exe`) and go to the directory `C:\Apache24\bin`:

```
cd C:\Apache24\bin

# Install the windows service
httpd.exe -k install
```

Install ApacheMonitor

- Double click on `C:\Apache24\bin\ApacheMonitor.exe`, or put it in your startup folder.
- Right click on the icon and start Apache

Check the Apache installation

Go to http://localhost/ you should see a blank page with the text "It works!".

PHP installation

1. Download the last stable version of PHP from the official PHP website, choose the **Thread Safe** version and use the exact same build type as Apache: x86 or x64
2. Unzip the files to `C:\php`
3. Navigate to the PHP folder and rename the file `php.ini-production` to `php.ini`

Edit the `php.ini`:

Uncomment extension directory:

```
extension_dir = "C:/php/ext"
```

Uncomment these PHP modules:

```
extension=php_gd2.dll
extension=php_ldap.dll
extension=php_mbstring.dll
extension=php_openssl.dll
extension=php_pdo_sqlite.dll
```

Set the time zone:

```
date.timezone = America/Montreal
```

The list of supported time zones can be found in the PHP documentation.

Load the PHP module for Apache:

Add this configuration in the file `C:\Apache24\conf\httpd.conf`:

```
LoadModule php5_module "c:/php/php5apache2_4.dll"
AddHandler application/x-httpd-php .php

# configure the path to php.ini
PHPIniDir "C:/php"

# change this directive
DirectoryIndex index.php index.html
```

Restart Apache.

Test your PHP installation:

Create a file named `phpinfo.php` in the folder `C:\Apache24\htdocs`:

```
<?php

phpinfo();

?>
```

Go to http://localhost/phpinfo.php and should see all information about your PHP installation.

Kanboard installation

- Download the zip file
- Decompress the archive in `C:\Apache24\htdocs\kanboard`
- Open your web browser to use Kanboard http://localhost/kanboard/
- The default credentials are **admin/admin**

2.10.2 Windows Server and IIS

This guide will help you to setup step by step Kanboard on a Windows Server with IIS and PHP.

PHP Installation

- Install IIS on your server (Add a new role and don't forget to enable CGI/FastCGI)
- Install PHP by following the official documentation:
  - Microsoft IIS 5.1 and IIS 6.0
  - Microsoft IIS 7.0 and later
  - PHP for Windows is available here

PHP.ini

You need at least, these extensions in your `php.ini`:

```
extension=php_gd2.dll
extension=php_ldap.dll
extension=php_mbstring.dll
extension=php_openssl.dll
extension=php_pdo_sqlite.dll
```

Do not forget to set the time zone:

```
date.timezone = America/Montreal
```

The list of supported time zones can be found in the PHP documentation.

Note:

- Don't forget to enable the required php extensions mentioned above
- If you got an error about "the library MSVCP110.dll is missing", you probably need to download the Visual C++ Redistributable for Visual Studio from the Microsoft website.

IIS Modules

The Kanboard archive contains a `web.config` file to enable URL rewriting. This configuration require the Rewrite module for IIS.

If you don't have the rewrite module, you will get an internal server error (500) from IIS. If you don't want to have Kanboard with nice URLs, you can remove the file `web.config`.

Kanboard installation

- Download the zip file
- Decompress the archive in `C:\inetpub\wwwroot\kanboard`
- Make sure the directory `data` is writable by the IIS user
- Open your web browser to use Kanboard http://localhost/kanboard/
- The default credentials are **admin/admin**

2.11 Configuration File

You can customize the default settings of Kanboard by adding a file `config.php` at the project root or in the `data` folder. You can also rename the file `config.default.php` to `config.php` and change the desired values.

2.11.1 Enable/Disable debug mode

```
define('DEBUG', true);
define('LOG_DRIVER', 'file'); // Other drivers are: syslog, stdout, stderr, system or␣
↪file

// By default, the log file is in data/debug.log but you can change the path:
define('LOG_FILE', '/path/to/debug.log');
```

- The log driver must be defined if you enable the debug mode.
- The debug mode logs all SQL queries and the time taken to generate pages.
- The `system` driver use the built-in PHP logger which could be configured in the php.ini. By default, log messages are sent to the web server logs.

2.11.2 Plugins

Plugin folder:

```
define('PLUGINS_DIR', 'data/plugins');
```

Enable/disable plugin installation from the user interface:

```
define('PLUGIN_INSTALLER', false); // Default is false since Kanboard v1.2.8
```

Change default plugin directory URL:

```
define('PLUGIN_API_URL', 'https://kanboard.org/plugins.json');
```

2.11.3 Folder for uploaded files

```
define('FILES_DIR', 'data/files');
```

2.11.4 Cache parameters

```
// Available cache drivers are "file" and "memory"
define('CACHE_DRIVER', 'memory');

// Cache folder to use if cache driver is "file" (must be writeable by the web server␣
↪user)
define('CACHE_DIR', DATA_DIR.DIRECTORY_SEPARATOR.'cache');
```

2.11.5 Enable/disable url rewrite

```
define('ENABLE_URL_REWRITE', false);
```

2.11.6 Email configuration

```
// Enable/disable email configuration from the user interface
define('MAIL_CONFIGURATION', true);

// E-mail address used for the "From" header (notifications)
define('MAIL_FROM', 'notifications@kanboard.local');

// Mail transport to use: "smtp", "sendmail" or "mail" (PHP mail function)
define('MAIL_TRANSPORT', 'mail');
```

(continues on next page)

(continued from previous page)

```
// SMTP configuration to use when the "smtp" transport is chosen
define('MAIL_SMTP_HOSTNAME', '');
define('MAIL_SMTP_PORT', 25);
define('MAIL_SMTP_USERNAME', '');
define('MAIL_SMTP_PASSWORD', '');
define('MAIL_SMTP_HELO_NAME', null); // valid: null (default), or FQDN
define('MAIL_SMTP_ENCRYPTION', null); // Valid values are "null", "ssl" or "tls"

// Sendmail command to use when the transport is "sendmail"
define('MAIL_SENDMAIL_COMMAND', '/usr/sbin/sendmail -bs');

// E-mail address used for the "Bcc" header to send a copy of all notifications
define('MAIL_BCC', '');
```

2.11.7 Database settings

```
// Run automatically database migrations
// If set to false, you will have to run manually the SQL migrations from the CLI␣
↪during the next Kanboard upgrade
// Do not run the migrations from multiple processes at the same time (example: web␣
↪page + background worker)
define('DB_RUN_MIGRATIONS', true);

// Database driver: sqlite, mysql or postgres (sqlite by default)
define('DB_DRIVER', 'sqlite');

// Mysql/Postgres username
define('DB_USERNAME', 'root');

// Mysql/Postgres password
define('DB_PASSWORD', '');

// Mysql/Postgres hostname
define('DB_HOSTNAME', 'localhost');

// Mysql/Postgres database name
define('DB_NAME', 'kanboard');

// Mysql/Postgres custom port (null = default port)
define('DB_PORT', null);

// Mysql SSL key
define('DB_SSL_KEY', null);

// Mysql SSL certificate
define('DB_SSL_CERT', null);

// Mysql SSL CA
define('DB_SSL_CA', null);
```

2.11.8 LDAP settings

```
// Enable LDAP authentication (false by default)
define('LDAP_AUTH', false);

// LDAP server hostname
define('LDAP_SERVER', '');

// LDAP server port (389 by default)
define('LDAP_PORT', 389);

// By default, require certificate to be verified for ldaps:// style URL. Set to␣
↪false to skip the verification
define('LDAP_SSL_VERIFY', true);

// Enable LDAP START_TLS
define('LDAP_START_TLS', false);

// By default Kanboard lowercase the ldap username to avoid duplicate users (the␣
↪database is case sensitive)
// Set to true if you want to preserve the case
define('LDAP_USERNAME_CASE_SENSITIVE', false);

// LDAP bind type: "anonymous", "user" or "proxy"
define('LDAP_BIND_TYPE', 'anonymous');

// LDAP username to use with proxy mode
// LDAP username pattern to use with user mode
define('LDAP_USERNAME', null);

// LDAP password to use for proxy mode
define('LDAP_PASSWORD', null);

// LDAP DN for users
// Example for ActiveDirectory: CN=Users,DC=kanboard,DC=local
// Example for OpenLDAP: ou=People,dc=example,dc=com
define('LDAP_USER_BASE_DN', '');

// LDAP pattern to use when searching for a user account
// Example for ActiveDirectory: '(&(objectClass=user)(sAMAccountName=%s))'
// Example for OpenLDAP: 'uid=%s'
define('LDAP_USER_FILTER', '');

// LDAP attribute for the user in the group filter
// 'username' or 'dn'
define('LDAP_GROUP_USER_ATTRIBUTE', 'username');

// LDAP attribute for username
// Example for ActiveDirectory: 'samaccountname'
// Example for OpenLDAP: 'uid'
define('LDAP_USER_ATTRIBUTE_USERNAME', 'uid');

// LDAP attribute for user full name
// Example for ActiveDirectory: 'displayname'
// Example for OpenLDAP: 'cn'
define('LDAP_USER_ATTRIBUTE_FULLNAME', 'cn');
```

(continues on next page)

(continued from previous page)

```
// LDAP attribute for user email
define('LDAP_USER_ATTRIBUTE_EMAIL', 'mail');

// LDAP attribute to find groups in user profile
define('LDAP_USER_ATTRIBUTE_GROUPS', 'memberof');

// LDAP attribute for user avatar image: thumbnailPhoto or jpegPhoto
define('LDAP_USER_ATTRIBUTE_PHOTO', '');

// LDAP attribute for user language, example: 'preferredlanguage'
// Put an empty string to disable language sync
define('LDAP_USER_ATTRIBUTE_LANGUAGE', '');

// Allow automatic LDAP user creation
define('LDAP_USER_CREATION', true);

// Set new user as Manager
define('LDAP_USER_DEFAULT_ROLE_MANAGER', false);

// LDAP DN for administrators
// Example: CN=Kanboard-Admins,CN=Users,DC=kanboard,DC=local
define('LDAP_GROUP_ADMIN_DN', '');

// LDAP DN for managers
// Example: CN=Kanboard Managers,CN=Users,DC=kanboard,DC=local
define('LDAP_GROUP_MANAGER_DN', '');

// Enable LDAP group provider for project permissions
// The end-user will be able to browse LDAP groups from the user interface and allow␣
↪access to specified projects
define('LDAP_GROUP_PROVIDER', false);

// LDAP Base DN for groups
define('LDAP_GROUP_BASE_DN', '');

// LDAP group filter
// Example for ActiveDirectory: (&(objectClass=group)(sAMAccountName=%s*))
define('LDAP_GROUP_FILTER', '');

// LDAP user group filter
// If this filter is configured, Kanboard will search user groups in LDAP_GROUP_BASE_
↪DN
// Example for OpenLDAP: (&(objectClass=posixGroup)(memberUid=%s))
define('LDAP_GROUP_USER_FILTER', '');

// LDAP attribute for the group name
define('LDAP_GROUP_ATTRIBUTE_NAME', 'cn');
```

2.11.9 Reverse-Proxy Authentication settings

```
// Enable/disable the reverse proxy authentication
define('REVERSE_PROXY_AUTH', false);

// Header name to use for the username
define('REVERSE_PROXY_USER_HEADER', 'REMOTE_USER');
```

(continues on next page)

(continued from previous page)

```
// Header name to use for the username
define('REVERSE_PROXY_EMAIL_HEADER', 'REMOTE_EMAIL');

// Username of the admin, by default blank
define('REVERSE_PROXY_DEFAULT_ADMIN', '');

// Default domain to use for setting the email address
define('REVERSE_PROXY_DEFAULT_DOMAIN', '');
```

2.11.10 RememberMe Authentication settings

```
// Enable/disable remember me authentication
define('REMEMBER_ME_AUTH', true);
```

2.11.11 Secure HTTP headers settings

```
// Enable or disable "Strict-Transport-Security" HTTP header
define('ENABLE_HSTS', true);

// Enable or disable "X-Frame-Options: DENY" HTTP header
define('ENABLE_XFRAME', true);
```

2.11.12 Logging

By default, Kanboard do not log anything. If you want to enable the logging, you have to set a log driver.

```
// Available log drivers: syslog, stderr, stdout or file
define('LOG_DRIVER', '');

// Log filename if the log driver is "file"
define('LOG_FILE', __DIR__.DIRECTORY_SEPARATOR.'data'.DIRECTORY_SEPARATOR.'debug.log
↪');
```

2.11.13 Brute-force protection

```
// Enable captcha after 3 authentication failure
define('BRUTEFORCE_CAPTCHA', 3);

// Lock the account after 6 authentication failure
define('BRUTEFORCE_LOCKDOWN', 6);

// Lock account duration in minute
define('BRUTEFORCE_LOCKDOWN_DURATION', 15);
```

2.11.14 Session

```
// Session duration in second (0 = until the browser is closed)
// See http://php.net/manual/en/session.configuration.php#ini.session.cookie-lifetime
define('SESSION_DURATION', 0);

// Session handler: db or php
//    db: session information is stored inside the database (default)
//    php: session information is stored by the internal PHP session handlers
// See https://www.php.net/manual/en/session.customhandler.php
define('SESSION_HANDLER', 'db');
```

2.11.15 HTTP Client

HTTP proxy configuration:

```
define('HTTP_PROXY_HOSTNAME', '');
define('HTTP_PROXY_PORT', '3128');
define('HTTP_PROXY_USERNAME', '');
define('HTTP_PROXY_PASSWORD', '');
define('HTTP_PROXY_EXCLUDE', 'localhost'); // Only for cURL
```

To allow self-signed certificates:

```
// Set to false to allow self-signed certificates
define('HTTP_VERIFY_SSL_CERTIFICATE', true);
```

2.11.16 Various settings

```
// Escape html inside markdown text
define('MARKDOWN_ESCAPE_HTML', true);

// API alternative authentication header, the default is HTTP Basic Authentication␣
↪defined in RFC2617
define('API_AUTHENTICATION_HEADER', '');

// Hide login form, useful if all your users use Google/Github/ReverseProxy␣
↪authentication
define('HIDE_LOGIN_FORM', false);

// Disabling logout (for external SSO authentication)
define('DISABLE_LOGOUT', false);

// Override API token stored in the database, useful for automated tests
define('API_AUTHENTICATION_TOKEN', 'My unique API Token');

// TOTP (2FA) issuer name
define('TOTP_ISSUER', 'Kanboard');

// Comma separated list of fields to not synchronize when using external␣
↪authentication providers
define('EXTERNAL_AUTH_EXCLUDE_FIELDS', 'username');
```

2.12 Command Line Interface

Kanboard provides a simple command line interface that can be used from any Unix terminal. This tool can be used only on the local machine.

This feature is useful to run commands outside of the web server processes.

2.12.1 Usage

- Open a terminal and go to your Kanboard directory (example: `cd /var/www/kanboard`)
- Run the command `./cli` or `php cli`

```
Kanboard v1.2.11


Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output, 2␣
↪for more verbose output and 3 for debug

Available commands:
  cronjob                            Execute daily cronjob
  css                                Minify CSS files
  help                               Displays help for a command
  job                                Execute individual job (read payload from stdin)
  js                                 Minify Javascript files
  list                               Lists commands
  version                            Display Kanboard version
  worker                             Execute queue worker
 db
  db:migrate                         Execute SQL migrations
  db:version                         Show database schema version
 export
  export:daily-project-column-stats  Daily project column stats CSV export (number of␣
↪tasks per column and per day)
  export:subtasks                    Subtasks CSV export
  export:tasks                       Tasks CSV export
  export:transitions                 Task transitions CSV export
 locale
  locale:compare                     Compare application translations with the fr_FR␣
↪locale
  locale:sync                        Synchronize all translations based on the fr_FR␣
↪locale
 notification
  notification:overdue-tasks         Send notifications for overdue tasks
 plugin
  plugin:install                     Install a plugin from a remote Zip archive
```

(continues on next page)

(continued from previous page)

```
  plugin:uninstall                    Remove a plugin
  plugin:upgrade                      Update all installed plugins
 projects
  projects:archive                    Disable projects not touched during one year
  projects:archive-activities         Remove project activities after one year
  projects:daily-stats                Calculate daily statistics for all projects
 trigger
  trigger:tasks                       Trigger scheduler event for all tasks
 user
  user:reset-2fa                      Remove two-factor authentication for a user
  user:reset-password                 Change user password
```

2.12.2 Available commands

Tasks CSV export

Usage:

```
./cli export:tasks <project_id> <start_date> <end_date>
```

Example:

```
./cli export:tasks 1 2014-10-01 2014-11-30 > /tmp/my_custom_export.csv
```

CSV data are sent to `stdout`.

Subtasks CSV export

Usage:

```
./cli export:subtasks <project_id> <start_date> <end_date>
```

Example:

```
./cli export:subtasks 1 2014-10-01 2014-11-30 > /tmp/my_custom_export.csv
```

Task transitions CSV export

Usage:

```
./cli export:transitions <project_id> <start_date> <end_date>
```

Example:

```
./cli export:transitions 1 2014-10-01 2014-11-30 > /tmp/my_custom_export.csv
```

Export daily summaries data in CSV

The exported data will be printed on the standard output:

```
./cli export:daily-project-column-stats <project_id> <start_date> <end_date>
```

Example:

```
./cli export:daily-project-column-stats 1 2014-10-01 2014-11-30 > /tmp/my_custom_
↪export.csv
```

Send notifications for overdue tasks

Emails will be sent to all users with notifications enabled.

```
./cli notification:overdue-tasks
```

Optional parameters:

- `--show`: Display notifications sent
- `--group`: Group all overdue tasks for one user (from all projects) in one email
- `--manager`: Send all overdue tasks to project manager(s) in one email
- `-p|--project project_id|identifier`: Send notifications only for the given project

You can also display the overdue tasks with the flag `--show`:

```
./cli notification:overdue-tasks --show
+-----+---------+------------+------------+--------------+----------+
| Id  | Title   | Due date   | Project Id | Project name | Assignee |
+-----+---------+------------+------------+--------------+----------+
| 201 | Test    | 2014-10-26 | 1          | Project #0   | admin    |
| 202 | My task | 2014-10-28 | 1          | Project #0   |          |
+-----+---------+------------+------------+--------------+----------+
```

Example to filter by project:

```
./cli notification:overdue-tasks --project 123
```

Or if you have defined a project identifier:

```
./cli notification:overdue-tasks --project MY_PROJECT
```

Run daily project stats calculation

This command calculate the statistics of each project:

```
./cli projects:daily-stats
Run calculation for Project #0
Run calculation for Project #1
Run calculation for Project #10
```

Trigger for tasks

This command send a “daily cronjob event” to all open tasks of each project.

```
./cli trigger:tasks
Trigger task event: project_id=2, nb_tasks=1
```

Reset user password

```
./cli user:reset-password my_user
```

You will be prompted for a password and confirmation. Characters are not printed to the screen.

Remove two-factor authentication for a user

```
./cli user:reset-2fa my_user
```

Install a plugin

```
./cli plugin:install https://github.com/kanboard/plugin-github-auth/releases/download/
↪v1.0.1/GithubAuth-1.0.1.zip
```

Note: Installed files will have the same permissions as the current user

Remove a plugin

```
./cli plugin:uninstall Budget
```

Upgrade all plugins

```
./cli plugin:upgrade
* Updating plugin: Budget Planning
* Plugin up to date: Github Authentication
```

Run Background worker

```
./cli worker
```

Warning: The background worker is not really maintained anymore.

Execute individual job (mostly for debugging)

```
echo 'RAW_JOB_DATA' | ./cli job
```

Execute database migrations

If the parameter `DB_RUN_MIGRATIONS` is set to `false`, you have run the database migrations manually:

```
./cli db:migrate
```

Check database schema version

```
./cli db:version
Current version: 95
Last version: 96
```

2.13 Sqlite

Kanboard uses Sqlite by default to store its data. All tasks, projects and users are stored inside this database.

Technically, the database is just a single file located inside the directory `data` and named `db.sqlite`.

2.13.1 Export/Backup

Command line

Doing a backup is very easy, just copy the file `data/db.sqlite` somewhere else when nobody use the software.

If you want to do a backup while users are connected, you can use `sqlite3` to create the backup.

- You can dump the database to a text file of sql commands like this: `sqlite3 db.sqlite .dump > kanboard.dump.sql`
- Another option is to create a backup in sqlite format: `sqlite3 db.sqlite ".backup kanboard.backup.sqlite"`

User interface

You can also download at any time the database directly from the **settings** menu.

The downloaded database is compressed with Gzip, the filename becomes `db.sqlite.gz`.

2.13.2 Import/Restoration

There is actually no way to restore the database from the user interface. The restoration must be done manually when no body use the software.

- To restore an old backup, just replace and overwrite the actual file `data/db.sqlite`.
- To uncompress a gzipped database, execute this command from a terminal `gunzip db.sqlite.gz`.
- A backup in sql format, can be restored like this: `sqlite3 db.sqlite < kanboard.dump.sql` (db.sqlite must not exist)
- A backup in sqlite format, can be restored like this: `sqlite3 db.sqlite ".restore kanboard.backup.sqlite"`

2.13.3 Optimization

Occasionally, it's possible to optimize the database file by running the command `VACUUM`. This command rebuild the entire database and can be used for several reasons:

- Reduce the file size, deleting data produce empty space but doesn't change the file size.
- The database is fragmented due to frequent inserts or updates.

From the command line

```
sqlite3 data/db.sqlite 'VACUUM'
```

From the user interface

Go to the menu **settings** and click on the link **Optimize the database**.

For more information, read the Sqlite documentation.

2.14 MySQL/MariaDB

By default Kanboard use Sqlite to stores its data. However it's possible to use MySQL or MariaDB instead of Sqlite.

2.14.1 Requirements

- MySQL >= 5.6 or MariaDB >= 10
- The PHP extension `pdo_mysql` installed

2.14.2 MySQL configuration

Create a database

The first step is to create a database on your MySQL server. For example, you can do that from MySQL client:

```
CREATE DATABASE kanboard CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Note: Since the version 1.2.3, Kanboard uses the utf8mb4 encoding instead of utf8.

You can then assign the required permissions on the database:

```
GRANT ALTER, CREATE, DELETE, DROP, INDEX, INSERT, REFERENCES, SELECT, UPDATE, LOCK␣
↪TABLES ON kanboard.* TO 'USERNAME'@'HOST' IDENTIFIED BY 'PASSWORD';
```

Create a config file

The file `config.php` should contains these values:

```
<?php

// We choose to use MySQL instead of Sqlite
define('DB_DRIVER', 'mysql');

// MySQL parameters
define('DB_USERNAME', 'REPLACE_ME');
define('DB_PASSWORD', 'REPLACE_ME');
define('DB_HOSTNAME', 'REPLACE_ME');
define('DB_NAME', 'kanboard');
```

Note: You can also rename the template file `config.default.php` to `config.php`.

Importing SQL dump (alternative method)

For the first time, Kanboard will run one by one each database migration and this process can take some time according to your configuration.

To avoid any potential timeout you can initialize the database directly by importing the SQL schema:

```
mysql -u root -p my_database < app/Schema/Sql/mysql.sql
```

The file `app/Schema/Sql/mysql.sql` is a SQL dump that represents the last version of the database.

If you would like to use Unix socket connection, try this:

```
define('DB_HOSTNAME', '127.0.0.1;unix_socket=/var/run/mysqld/mysqld.sock');
```

2.14.3 SSL configuration

These parameters have to be defined to enable the MySQL SSL connection:

```
// MySQL SSL key
define('DB_SSL_KEY', '/path/to/client-key.pem');

// MySQL SSL certificate
define('DB_SSL_CERT', '/path/to/client-cert.pem');

// MySQL SSL CA
define('DB_SSL_CA', '/path/to/ca-cert.pem');
```

2.15 Postgresql

By default, Kanboard use Sqlite to store its data but it's also possible to use Postgresql.

2.15.1 Requirements

- Postgresql >= 9.3

- The PHP extension `pdo_pgsql` installed (Debian/Ubuntu: `apt-get install php5-pgsql`)

2.15.2 Configuration

Create an empty database with the command `pgsql`:

```
CREATE DATABASE kanboard;
```

Create a config file

The file `config.php` should contain those values:

```
<?php

// We choose to use Postgresql instead of Sqlite
define('DB_DRIVER', 'postgres');

// Mysql parameters
define('DB_USERNAME', 'REPLACE_ME');
define('DB_PASSWORD', 'REPLACE_ME');
define('DB_HOSTNAME', 'REPLACE_ME');
define('DB_NAME', 'kanboard');
```

Note: You can also rename the template file `config.default.php` to `config.php`.

Importing SQL dump (alternative method)

For the first time, Kanboard will run one by one each database migration and this process can take some time according to your configuration.

To avoid any issues or potential timeouts, you can initialize the database directly by importing the SQL schema:

```
psql -U postgres my_database < app/Schema/Sql/postgres.sql
```

The file `app/Schema/Sql/postgres.sql` is a SQL dump that represents the last version of the database.

2.16 Cron Job

To work properly, Kanboard requires that a background job run on a daily basis. Usually on Unix platforms, this process is done by `cron`.

This background job is necessary for these features:

- Reports and analytics (calculate daily stats of each projects)
- Send overdue task notifications
- Execute automatic actions connected to the event “Daily background job for tasks”

2.16.1 Configuration on Unix and Linux platforms

There are multiple ways to define a cronjob on Unix/Linux operating systems, this example is for Ubuntu 14.04. The procedure is similar for other systems.

Edit the crontab of your web server user:

```
sudo crontab -u www-data -e
```

Example to execute the daily cronjob at 8am:

```
0 8 * * * cd /path/to/kanboard && ./cli cronjob >/dev/null 2>&1
```

Note: the cronjob process must have write access to the database in case you are using Sqlite. Usually, running the cronjob under the web server user is enough.

2.16.2 Configuration on Microsoft Windows Server

Before to configure the recurring task, create a batch file (*.bat or*.cmd) that run the Kanboard CLI script.

Here is an example (`C:\kanboard.bat`):

```
"C:\php\php.exe" -f "C:\inetpub\wwwroot\kanboard\cli" cronjob
```

You must change the path of the PHP executable and the path of the Kanboard's script according to your installation.

Configure the Windows Task Scheduler:

1. Go to "Administrative Tools"
2. Open the "Task Scheduler"
3. On the right, choose "Create Task"
4. Choose a name, for example you can use "Kanboard"
5. Under "Security Options", choose a user that can write to the database in case you are using Sqlite (might be IIS_IUSRS depending of your configuration)
6. Create a new "Trigger", choose daily and a time, midnight for example
7. Add a new action, choose "Start a program" and select the batch file created above

2.16.3 Configuration by calling a URL

In case your hosting provider doesn't offer CLI access, you can run cron jobs by calling a URL. The URL is protected using the webhook token and should be called via HTTPS for added security.

Cron job URL (with URL rewriting enabled): `https://domain.tld/cronjob?token=WEBHOOK_TOKEN_HERE`

2.17 Email Configuration

2.17.1 User Settings

To receive email notifications, users of Kanboard must have:

- Activated notifications in their profile
- Have a valid email address in their profile
- Be a member of the project that will trigger notifications

Note: The logged user who performs the action doesn't receive any notifications, only other project members.

2.17.2 Email Transports

There are several email transports available:

- SMTP
- Sendmail
- PHP native mail function
- Other methods can be provided by external plugins: Postmark, Sendgrid and Mailgun

2.17.3 Server Settings

By default, Kanboard will use the bundled PHP mail function to send emails. Usually that requires no configuration if your server can already send emails.

However, it's possible to use other methods, the SMTP protocol and Sendmail.

SMTP Configuration

Rename the file `config.default.php` to `config.php` and change these values:

```
// We choose "smtp" as mail transport
define('MAIL_TRANSPORT', 'smtp');

// We define our server settings
define('MAIL_SMTP_HOSTNAME', 'mail.example.com');
define('MAIL_SMTP_PORT', 25);

// Credentials for authentication on the SMTP server (not mandatory)
define('MAIL_SMTP_USERNAME', 'username');
define('MAIL_SMTP_PASSWORD', 'super password');
```

It's also possible to use a secure connection, TLS or SSL:

```
define('MAIL_SMTP_ENCRYPTION', 'ssl'); // Valid values are "null", "ssl" or "tls"
```

Some servers will reject emails based on the hostname transmitted with the HELO (EHLO) command (see RFC 5321). The (fully-qualified) hostname supplied in the HELO command can be set explicitly by:

```
define('MAIL_SMTP_HELO_NAME', null); // valid: null (default), or FQDN
```

Sendmail Configuration

By default the sendmail command will be `/usr/sbin/sendmail -bs` but you can customize that in your config file.

Example:

```
// We choose "sendmail" as mail transport
define('MAIL_TRANSPORT', 'sendmail');

// If you need to change the sendmail command, replace the value
define('MAIL_SENDMAIL_COMMAND', '/usr/sbin/sendmail -bs');
```

PHP built-in mail function

This is the default configuration:

```
define('MAIL_TRANSPORT', 'mail');
```

Sender Email Address

By default, emails will use the sender address `notifications@kanboard.local`. It's not possible to reply to this address.

You can customize this address by changing the value of the constant `MAIL_FROM` in your config file.

```
define('MAIL_FROM', 'kanboard@mydomain.tld');
```

That can be useful if your SMTP server configuration doesn't accept the default address.

2.17.4 Troubleshooting

If no emails are sent and you are sure that everything is configured correctly:

- Check your spam folder
- Enable the debug mode and check the debug file `data/debug.log`, you should see the exact error
- Be sure that your server or your hosting provider allows you to send emails
- If you use SeLinux, allow PHP to send emails

2.18 Background Workers

Warning: This feature is not maintained anymore. Use at your own risk.

Depending on your configuration, some features can slow down the application if they are executed in the same process as the HTTP request. Kanboard can delegate these tasks to a background worker that listen for incoming events.

Example of feature that may slow down Kanboard:

- Sending emails via an external SMTP server can take several seconds
- Sending notifications to external services

This feature is optional and require the installation of a queue daemon on your server.

2.18.1 Beanstalk

Beanstalk is a simple, fast work queue.

- To install Beanstalk, you can simply use the package manager of your Linux distribution
- Install the Kanboard plugin for Beanstalk
- Start the worker with the Kanboard command line tool: `./cli worker`

2.18.2 RabbitMQ

RabbitMQ is a robust messaging system that is more suitable for high-availability infrastructure.

- Follow the official documentation of RabbitMQ for the installation and the configuration
- Install the Kanboard plugin for RabbitMQ
- Start the worker with the Kanboard command line tool: `./cli worker`

Note:

- You should start the Kanboard worker with a process supervisor (systemd, upstart or supervisord)
- The process must have access to the data folder if you store files on the local filesystem or use Sqlite

2.19 Plugin Directory Configuration

To install, update and remove plugins from the user interface, you must have those requirements:

- The plugin directory must be writeable by the web server user
- The Zip extension must be available on your server
- The config parameter `PLUGIN_INSTALLER` must be set to `true`

Only administrators are allowed to install plugins from the user interface.

By default, only plugin listed on Kanboard's website are available. You can change the directory URL in the config file.

Warning: Since Kanboard v1.2.8, the plugin directory is disabled by default for security reasons.

Note:

- There is no code review or any approval process to submit a plugin.
- This is up to the Kanboard instance owner to validate if a plugin is legit.

2.20 Performances

According to your configuration, some features can slow down the usage of Kanboard. By default, all operations are synchronous and performed in the same thread as the HTTP request. This is a PHP limitation. However, it's possible to improve that.

Depending on the plugins you install, communicating to external services can take hundred of milliseconds or even seconds. To avoid blocking the main thread, it's possible to delegate these operations to a pool of background workers. This setup require that you install additional software in your infrastructure.

2.20.1 How to detect the bottleneck?

- Enable the debug mode
- Monitor the log file
- Do something in Kanboard (drag and drop a task for example)
- All operations are logged with the execution time (HTTP requests, Email notifications, SQL requests)

2.20.2 Improve Email notifications speed

Using the SMTP method with an external server can be very slow.

Possible solutions:

- Use the background workers if you still want to use SMTP
- Use a local email relay with Postfix and use the "mail" transport
- Use an email provider that use an HTTP API to send emails (Sendgrid, Mailgun or Postmark)

2.20.3 Improve Sqlite performances

Possible solutions:

- Do not use Sqlite when you have a lot of concurrency (several users), choose Postgres or Mysql instead
- Do not use Sqlite on a shared NFS mount
- Do not use Sqlite on a disk with poor IOPS, it's always preferable to use local SSD drives

2.21 URL Rewriting

Kanboard is able to work indifferently with URL rewriting enabled or not.

- Example of URL rewritten: `/board/123`
- Otherwise: `?controller=board&action=show&project_id=123`

If you use Kanboard with Apache and with the mode rewrite enabled, nice URLs will be used automatically. In case you get a "404 Not Found", you might need to set at least the following overrides for your DocumentRoot to get the .htaccess files working:

```
<Directory /var/www/kanboard/>
    AllowOverride FileInfo Options=All,MultiViews AuthConfig
</Directory>
```

2.21.1 URL Shortcuts

- Go to the task #123: **/t/123**
- Go to the board of the project #2: **/b/2**
- Go to the project calendar #5: **/c/5**
- Go to the list view of the project #8: **/l/8**
- Go to the project settings for the project id #42: **/p/42**

2.21.2 Configuration

By default, Kanboard will check if the Apache mode rewrite is enabled.

To avoid the automatic detection of URL rewriting from the web server, you can enable this feature in your config file:

```
define('ENABLE_URL_REWRITE', true);
```

When this constant is at `true`:

- URLs generated from command line tools will be also converted
- If you use another web server than Apache, for example Nginx or Microsoft IIS, you have to configure yourself the URL rewriting

Note: Kanboard always fallback to old school URLs when it's not configured, this configuration is optional.

2.21.3 Nginx Configuration Example

In the section `server` of your Nginx config file you can use this example:

```
index index.php;

location / {
    try_files $uri $uri/ /index.php$is_args$args;

    # If Kanboard is under a subfolder
    # try_files $uri $uri/ /kanboard/index.php;
}

location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass unix:/var/run/php5-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_index index.php;
    include fastcgi_params;
}

# Deny access to the directory data
```

(continues on next page)

(continued from previous page)

```
location ~* /data {
    deny all;
    return 404;
}

# Deny access to .htaccess
location ~ /\.ht {
    deny all;
    return 404;
}
```

In your Kanboard `config.php`:

```
define('ENABLE_URL_REWRITE', true);
```

Another example with Kanboard in a subfolder:

```
server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    root /var/www/html;
    index index.php index.html index.htm;
    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }

    location ^~ /kanboard {

        location /kanboard {
            try_files $uri $uri/ /kanboard/index.php$is_args$args;
        }

        location ~ ^/kanboard/(?:kanboard|config.php|config.default.php) {
            deny all;
        }

        location ~* /kanboard/data {
            deny all;
        }

        location ~ \.php(?:$|/) {
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
            fastcgi_param HTTPS on; # Use only if HTTPS is configured
            include fastcgi_params;
            fastcgi_pass unix:/var/run/php5-fpm.sock;
        }

        location ~ /kanboard/\.ht {
            deny all;
        }
    }
}
```

Adapt the example above according to your own configuration.

2.21.4 Lighttpd Configuration Example

1. Enable "mod_rewrite"

```
server.modules += (
    "mod_rewrite",
    ...
    ...
)
```

2. Add url rewrites to the relevant sections of your lighttpd.conf (in this case, for host example.com). Also keep the assets directory and the favicon static:

```
$HTTP["host"] == "example.com" {
  server.document-root = "/var/www/kanboard/"
  url.rewrite-once = (
    "^(/[^\?]*)(\?.*)?" => "/index.php$2",
    "^/assets/.+" => "$0",
    "^/favicon\.png$" => "$0",
  )
}
```

3. Reload the Lighttpd config:

```
/etc/init.d/lighttpd reload
```

2.21.5 IIS Configuration Example

1. Download and install the Rewrite module for IIS: Download link
2. Create a web.config in you installation folder:

```
<?xml version="1.0"?>
<configuration>
    <system.webServer>
        <defaultDocument>
            <files>
                <clear />
                <add value="index.php" />
            </files>
        </defaultDocument>
        <rewrite>
            <rules>
                <rule name="Kanboard URL Rewrite" stopProcessing="true">
                    <match url="^(.*)$" ignoreCase="false" />
                    <conditions logicalGrouping="MatchAll">
                        <add input="{REQUEST_FILENAME}" matchType="IsFile" ignoreCase=
↪"false" negate="true" />
                    </conditions>
                    <action type="Rewrite" url="index.php" appendQueryString="true" />
                </rule>
            </rules>
        </rewrite>
```

(continues on next page)

(continued from previous page)

```
    </system.webServer>
</configuration>
```

In your Kanboard `config.php`:

```
define('ENABLE_URL_REWRITE', true);
```

Adapt the example above according to your own configuration.

2.22 LDAP Authentication

2.22.1 Requirements

- PHP LDAP extension enabled
- LDAP server: - OpenLDAP - Microsoft Active Directory - Novell eDirectory

2.22.2 Workflow

When the LDAP authentication is activated, the login process works like that:

1. Try first to authenticate the user by using the database
2. If the user is not found inside the database, a LDAP authentication is performed
3. If the LDAP authentication is successful, by default a local user is created automatically with no password and marked as LDAP users.

The full name and the email address are automatically fetched from the LDAP server.

2.22.3 Authentication Types

Type	Description
Proxy User	A specific user is used to browse LDAP directory
User	The end-user credentials are used for browsing LDAP directory
Anonymous	No authentication is performed for LDAP browsing

Note: The recommended authentication method is `Proxy`.

Anonymous Mode

```
define('LDAP_BIND_TYPE', 'anonymous');
define('LDAP_USERNAME', null);
define('LDAP_PASSWORD', null);
```

This is the default value but some LDAP servers don't allow anonymous browsing for security reasons.

Proxy Mode

A specific user is used to browse the LDAP directory:

```
define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'my proxy user');
define('LDAP_PASSWORD', 'my proxy password');
```

User Mode

This method uses the credentials provided by the end-user.

For example, Microsoft Active Directory doesn't allow anonymous browsing by default and if you don't want to use a proxy user you can use this method.

```
define('LDAP_BIND_TYPE', 'user');
define('LDAP_USERNAME', '%s@kanboard.local');
define('LDAP_PASSWORD', null);
```

In this case, the constant `LDAP_USERNAME` is used as a pattern to the ldap username, examples:

- `%s@kanboard.local` will be replaced by `my_user@kanboard.local`
- `KANBOARD\\%s` will be replaced by `KANBOARD\my_user`

2.22.4 User LDAP filter

The configuration parameter `LDAP_USER_FILTER` is used to find users in LDAP directory.

Examples:

- `(&(objectClass=user)(sAMAccountName=%s))` is replaced by `(&(objectClass=user)(sAMAccountName=my_username))`
- `uid=%s` is replaced by `uid=my_username`

Other examples of filters for Active Directory

Example to filter access to Kanboard:

`(&(objectClass=user)(sAMAccountName=%s)(memberOf=CN=Kanboard Users,CN=Users,DC=kanboard,DC=local))`

This example allows only people members of the group "Kanboard Users" to connect to Kanboard.

2.22.5 Example for Microsoft Active Directory

Let's say we have a domain `KANBOARD` (kanboard.local) and the primary controller is `myserver.kanboard.local`.

First example with proxy mode:

```
<?php

// Enable LDAP authentication (false by default)
define('LDAP_AUTH', true);

```

(continues on next page)

(continued from previous page)

```
define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'administrator@kanboard.local');
define('LDAP_PASSWORD', 'my super secret password');

// LDAP server hostname
define('LDAP_SERVER', 'myserver.kanboard.local');

// LDAP properties
define('LDAP_USER_BASE_DN', 'CN=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', '(&(objectClass=user)(sAMAccountName=%s))');
```

Second example with user mode:

```
<?php

// Enable LDAP authentication (false by default)
define('LDAP_AUTH', true);

define('LDAP_BIND_TYPE', 'user');
define('LDAP_USERNAME', '%s@kanboard.local');
define('LDAP_PASSWORD', null);

// LDAP server hostname
define('LDAP_SERVER', 'myserver.kanboard.local');

// LDAP properties
define('LDAP_USER_BASE_DN', 'CN=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', '(&(objectClass=user)(sAMAccountName=%s))');
```

2.22.6 Example for OpenLDAP

Our LDAP server is `myserver.example.com` and all users are stored under `ou=People,dc=example, dc=com`.

For this example we use the anonymous binding.

```
<?php

// Enable LDAP authentication (false by default)
define('LDAP_AUTH', true);

// LDAP server hostname
define('LDAP_SERVER', 'myserver.example.com');

// LDAP properties
define('LDAP_USER_BASE_DN', 'ou=People,dc=example,dc=com');
define('LDAP_USER_FILTER', 'uid=%s');
```

2.22.7 Example for LDAPS (SSL-encryption)

Some LDAP servers are configured for "LDAPS" connectivity only (on port 636). This is different to TLS, which starts off in cleartext (port 389 by default) and then sets up encryption over the same channel.

To tell PHP to use LDAPS, you need to prefix the name of your LDAP server with "ldaps://", as in the example below:

Our LDAP server is `myserver.example.com` and is only accessible via LDAPS. Most likely we won't want to validate the server cert, and we DON'T want TLS.

For this example we use the anonymous binding.

```
<?php

// Enable LDAP authentication (false by default)
define('LDAP_AUTH', true);

// LDAP server hostname
define('LDAP_SERVER', 'ldaps://myserver.example.com');

// By default, require certificate to be verified for ldaps:// style URL. Set to␣
↪false to skip the verification
define('LDAP_SSL_VERIFY', false);

// Enable LDAP START_TLS
define('LDAP_START_TLS', false);;
```

2.22.8 Disable Automatic Account Creation

By default, Kanboard will create a user account automatically if nothing is found.

You can disable this behavior if you prefer to create user accounts manually to restrict Kanboard to only some people.

Change the value of `LDAP_USER_CREATION` to `false`:

```
// Automatically create user account
define('LDAP_USER_CREATION', false);
```

2.22.9 Synchronization

By default, Kanboard will synchronize all fields (role, name, email…) except the username.

If you would like to change this behavior, use this config parameter:

```
// This example will not synchronize the fields "username" and "role" from LDAP to␣
↪Kanboard.
define('EXTERNAL_AUTH_EXCLUDE_FIELDS', 'username,role');
```

2.22.10 Troubleshooting

SELinux Restrictions

If SELinux is enabled, you have to allow Apache to reach out your LDAP server.

- You can switch SELinux to the permissive mode or disable it (not recommended)
- You can allow all network connections, for example `setsebool -P httpd_can_network_connect=1` or have a more restrictive rule

In any case, refer to the official Redhat/Centos documentation.

Debug Mode

If you are not able to setup correctly the LDAP authentication, you can enable the debug mode and watch log files.

2.23 LDAP User Profile Photo

Kanboard can download automatically user pictures from the LDAP server.

This feature is enabled only if LDAP authentication is activated and the parameter `LDAP_USER_ATTRIBUTE_PHOTO` is defined.

2.23.1 Configuration

In your `config.php`, you have to set the LDAP attribute used to store the image.

```
define('LDAP_USER_ATTRIBUTE_PHOTO', 'jpegPhoto');
```

Usually, the attributes `jpegPhoto` or `thumbnailPhoto` are used. The image can be stored in JPEG or PNG format.

To upload the image in the user profile, Active Directory administrators may use software like AD Photo Edit.

Note: The profile image is **downloaded at login time only if the user do not already have uploaded an image previously**.

To change the user photo, the previous one have to be removed manually in the user profile.

2.24 LDAP Configuration Examples

Note:

- Nested groups are not implemented, send a pull-request if you need this feature.

2.24.1 Microsoft Active Directory

- User authentication
- Download the user profile picture from Active Directory
- Set user language from LDAP attribute
- Kanboard roles are mapped to Active Directory groups
- LDAP group providers is enabled

```
define('LDAP_AUTH', true);

define('LDAP_SERVER', 'my-ldap-server');
define('LDAP_PORT', 389);
```

(continues on next page)

(continued from previous page)

```
define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'administrator@kanboard.local');
define('LDAP_PASSWORD', 'secret');

define('LDAP_USER_BASE_DN', 'CN=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', '(&(objectClass=user)(sAMAccountName=%s))');

define('LDAP_USER_ATTRIBUTE_USERNAME', 'sAMAccountName');
define('LDAP_USER_ATTRIBUTE_FULLNAME', 'displayname');
define('LDAP_USER_ATTRIBUTE_PHOTO', 'jpegPhoto');
define('LDAP_USER_ATTRIBUTE_LANGUAGE', 'preferredLanguage');

define('LDAP_GROUP_ADMIN_DN', 'CN=Kanboard Admins,CN=Users,DC=kanboard,DC=local');
define('LDAP_GROUP_MANAGER_DN', 'CN=Kanboard Managers,CN=Users,DC=kanboard,DC=local');

define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'CN=Users,DC=kanboard,DC=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=group)(sAMAccountName=%s*))');
define('LDAP_GROUP_ATTRIBUTE_NAME', 'cn');
```

2.24.2 OpenLDAP with memberOf overlay

User LDIF example:

```
dn: uid=manager,ou=Users,dc=kanboard,dc=local
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: manager
sn: Lastname
givenName: Firstname
cn: Kanboard Manager
displayName: Kanboard Manager
mail: manager@kanboard.local
userPassword: password
memberOf: cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
```

Group LDIF example:

```
dn: cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
objectClass: top
objectClass: groupOfNames
cn: Kanboard Managers
member: uid=manager,ou=Users,dc=kanboard,dc=local
```

Kanboard Configuration:

- User authentication
- Kanboard roles are mapped to LDAP groups
- LDAP group providers is enabled

```
define('LDAP_AUTH', true);

define('LDAP_SERVER', 'my-ldap-server');
define('LDAP_PORT', 389);

define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'cn=admin,DC=kanboard,DC=local');
define('LDAP_PASSWORD', 'password');

define('LDAP_USER_BASE_DN', 'OU=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', 'uid=%s');

define('LDAP_GROUP_ADMIN_DN', 'cn=Kanboard Admins,ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_MANAGER_DN', 'cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
↪');

define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=groupOfNames)(cn=%s*))');
define('LDAP_GROUP_ATTRIBUTE_NAME', 'cn');
```

2.24.3 OpenLDAP with Posix groups (memberUid)

User LDIF example:

```
dn: uid=manager,ou=Users,dc=kanboard,dc=local
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: manager
sn: Lastname
givenName: Firstname
cn: Kanboard Manager
displayName: Kanboard Manager
uidNumber: 10001
gidNumber: 8000
userPassword: password
homeDirectory: /home/manager
mail: manager@kanboard.local
```

Group LDIF example:

```
dn: cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
objectClass: posixGroup
cn: Kanboard Managers
gidNumber: 5001
memberUid: manager
```

Kanboard Configuration:

- User authentication
- Kanboard roles are mapped to LDAP groups
- LDAP group providers is enabled

```
define('LDAP_AUTH', true);

define('LDAP_SERVER', 'my-ldap-server');
define('LDAP_PORT', 389);

define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'cn=admin,DC=kanboard,DC=local');
define('LDAP_PASSWORD', 'password');

define('LDAP_USER_BASE_DN', 'OU=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', 'uid=%s');

define('LDAP_GROUP_ADMIN_DN', 'cn=Kanboard Admins,ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_MANAGER_DN', 'cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
↪');

// This filter is used to find the groups of our user
define('LDAP_GROUP_USER_FILTER', '(&(objectClass=posixGroup)(memberUid=%s))');

define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=posixGroup)(cn=%s*))');
define('LDAP_GROUP_ATTRIBUTE_NAME', 'cn');
```

2.24.4 OpenLDAP with groupOfNames

User LDIF example:

```
dn: uid=manager,ou=Users,dc=kanboard,dc=local
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: manager
sn: Lastname
givenName: Firstname
cn: Kanboard Manager
displayName: Kanboard Manager
mail: manager@kanboard.local
userPassword: password
```

Group LDIF example:

```
dn: cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
objectClass: top
objectClass: groupOfNames
cn: Kanboard Managers
member: uid=manager,ou=Users,dc=kanboard,dc=local
```

Kanboard Configuration:

- User authentication
- Kanboard roles are mapped to LDAP groups
- LDAP group providers is enabled

```
define('LDAP_AUTH', true);

define('LDAP_SERVER', 'my-ldap-server');
define('LDAP_PORT', 389);

define('LDAP_BIND_TYPE', 'proxy');
define('LDAP_USERNAME', 'cn=admin,DC=kanboard,DC=local');
define('LDAP_PASSWORD', 'password');

define('LDAP_USER_BASE_DN', 'OU=Users,DC=kanboard,DC=local');
define('LDAP_USER_FILTER', 'uid=%s');

define('LDAP_GROUP_ADMIN_DN', 'cn=Kanboard Admins,ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_MANAGER_DN', 'cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
↪');

// This filter is used to find the groups of our user
define('LDAP_GROUP_USER_FILTER', '(&(objectClass=groupOfNames)(member=uid=%s,ou=Users,
↪dc=kanboard,dc=local))');

define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=groupOfNames)(cn=%s*))');
define('LDAP_GROUP_ATTRIBUTE_NAME', 'cn');
```

2.25 LDAP Group Synchronization

Note:

- Nested groups are not implemented, send a pull-request if you need this feature.

2.25.1 Requirements

- Have LDAP authentication properly configured
- Use a LDAP server that supports `memberOf` or `memberUid` (PosixGroups)

2.25.2 Define automatically user roles based on LDAP groups

Use these constants in your config file:

- `LDAP_GROUP_ADMIN_DN`: Distinguished names for application administrators
- `LDAP_GROUP_MANAGER_DN`: Distinguished names for application managers

Example for Active Directory:

```
define('LDAP_GROUP_ADMIN_DN', 'CN=Kanboard Admins,CN=Users,DC=kanboard,DC=local');
define('LDAP_GROUP_MANAGER_DN', 'CN=Kanboard Managers,CN=Users,DC=kanboard,DC=local');
```

- People member of “Kanboard Admins” will have the role “Administrator”

- People member of "Kanboard Managers" will have the role "Managers"
- Everybody else will have the role "User"

Example for OpenLDAP with Posix Groups:

```
define('LDAP_GROUP_BASE_DN', 'ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_USER_FILTER', '(&(objectClass=posixGroup)(memberUid=%s))');
define('LDAP_GROUP_ADMIN_DN', 'cn=Kanboard Admins,ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_MANAGER_DN', 'cn=Kanboard Managers,ou=Groups,dc=kanboard,dc=local
↪');
```

You **must define the parameter** `LDAP_GROUP_USER_FILTER` if your LDAP server use `memberUid` instead of `memberOf`. All parameters of this example are mandatory.

2.25.3 Automatically load LDAP groups for project permissions

This feature allows you to sync automatically LDAP groups with Kanboard groups. Each group can have a different project role assigned.

On the project permissions page, people can enter groups in the auto-complete field and Kanboard can search for groups with any provider enabled.

If the group doesn't exist in the local database, it will be automatically synced.

- `LDAP_GROUP_PROVIDER`: Enable the LDAP group provider
- `LDAP_GROUP_BASE_DN`: Distinguished names to find groups in LDAP directory
- `LDAP_GROUP_FILTER`: LDAP filter used to perform the query
- `LDAP_GROUP_ATTRIBUTE_NAME`: LDAP attribute used to fetch the group name

Example for Active Directory:

```
define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'CN=Groups,DC=kanboard,DC=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=group)(sAMAccountName=%s*))');
```

With the filter given as example above, Kanboard will search for groups that match the query. If the end-user enter the text "My group" in the auto-complete box, Kanboard will return all groups that match the pattern: `(&(objectClass=group)(sAMAccountName=My group*))`.

- Note 1: The special characters `*` is important here, **otherwise an exact match will be done**.
- Note 2: This feature is only compatible with LDAP authentication configured in "proxy" or "anonymous" mode

More examples of LDAP filters for Active Directory

Example for OpenLDAP with Posix Groups:

```
define('LDAP_GROUP_PROVIDER', true);
define('LDAP_GROUP_BASE_DN', 'ou=Groups,dc=kanboard,dc=local');
define('LDAP_GROUP_FILTER', '(&(objectClass=posixGroup)(cn=%s*))');
```

2.26 LDAP Configuration Parameters

Note:

- LDAP attributes must be lowercase.
- Nested groups are not implemented, send a pull-request if you need this feature.

Parameter	Default value	Description
`LDAP_AUTH`	false	Enable LDAP authentication
`LDAP_SERVER`	Empty	LDAP server hostname
`LDAP_PORT`	389	LDAP server port
`LDAP_SSL_VERIFY`	true	Validate certificate for `ldaps://` style URL
`LDAP_START_TLS`	false	Enable LDAP start TLS
`LDAP_USERNAME_CASE_SENSITIVE`	false	Kanboard lowercase the ldap username to avoid duplicate users (the database is case sensitive)
`LDAP_BIND_TYPE`	anonymous	Bind type: “anonymous”, “user” or “proxy”
`LDAP_USERNAME`	null	LDAP username to use with proxy mode or username pattern to use with user mode
`LDAP_PASSWORD`	null	LDAP password to use for proxy mode
`LDAP_USER_BASE_DN`	Empty	LDAP DN for users (Example: “CN=Users,DC=kanboard,DC=local”)
`LDAP_USER_FILTER`	Empty	LDAP pattern to use when searching for a user account (Example: “(&(objectClass=user)(sAMAccount Name=%s))”)
`LDAP_USER_ATTRIBUTE_USERNAME`	uid	LDAP attribute for username (Example: “samaccountname”)
`LDAP_USER_ATTRIBUTE_FULLNAME`	cn	LDAP attribute for user full name (Example: “displayname”)
`LDAP_USER_ATTRIBUTE_EMAIL`	mail	LDAP attribute for user email
`LDAP_USER_ATTRIBUTE_GROUPS`	memberof	LDAP attribute to find groups in user profile
`LDAP_USER_ATTRIBUTE_PHOTO`	Empty	LDAP attribute to find user photo (jpegPhoto or thumbnailPhoto
`LDAP_USER_ATTRIBUTE_LANGUAGE`	Empty	LDAP attribute for user language (preferredlanguage), the accepted language format is “fr-FR”
`LDAP_USER_CREATION`	true	Enable automatic LDAP user creation
`LDAP_GROUP_ADMIN_DN`	Empty	LDAP DN for administrators (Example: “CN=Kanboard-Admins,CN=Users,DC= kanboard,DC=local”)
`LDAP_GROUP_MANAGER_DN`	Empty	LDAP DN for managers (Example: “CN=Kanboard Managers,CN=Users,DC=kanboard,DC =local”)
`LDAP_GROUP_PROVIDER`	false	Enable LDAP group provider for project permissions
`LDAP_GROUP_BASE_DN`	Empty	LDAP Base DN for groups
`LDAP_GROUP_FILTER`	Empty	LDAP group filter (Example: “(&(objectClass=group)(sAMAccoun tName=%s*))“)
`LDAP_GROUP_USER_FILTER`	Empty	If defined, Kanboard will search user groups in LDAP_GROUP_BASE_DN with this filter, it’s useful only for posixGroups (Example: `(&(objectClass=posixGroup)(mem berUid=%s))`)
`LDAP_GROUP_ATTRIBUTE_NAME`	cn	LDAP attribute for the group name
`LDAP_GROUP_USER_ATTRIBUTE`	username	LDAP attribute for the user in the group filter

2.27 Reverse Proxy Authentication

This authentication method is often used for SSO (Single Sign-On) especially for large organizations.

The authentication is done by another system, Kanboard doesn't know your password and suppose you are already authenticated.

2.27.1 Requirements

Apache Auth on the same server or a well-configured reverse proxy.

2.27.2 How does this work?

1. Your reverse proxy authenticates the user and send the username through a HTTP header.
2. Kanboard retrieve the username from the request
    - The user is created automatically if necessary
    - Open a new Kanboard session without any prompt assuming it's valid

2.27.3 Installation instructions

Setting up your reverse proxy

This is not in the scope of this documentation. You should check the user login is sent by the reverse proxy using a HTTP header, and find out which one.

Setting up Kanboard

Create a custom `config.php` file or copy the `config.default.php` file:

```
<?php

// Enable/disable reverse proxy authentication
define('REVERSE_PROXY_AUTH', true); // Set this value to true

// The HTTP header to retrieve. If not specified, REMOTE_USER is the default
define('REVERSE_PROXY_USER_HEADER', 'REMOTE_USER');

// The default Kanboard admin for your organization.
// Since everything should be filtered by the reverse proxy,
// you should want to have a bootstrap admin user.
define('REVERSE_PROXY_DEFAULT_ADMIN', 'myadmin');

// The default domain to assume for the email address.
// In case the username is not an email address, it
// will be updated automatically as USER@mydomain.com
define('REVERSE_PROXY_DEFAULT_DOMAIN', 'mydomain.com');
```

Note:

- If the proxy is the same web server that runs Kanboard, according the CGI protocol the header name will be `REMOTE_USER`. For example, Apache add `REMOTE_USER` by default if `Require valid-user` is set.
- If you use a different header for `REVERSE_PROXY_USER_HEADER`, the value must be prefixed by `HTTP_` because it's fetched from the `$_SERVER` array.
- If Apache is a reverse proxy to another Apache running Kanboard, the header `REMOTE_USER` is not set (same behavior with IIS and Nginx).
- If you have a real reverse proxy, the HTTP ICAP draft proposes the header to be `X-Authenticated-User`. This de facto standard has been adopted by a number of tools.

2.28 Security

2.28.1 General Advices

- Do not forget to change the default user/password.
- Do not allow everybody to access to the directory `data` from the URL. A `.htaccess` file for Apache and a `web.config` file for IIS is included but other web servers have to be configured manually.

Note:

- Plugins installation from the web user interface is disabled by default.
- There is no code review or any approval process to submit a plugin.
- This is up to the Kanboard instance owner to check if a plugin is legit.

2.28.2 Installation Outside Document Root

If you would like to install Kanboard outside of the web server document root, you need to create at least these symlinks:

```
.
├── assets -> ../kanboard/assets
├── cli -> ../kanboard/cli
├── favicon.ico -> ../kanboard/favicon.ico
├── index.php -> ../kanboard/index.php
├── jsonrpc.php -> ../kanboard/jsonrpc.php
└── robots.txt -> ../kanboard/robots.txt
```

The `.htaccess` is optional because its content can be included directly in the Apache configuration.

You can also define a custom location for the plugins and files folders by changing the config file.

Note: Some plugins may requires to be installed into the document root.

2.28.3 Brute Force Protection

The brute force protection of Kanboard works at the user account level:

- After 3 authentication failure for the same username, the login form shows a captcha image to prevent automated bot tentatives.
- After 6 authentication failure, the user account is locked for a period of 15 minutes.

This feature works only for authentication methods that use the login form.

However, **after three authentication failure through the user API**, the account has to be unlocked by using the login form.

Kanboard doesn't block any IP addresses since bots can use several anonymous proxies. However, you can use external tools like fail2ban to avoid massive scans.

Default settings can be changed with these configuration variables:

```
// Enable captcha after 3 authentication failure
define('BRUTEFORCE_CAPTCHA', 3);

// Lock the account after 6 authentication failure
define('BRUTEFORCE_LOCKDOWN', 6);

// Lock account duration in minutes
define('BRUTEFORCE_LOCKDOWN_DURATION', 15);
```

If you don't want to wait 15 minutes, you can unlock a user from the user interface. As administrator, go to the user profile and click on "Unlock this user".

2.29 Frequently Asked Questions

2.29.1 Can you recommend a web hosting provider for Kanboard?

Kanboard works well with any great VPS hosting provider such as Digital Ocean, Linode or Gandi.

To have the best performances, choose a provider with fast disk I/O because Kanboard use Sqlite by default. Avoid hosting providers that use a shared NFS mount point.

Note: Using a shared hosting provider is not recommended. Use your own server.

2.29.2 How to display a link to the task in notifications?

To do that, you have to specify the URL of your Kanboard installation in your Application Settings. By default, nothing is defined, so no links will be displayed.

Examples:

- http://myserver/kanboard/
- http://kanboard.mydomain.com/

Don't forget the ending slash `/`.

You need to define that manually because Kanboard cannot guess the URL from a command line script and some people have a very specific configuration.

2.29.3 I have the error "There is no suitable CSPRNG installed on your system"

If you use PHP < 7.0, you need to have the openssl extension enabled or `/dev/urandom` accessible from the application if restricted by an `open_basedir` restriction.

2.29.4 Page not found and the URL seems wrong (&amp;)

- The URL looks like `/?controller=auth&amp;action=login&amp;redirect_query=` instead of `?controller=auth&action=login&redirect_query=`
- Kanboard returns a "Page not found" error

This issue comes from your PHP configuration, the value of `arg_separator.output` is not the PHP's default, there is different ways to fix that:

Change the value directly in your `php.ini` if you can:

```
arg_separator.output = "&"
```

Override the value with a `.htaccess`:

```
php_value arg_separator.output "&"
```

Otherwise Kanboard will try to override the value directly in PHP.

2.29.5 Authentication failure with the API and Apache + PHP-FPM

php-cgi under Apache does not pass HTTP Basic user/pass to PHP by default. For this workaround to work, add these lines to your `.htaccess` file:

```
RewriteCond %{HTTP:Authorization} ^(.+)$
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

2.29.6 How to test Kanboard with the PHP built-in web server?

If you don't want to install a web server like Apache on localhost. You can test with the embedded web server of PHP:

```
unzip kanboard-VERSION.zip
cd kanboard
php -S localhost:8000
open http://localhost:8000/
```

2.29.7 I get a blank page after installing or upgrading Kanboard

- Check if you have installed all requirements on your server
- Check the PHP and Apache error logs
- Check if the files have the correct permissions
- If you use an aggressive OPcode caching, reload your web-server or php-fpm

2.29.8 Solving Database Migration Issues

- SQL migrations are executed automatically when you upgrade Kanboard to a new version
- For Postgres and Mysql, the current schema version number is stored in the table `schema_version` and for Sqlite this is stored in the variable `user_version`
- Migrations are defined in the file `app/Schema/<DatabaseType>.php`
- Each function is a migration
- Each migration is executed in a transaction
- If migration generate an error, a rollback is performed

When upgrading:

- Always backup your data
- Do not run migrations in parallel from multiple processes

If you got the error "Unable to run SQL migrations [. . .]", here are the steps to fix it manually:

1. Open the file corresponding to your database `app/Schema/Sqlite.php` or `app/Schema/Mysql.php`
2. Go to the failed migration function
3. Execute manually the SQL queries defined in the function
4. If you encounter an error, report the issue to the bug tracker with the exact SQL error
5. When all SQL statements of the migration are executed, update the schema version number
6. Run other migrations

2.29.9 I'm not able to login with Internet Explorer and Microsoft IIS

If you are not able to login and always get the error **"Username or password required"** even if you have entered the right credentials, that means there is a problem with the session.

For example, this is a known issue if you meet these criteria:

- You are using a domain name with an underscore: `kanboard_something.mycompany.tld`
- You are using Microsoft Windows Server and IIS
- Your browser is Internet Explorer

Solution: **Do not use underscore in the domain name because this is not a valid domain name**.

Explanation: Internet Explorer doesn't accept cookies with a domain name with underscores because it's not valid.

Reference:

- https://support.microsoft.com/en-us/kb/316112

2.29.10 How to change attachment size limit?

The file upload size is not defined by Kanboard itself but by PHP and your webserver.

In your `php.ini`, change the following lines:

```
# Set size limit to 20MB
upload_max_filesize = 20M
post_max_size = 20M
```

If you use Nginx, define this value:

```
client_max_body_size 20M;
```

See http://nginx.org/en/docs/http/ngx_http_core_module.html#client_max_body_size.

2.29.11 Is it possible to customize table names prefix?

Short answer: No.

- Kanboard is designed to use its own database.
- Changing existing code will require too many changes.
- Mixing multiple software into the same database is a bad practice (shared hosting providers are not recommended).

2.29.12 Why there is no official native mobile application?

The development of a native mobile application is let to the community.

- Developing a native mobile application for each platform (iOS/Android) for each device type (Smartphone/Tablet) require a lot of work and money.
- That require different skill set than developing a web application.
- To develop a quality application, you have to use the official SDK of each platform. So, you end up developing twice the same application.
- Publishing a mobile application on a store (App Store/Play Store) is not free, you have to pay, even if your software is free.
- The web user interface is responsive, this is not prefect but that allows you to quickly check something.
- This is not really practical to use the board on a tiny screen.

CHAPTER 3

Developer's Guide

Contributing to the open source project.

3.1 Contributor Guidelines

3.1.1 How can I help?

Kanboard is not perfect but there are many ways to help:

- Report bugs
- Add or update translations
- Improve the documentation
- Fix bugs

3.1.2 I want to report a bug

Be sure to read the instructions

3.1.3 I want to translate Kanboard

Kanboard is translated in many languages. You could improve existing translations or add new ones.

Read the instructions

3.1.4 I want to improve the documentation

You think something is not clear, there is grammatical errors, typos, anything.

Read the instructions

3.1.5 I want to contribute to the code

Pull-requests are always welcome. However, to be accepted you have to follow those directives:

- **Test your changes!** Do not introduce regressions.
- Before doing any large changes, open a new ticket to start a discussion.
- If you want to add a new feature, respect the philosophy behind Kanboard. **We focus on simplicity**, we don't want to have a bloated software.
- The same apply for the user interface, **simplicity and efficiency**.
- Send only one pull-request for each feature or bug fix.
- A smaller pull-request is easier to review and it will be merged faster.
- Make sure the unit tests pass.
- Respect the coding standards.
- Write maintainable code, avoid code duplication, use good practices.
- Avoid introducing new dependencies.

3.2 Translations

3.2.1 How to translate Kanboard to a new language?

- Translations are stored inside the directory `app/Locale`
- There is a subdirectory for each language, for example in French we have `fr_FR`, Italian `it_IT` etc.
- A translation is a PHP file that returns an Array with a key-value pairs
- The key is the original text in English and the value is the translation of the corresponding language
- **French translations are always up to date**
- Always use the last version (master branch)

Create a new translation:

1. Make a new directory: `app/Locale/xx_XX` for example `app/Locale/fr_CA` for French Canadian
2. Create a new file for the translation: `app/Locale/xx_XX/translations.php`
3. Use the content of the French locales and replace the values
4. Update the file `app/Model/Language.php`
5. Check with your local installation of Kanboard if everything is OK
6. Send a pull-request with Github

3.2.2 How to update an existing translation?

1. Open the translation file `app/Locale/xx_XX/translations.php`
2. Missing translations are commented with `//` and the values are empty, just fill blank and remove the comment
3. Check with your local installation of Kanboard and send a pull-request

3.2.3 How to add new translated text in the application?

Translations are displayed with the following functions in the source code:

- `t()`: display text with HTML escaping
- `e()`: display text without HTML escaping

Always use the english version in the source code.

Text strings use the function `sprintf()` to replace elements:

- `%s` is used to replace a string
- `%d` is used to replace an integer

All formats are available in the PHP documentation.

3.2.4 How to find missing translations in the applications?

From a terminal, run the following command:

```
./cli locale:compare
```

All missing and unused translations are displayed on the screen. Put that in the French locale and sync other locales (see below).

3.2.5 How to synchronize translation files?

From a Unix shell run this command:

```
./cli locale:sync
```

The French translation is used a reference to other locales.

3.3 Building Assets (Javascript and CSS files)

Stylesheet and Javascript files are merged together and minified.

- Original CSS files are stored in the folder `assets/css/src/*.css`
- Original Javascript code is stored in the folder `assets/js/src/*.js`

3.3.1 Requirements

- PHP
- Local checkout of the Git repository

3.3.2 Instructions

Build Javascript:

```
./cli js
```

Build stylesheets:

```
./cli css
```

Note:

- Building assets is not possible from the Kanboard's archive, you have to clone the repository.
- Since Kanboard v1.2.11, the dependency on nodejs, Sass, and Gulp has been removed.

3.4 Coding Standards

3.4.1 PHP code

- Indentation: 4 spaces
- Line return: Unix => `\n`
- Encoding: UTF-8
- Use only the opening tags `<?php` or `<?=` for templates, but **never** use `<?`
- Always write PHPdoc comments for methods and class properties
- Coding style: PSR-1 and PSR-2

3.4.2 JavaScript code

- Indentation: 4 spaces
- Line return: Unix => `\n`

3.4.3 CSS code

- Indentation: 4 spaces
- Line return: Unix => `\n`

3.5 Automated Tests

PHPUnit is used to run automated tests on Kanboard.

You can run tests across different databases (Sqlite, Mysql and Postgresql) to be sure that the result is the same everywhere.

3.5.1 Requirements

- Linux/Unix machine
- PHP
- PHPUnit installed
- Mysql and Postgresql (optional)
- Selenium (optional)
- Firefox (optional)

3.5.2 Unit Tests

Test with Sqlite

Sqlite tests use a in-memory database, nothing is written on the file system.

The PHPUnit config file is `tests/units.sqlite.xml`. From your Kanboard directory, run the command `phpunit -c tests/units.sqlite.xml`.

Example:

```
phpunit -c tests/units.sqlite.xml

PHPUnit 5.0.0 by Sebastian Bergmann and contributors.

...............................................................  63 / 649 (  9%)
............................................................... 126 / 649 ( 19%)
............................................................... 189 / 649 ( 29%)
............................................................... 252 / 649 ( 38%)
............................................................... 315 / 649 ( 48%)
............................................................... 378 / 649 ( 58%)
............................................................... 441 / 649 ( 67%)
............................................................... 504 / 649 ( 77%)
............................................................... 567 / 649 ( 87%)
............................................................... 630 / 649 ( 97%)
...................                                             649 / 649 (100%)

Time: 1.22 minutes, Memory: 151.25Mb

OK (649 tests, 43595 assertions)
```

Test with Mysql

You must have Mysql or MariaDb installed on localhost.

By default, those credentials are used:

- Hostname: **localhost**
- Username: **root**
- Password: none
- Database: **kanboard_unit_test**

For each execution the database is dropped and created again.

The PHPUnit config file is `tests/units.mysql.xml`. From your Kanboard directory, run the command `phpunit -c tests/units.mysql.xml`.

Test with Postgresql

You must have Postgresql installed on localhost.

By default, those credentials are used:

- Hostname: **localhost**
- Username: **postgres**
- Password: none
- Database: **kanboard_unit_test**

Be sure to allow the user `postgres` to create and drop databases. The database is recreated for each execution.

The PHPUnit config file is `tests/units.postgres.xml`. From your Kanboard directory, run the command `phpunit -c tests/units.postgres.xml`.

3.5.3 Integration Tests

Integration tests are mainly used to test the API. The test suites are making real HTTP calls to the application that run inside a container.

Requirements

- PHP
- Composer
- Unix operating system (Mac OS or Linux)
- Docker
- Docker Compose

Running integration tests

Integration tests are using Docker containers. There are 3 different environment available to run tests against each supported database.

You can use these commands to run each test suite:

```
# Run tests with Sqlite
make integration-test-sqlite

# Run tests with Mysql
make integration-test-mysql

# Run tests with Postgres
make integration-test-postgres
```

3.5.4 Acceptance Tests

Acceptance tests (also sometimes known as end-to-end tests, and functional tests) test the actual functionality of the UI in a browser using Selenium.

In order to run these tests you must have Selenium Standalone Server installed, and a compatible version of Firefox.

The PHPUnit config file is `tests/acceptance.xml`. With Selenium and the Kanboard app running, from your Kanboard directory, run the command `make test-browser`. This will initiate the testing suite and you will see Firefox open automatically and perform the actions specified in the acceptance tests.

Example:

```
$ make test-browser
PHPUnit 4.8.26 by Sebastian Bergmann and contributors.

..

Time: 5.59 seconds, Memory: 5.25MB

OK (2 tests, 5 assertions)
```

3.5.5 Continuous Integration with Travis-CI

After each commit pushed on the main repository, unit tests are executed across various versions of PHP:

- PHP 7.4
- PHP 7.3
- PHP 7.2

Each version of PHP is tested against the 3 supported database: Sqlite, Mysql and Postgresql.

The Travis config file `.travis.yml` is located on the root directory of Kanboard.

3.6 Webhooks

Webhooks are useful to perform actions with external applications.

- Webhooks can be used to create a task by calling a simple URL (You can also do that with the API)
- An external URL can be called automatically when an event occurs in Kanboard (task creation, comment updated, etc)

3.6.1 How to write a web hook receiver?

All internal events of Kanboard can be sent to an external URL.

- The web hook URL has to be defined in **Settings > Webhooks > Webhook URL**.
- When an event is triggered Kanboard calls the pre-defined URL automatically
- The data are encoded in JSON format and sent with a POST HTTP request
- The web hook token is also sent as a query string parameter, so you can check if the request really comes from Kanboard.

- **Your custom URL must answer in less than 1 second**, those requests are synchronous (PHP limitation) and that can slow down the user interface if your script is too slow!

List of supported events

- comment.create
- comment.update
- comment.delete
- file.create
- task.move.project
- task.move.column
- task.move.position
- task.move.swimlane
- task.update
- task.create
- task.close
- task.open
- task.assignee_change
- subtask.update
- subtask.create
- subtask.delete
- task_internal_link.create_update
- task_internal_link.delete

Example of HTTP request

```
POST https://your_webhook_url/?token=WEBHOOK_TOKEN_HERE
User-Agent: Kanboard Webhook
Content-Type: application/json
Connection: close

{
    "event_name": "task.move.column",
    "event_data": {
        "task_id": "4",
        "task": {
            "id": "4",
            "reference": "",
            "title": "My task",
            "description": "",
            "date_creation": "1469314356",
            "date_completed": null,
            "date_modification": "1469315422",
            "date_due": "1469491200",
```

(continues on next page)

(continued from previous page)

```
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "green",
            "project_id": "1",
            "column_id": "1",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "0",
            "category_id": "0",
            "priority": "0",
            "swimlane_id": "0",
            "date_moved": "1469315422",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Backlog",
            "assignee_username": "admin",
            "assignee_name": null,
            "creator_username": "admin",
            "creator_name": null
        },
        "changes": {
            "src_column_id": "2",
            "dst_column_id": "1",
            "date_moved": "1469315398"
        },
        "project_id": "1",
        "position": 1,
        "column_id": "1",
        "swimlane_id": "0",
        "src_column_id": "2",
        "dst_column_id": "1",
        "date_moved": "1469315398",
        "recurrence_status": "0",
        "recurrence_trigger": "0"
    }
}
```

All event payloads are in the following format:

```
{
  "event_name": "model.event_name",
  "event_data": {
    "key1": "value1",
    "key2": "value2"
  }
```

(continues on next page)

(continued from previous page)

```
}
```

The `event_data` values are not necessary normalized across events.

Examples of event payloads

Task creation:

```
{
    "event_name": "task.create",
    "event_data": {
        "task_id": 5,
        "task": {
            "id": "5",
            "reference": "",
            "title": "My new task",
            "description": "",
            "date_creation": "1469315481",
            "date_completed": null,
            "date_modification": "1469315481",
            "date_due": "0",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "orange",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "3",
            "category_id": "0",
            "priority": "2",
            "swimlane_id": "0",
            "date_moved": "1469315481",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Ready",
            "assignee_username": "admin",
            "assignee_name": null,
            "creator_username": "admin",
            "creator_name": null
        }
    }
}
```

Task modification:

```
{
    "event_name": "task.update",
    "event_data": {
        "task_id": "5",
        "task": {
            "id": "5",
            "reference": "",
            "title": "My new task",
            "description": "New description",
            "date_creation": "1469315481",
            "date_completed": null,
            "date_modification": "1469315531",
            "date_due": "1469836800",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "purple",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "3",
            "category_id": "0",
            "priority": "2",
            "swimlane_id": "0",
            "date_moved": "1469315481",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Ready",
            "assignee_username": "admin",
            "assignee_name": null,
            "creator_username": "admin",
            "creator_name": null
        },
        "changes": {
            "description": "New description",
            "color_id": "purple",
            "date_due": 1469836800
        }
    }
}
```

Task update events have a field called `changes` that contains updated values.

Comment creation:

```
{
    "event_name": "comment.create",
    "event_data": {
        "comment": {
            "id": "1",
            "task_id": "5",
            "user_id": "1",
            "date_creation": "1469315727",
            "comment": "My comment.",
            "reference": null,
            "username": "admin",
            "name": null,
            "email": null,
            "avatar_path": null
        },
        "task": {
            "id": "5",
            "reference": "",
            "title": "My new task",
            "description": "New description",
            "date_creation": "1469315481",
            "date_completed": null,
            "date_modification": "1469315531",
            "date_due": "1469836800",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "purple",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "3",
            "category_id": "0",
            "priority": "2",
            "swimlane_id": "0",
            "date_moved": "1469315481",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Ready",
            "assignee_username": "admin",
            "assignee_name": null,
            "creator_username": "admin",
            "creator_name": null
        }
    }
}
```

Subtask creation:

```
{
    "event_name": "subtask.create",
    "event_data": {
        "subtask": {
            "id": "1",
            "title": "My subtask",
            "status": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "task_id": "5",
            "user_id": "1",
            "position": "1",
            "username": "admin",
            "name": null,
            "timer_start_date": 0,
            "status_name": "Todo",
            "is_timer_started": false
        },
        "task": {
            "id": "5",
            "reference": "",
            "title": "My new task",
            "description": "New description",
            "date_creation": "1469315481",
            "date_completed": null,
            "date_modification": "1469315531",
            "date_due": "1469836800",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "purple",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "3",
            "category_id": "0",
            "priority": "2",
            "swimlane_id": "0",
            "date_moved": "1469315481",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Ready",
            "assignee_username": "admin",
            "assignee_name": null,
```

(continues on next page)

(continued from previous page)

```
            "creator_username": "admin",
            "creator_name": null
        }
    }
}
```

File upload:

```
{
    "event_name": "task.file.create",
    "event_data": {
        "file": {
            "id": "1",
            "name": "kanboard-latest.zip",
            "path": "tasks/5/6f32893e467e76671965b1ec58c06a2440823752",
            "is_image": "0",
            "task_id": "5",
            "date": "1469315613",
            "user_id": "1",
            "size": "4907308"
        },
        "task": {
            "id": "5",
            "reference": "",
            "title": "My new task",
            "description": "New description",
            "date_creation": "1469315481",
            "date_completed": null,
            "date_modification": "1469315531",
            "date_due": "1469836800",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "purple",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "3",
            "category_id": "0",
            "priority": "2",
            "swimlane_id": "0",
            "date_moved": "1469315481",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Ready",
```

(continues on next page)

(continued from previous page)

```
                "assignee_username": "admin",
                "assignee_name": null,
                "creator_username": "admin",
                "creator_name": null
            }
        }
}
```

Task link creation:

```
{
    "event_name": "task_internal_link.create_update",
    "event_data": {
        "task_link": {
            "id": "2",
            "opposite_task_id": "5",
            "task_id": "4",
            "link_id": "3",
            "label": "is blocked by",
            "opposite_link_id": "2"
        },
        "task": {
            "id": "4",
            "reference": "",
            "title": "My task",
            "description": "",
            "date_creation": "1469314356",
            "date_completed": null,
            "date_modification": "1469315422",
            "date_due": "1469491200",
            "date_started": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "color_id": "green",
            "project_id": "1",
            "column_id": "1",
            "owner_id": "1",
            "creator_id": "1",
            "position": "1",
            "is_active": "1",
            "score": "0",
            "category_id": "0",
            "priority": "0",
            "swimlane_id": "0",
            "date_moved": "1469315422",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "category_name": null,
            "swimlane_name": null,
            "project_name": "Demo Project",
            "default_swimlane": "Default swimlane",
            "column_title": "Backlog",
```

(continues on next page)

(continued from previous page)

```
            "assignee_username": "admin",
            "assignee_name": null,
            "creator_username": "admin",
            "creator_name": null
        }
    }
}
```

3.7 Using Vagrant

You can try Kanboard with Vagrant very easily:

- Clone the project from the git repository
- Execute `vagrant plugin install vagrant-vbguest` to install guest additions
- Execute `vagrant up`
- You can access to the application by using the URL `http://localhost:8001/`

The virtual machine is based on Ubuntu 20.04 and PHP 7.4.

3.8 Frequently Asked Questions

3.8.1 Why are you not developing my feature request?

Kanboard is an open source project with limited resources.

- Developing and maintaining a software takes a lot of time.
- Do not under estimate the complexity of introducing changes.
- This is a free and open source project, no one owes you anything.
- If you miss something, contribute to the project.
- Do not expect anyone to work for free.
- People are not going to spend days and weeks of their time to develop a feature just for you.
- No one manages projects in the same way, this is not possible to satisfy the workflow of everyone.
- The number of features is voluntarily limited. Nobody likes bloatware.
- Improving existing features is more important than adding new ones.

3.8.2 Why do you close inactive issues automatically?

- If nobody manifested any interest to develop your feature request, then there is no point of keeping it open.
- Keeping issues open indefinitely will not get fixed by itself.
- Stale issues create more noise.

3.8.3 Why did you close my question on the bug tracker?

- The bug tracker should be reserved only for bug reports.
- Bug triage takes a lot of time.
- If you have a question or if you need help, go to the forum.

3.8.4 How to make a bug report?

You should make sure that you give all information to be able to reproduce the problem.

1. Check for duplicates before creating a new issue.
2. Write in English even if you don't speak English.
3. **Describe your environment:**
    - Operating System
    - Browser
    - Database
    - Version of PHP
    - Version of Kanboard
4. **Describe the actual behavior:**
    - Add screenshots
    - Attach log files
    - Avoid ambiguity, be explicit
5. List all the steps to reproduce the problem.
6. Describe what you expect.

Note: Do not ask questions on the bug tracker, use the forum.

3.8.5 How to add a new plugin to the website?

Follow these instructions: https://github.com/kanboard/website#how-to-add-a-new-plugin-to-the-list

3.8.6 How to update this documentation?

- The documentation source code is available here: https://github.com/kanboard/documentation.
- We use Sphinx and the reStructuredText markup language to generate this documentation in multiple formats.
- To update this documentation, send a pull-request to the project mentioned above.

3.8.7 How to translate the documentation?

Each language has its own repository:

- Czech
- French
- Portuguese
- Russian
- Spanish
- Turkish

To update a translation, send a pull-request to the corresponding project. The directory layout and the file names must be the same as the english version.

If you would like to create a new translation, follow these steps:

- Create a new repository
- Run `sphinx-quickstart`
- Translate the documents
- Push your changes to GitHub
- Contact the maintainers on the forum to add your translation to the list

3.8.8 Why minified files are committed into the source tree?

- This is to simplify Kanboard release process.
- People can download the archive directly from GitHub.
- Occasional contributors can checkout the source code and work on a patch without having to worry about all Javascript dependencies.

3.8.9 Why PHP vendor directory is committed into the source tree?

- This is to simplify Kanboard release process.
- People can download the archive directly from GitHub.
- Occasional contributors can checkout the source code and work on a patch without having to worry about all Composer dependencies.

CHAPTER 4

Plugin Development

4.1 Plugin Registration

4.1.1 Project skeleton generator

You can use `cookiecutter` to create the project structure of your plugin automatically.

Install Cookiecutter:

```
pip install -U cookiecutter
```

Run Kanboard cookiecutter:

```
cookiecutter gh:kanboard/cookiecutter-plugin
plugin_name [My Plugin]: Some Plugin
plugin_namespace [MyPlugin]: SomePlugin
plugin_author [Plugin Author]: Me
plugin_description [My plugin is awesome]:
plugin_homepage [https://github.com/kanboard/plugin-myplugin]:
```

4.1.2 Directory structure

Plugins are stored in the `plugins` subdirectory. An example of a plugin directory structure:

```
plugins
└── Budget             <= Plugin name
    ├── Asset          <= Javascript/CSS files
    ├── Controller
    ├── LICENSE        <= Plugin license
    ├── Locale
    │   ├── fr_FR
    │   ├── it_IT
```

(continues on next page)

(continued from previous page)

```
│       ├── ja_JP
│       └── zh_CN
├── Model
├── Plugin.php      <= Plugin registration file
├── README.md
├── Schema          <= Database migrations
├── Template
└── Test            <= Unit tests
```

Only the registration file `Plugin.php` is required. Other folders are optional.

The first letter of the plugin name must be capitalized.

4.1.3 Plugin Registration File

Kanboard will scan the directory `plugins` and load automatically everything under this directory. The file `Plugin.php` is used to load and register the plugin.

Example of `Plugin.php` file (`plugins/Foobar/Plugin.php`):

```
<?php

namespace Kanboard\Plugin\Foobar;

use Kanboard\Core\Plugin\Base;

class Plugin extends Base
{
    public function initialize()
    {
        $this->template->hook->attach('template:layout:head', 'theme:layout/head');
    }

    public function getCompatibleVersion()
    {
        // Examples:
        // >=1.0.37
        // <1.0.37
        // <=1.0.37
        return '1.0.37';
    }
}
```

This file should contain a class `Plugin` defined under the namespace `Kanboard\Plugin\Yourplugin` and extends `Kanboard\Core\Plugin\Base`.

The only required method is `initialize()`. This method is called for each request when the plugin is loaded.

4.1.4 Plugin Methods

Available methods from `Kanboard\Core\Plugin\Base`:

- `initialize()`: Executed when the plugin is loaded
- `getClasses()`: Return all classes that should be stored in the dependency injection container
- `on($event, $callback)`: Listen on internal events

- `getPluginName()`: Should return plugin name (must match plugins.json `"title":` entry for "Plugin Directory" version update notifications to work)
- `getPluginAuthor()`: Should return plugin author
- `getPluginVersion()`: Should return plugin version
- `getPluginDescription()`: Should return plugin description
- `getPluginHomepage()`: Should return plugin Homepage (link)
- `setContentSecurityPolicy(array $rules)`: Override default HTTP CSP rules
- `onStartup()`: If present, this method is executed automatically when the event "app.bootstrap" is triggered
- `getCompatibleVersion()`: You may want to specify the Kanboard version compatible with the plugin

Your plugin registration class can also inherit from `Kanboard\Core\Base`, that way you can access all classes and methods of Kanboard easily.

This example will fetch the user #123:

```
$this->user->getById(123);
```

4.1.5 Plugin Translations

Plugin can be translated in the same way as the rest of the application. You must load the translations yourself when the session is created:

```
public function onStartup()
{
    Translator::load($this->languageModel->getCurrentLanguage(), __DIR__.'/Locale');
}
```

The translations must be stored in the file `plugins/Myplugin/Locale/xx_XX/translations.php` (replace xx_XX by the language code fr_FR, en_US…).

Translations are stored in a dictionary, if you would like to override an existing string, you just need to use the same key in your translation file.

4.1.6 Dependency Injection Container

Kanboard uses Pimple, a simple PHP Dependency Injection Container. However, Kanboard can register any class in the container easily.

Those classes are available everywhere in the application and only one instance is created.

Here an example to register your own models in the container:

```
public function getClasses()
{
    return array(
        'Plugin\Budget\Model' => array(
            'HourlyRateModel',
            'BudgetModel',
        )
    );
}
```

Now, if you use a class that extends from `Core\Base`, you can access directly to those class instance:

```
$this->hourlyRateModel->remove(123);
$this->budgetModel->getDailyBudgetBreakdown(456);

// It's the same thing as using the container:
$this->container['hourlyRateModel']->getAll();
```

Keys of the containers are unique across the application. If you override an existing class, you will change the default behavior.

4.2 Plugin Hooks

4.2.1 Add new API methods

Kanboard uses this library JSON-RPC to handle API calls.

To add a new method, you can do something like this from your plugin:

```
$this->api->getProcedureHandler()->withCallback('my_method', function() {
    return 'foobar';
});
```

`$this->container['api']` or `$this->api` expose an instance of the object `JsonRPC\Server`.

Read the library documentation for more information.

4.2.2 Add new console commands

Kanboard uses the library Symfony Console to handle local command lines.

Kanboard exposes an instance of the object `Symfony\Component\Console\Application` via `$this->cli`. You can add new commands from your plugin:

```
$this->cli->add(new MyCommand());
```

Read the library documentation for more information.

4.2.3 Add new task filters

Since the task lexer is a factory that returns a new instance each time, you have to extend the `taskLexer` container with the method `extend()` of Pimple.

Here is an example:

```
public function initialize()
{
    $this->container->extend('taskLexer', function($taskLexer, $c) {
        $taskLexer->withFilter(TaskBoardDateFilter::getInstance($c)->setDateParser($c[
↪'dateParser']));
        return $taskLexer;
    });
}
```

For the filter class implementation, there are several examples in the source code under the namespace `Kanboard\Filter`.

4.2.4 Application Hooks

Hooks can extend, replace, filter data or change the default behavior. Each hook is identified with a unique name, example: `controller:calendar:user:events`

Listen on hook events

In your `initialize()` method you need to call the method `on()` of the class `Kanboard\Core\Plugin\Hook`:

```
$this->hook->on('hook_name', $callable);
```

The first argument is the name of the hook and the second is a PHP callable.

Hooks executed only once

Some hooks can have only one listener: `model:subtask-time-tracking:calculate:time-spent`

- Override time spent calculation when sub-task timer is stopped
- Arguments:
  - `$user_id` (integer)
  - `$start` (DateTime)
  - `$end` (DateTime)

4.2.5 Merge hooks

"Merge hooks" act in the same way as the function `array_merge`. The hook callback must return an array. This array will be merged with the default one.

Example to add events in the user calendar:

```
class Plugin extends Base
{
    public function initialize()
    {
        $container = $this->container;

        $this->hook->on('controller:calendar:user:events', function($user_id, $start,
↪$end) use ($container) {
            $model = new SubtaskForecast($container);
            return $model->getCalendarEvents($user_id, $end); // Return new events
        });
    }
}
```

Example to override default values for task forms:

```
class Plugin extends Base
{
    public function initialize()
    {
        $this->hook->on('controller:task:form:default', function (array $default_
↪values) {
            return empty($default_values['score']) ? array('score' => 4) : array();
        });
    }
}
```

List of merging hooks:

`controller:task:form:default`

- Override default values for task forms
- Arguments:
  - `$default_values`: actual default values (array)

`controller:calendar:project:events`

- Add more events to the project calendar
- Arguments:
  - `$project_id` (integer)
  - `$start` Calendar start date (string, ISO-8601 format)
  - `$end` Calendar end date (string, ISO-8601 format)

`controller:calendar:user:events`

- Add more events to the user calendar
- Arguments:
  - `$user_id` (integer)
  - `$start` Calendar start date (string, ISO-8601 format)
  - `$end` Calendar end date (string, ISO-8601 format)

4.2.6 Asset Hooks

Asset hooks can be used to add a new stylesheet easily or a new JavaScript file in the layout. You can use this feature to create a theme and override all Kanboard default styles.

Example to add a new stylesheet:

```
<?php

namespace Kanboard\Plugin\Css;

use Kanboard\Core\Plugin\Base;

class Plugin extends Base
{
    public function initialize()
    {
```

(continues on next page)

(continued from previous page)

```
        $this->hook->on('template:layout:css', array('template' => 'plugins/Css/skin.
↪css'));
    }
}
```

List of asset Hooks:

- `template:layout:css`
- `template:layout:js`

4.2.7 Reference hooks

Reference hooks are passing a variable by reference.

Example:

```
$this->hook->on('formatter:board:query', function (\PicoDb\Table &$query) {
    $query->eq(TaskModel::TABLE.'color_id', 'red');
});
```

The code above will show only tasks in red on the board.

List of reference hooks:

Hook	Description
`formatter:board:query`	Alter database query before rendering board
`pagination:dashboard:project:query`	Alter database query for projects pagination on the dashboard
`pagination:dashboard:task:query`	Alter database query for tasks pagination on the dashboard
`pagination:dashboard:subtask:query`	Alter database query for subtasks pagination on the dashboard
`model:task:creation:prepare`	Alter form values before to save a task
`model:task:creation:aftersave`	Retrieve Task ID after creating a task
`model:task:modification:prepare`	Alter form values before to edit a task
`model:color:get-list`	Alter default_colors values
`model:subtask:modification:prepare`	Alter form values before to save a subtask
`model:subtask:creation:prepare`	Alter form values before to edit a subtask
`model:subtask:count:query`	Alter database query for subtask count

4.2.8 Template Hooks

Template hooks allow additional content in existing templates.

Example to add new content in the dashboard sidebar:

```
$this->template->hook->attach('template:dashboard:sidebar', 'myplugin:dashboard/
↪sidebar');
```

Example to attach a template with local variables:

```
$this->template->hook->attach('template:dashboard:sidebar', 'myplugin:dashboard/
↪sidebar', [
    'variable' => 'foobar',
]);
```

Example to attach a template with a callable:

```
$this->template->hook->attachCallable('template:dashboard:sidebar',
↪'myplugin:dashboard/sidebar', function($hook_param1, $hook_param2) {
    return ['new_template_variable' => 'foobar']; // Inject a new variable into the␣
↪plugin template
});
```

This call is usually defined in the `initialize()` method. The first argument is the name of the hook and the second argument is the template name.

Template names prefixed with the plugin name and colon indicate the location of the template.

Example with `myplugin:dashboard/sidebar`:

- `myplugin` is the name of your plugin (lowercase)
- `dashboard/sidebar` is the template name
- On the filesystem, the plugin will be located here: `plugins\Myplugin\Template\dashboard\sidebar.php`
- Templates are written in pure PHP (don't forget to escape data)

Template names without prefix are core templates.

List of template hooks:

Hook	Description
`template:analytic:sidebar`	Sidebar on analytic pages
`template:app:filters-helper:before`	Filter helper dropdown (top)
`template:app:filters-helper:after`	Filter helper dropdown (bottom)
`template:auth:login-form:before`	Login page (top)
`template:auth:login-form:after`	Login page (bottom)
`template:board:private:task:before-title`	Task in private board: before title
`template:board:private:task:after-title`	Task in private board: after title
`template:board:public:task:before-title`	Task in public board: before title
`template:board:public:task:after-title`	Task in public board: after title
`template:board:task:footer`	Task in board: footer
`template:board:task:icons`	Task in board: tooltip icon
`template:board:table:column:before-header-row`	Row before board column header
`template:board:table:column:after-header-row`	Row after board column header
`template:board:column:dropdown`	Dropdown menu in board columns
`template:board:column:header`	Board column header
`template:board:tooltip:subtasks:header:before-assignee`	Header of Subtask table on tootip befor
`template:board:tooltip:subtasks:rows`	Column on row of Subtask table on too
`template:config:sidebar`	Sidebar on settings page
`template:config:application`	Application settings form
`template:config:board`	Board settings form
`template:config:email`	Email settings page
`template:config:integrations`	Integration page in global settings
`template:dashboard:show`	Main page of the dashboard
`template:dashboard:page-header:menu`	Dashboard submenu
`template:dashboard:project:after-title`	Dashboard project after title
`template:dashboard:project:before-title`	Dashboard project before title
`template:dashboard:show:after-filter-box`	Dashboard after filter box

Continued on

Table 1 – continued from previous page

Hook	Description
`template:dashboard:show:before-filter-box`	Dashboard before filter box
`template:dashboard:task:footer`	Task in dashboard: footer
`template:dashboard:sidebar`	Dashboard sidebar
`template:export:header`	Export header
`template:header:dropdown`	Page header dropdown menu (user avat
`template:header:creation-dropdown`	Page header dropdown menu (plus icon
`template:layout:head`	Page layout `<head/>` tag
`template:layout:top`	Page layout top header
`template:layout:bottom`	Page layout footer
`template:project:dropdown`	"Actions" menu on left in different proj
`template:project:header:before`	Project filters (before)
`template:project:header:after`	Project filters (after)
`template:project:integrations`	Integration page in projects settings
`template:project:sidebar`	Sidebar in project settings
`template:project-user:sidebar`	Sidebar on project user overview page
`template:project-list:menu:before`	Project list: before menu entries
`template:project-list:menu:after`	Project list: after menu entries
`template:project-overview:before-description`	Project overview: before description
`template:project-overview:images:dropdown`	Project overview: image dropdown
`template:project-permission:after-adduser`	Project permission: after user
`template:project-header:view-switcher`	Project view switcher
`template:project-header:view-switcher-before-project-overview`	Project view switcher before overview
`template:project-header:view-switcher-before-board-view`	Project view switcher before board
`template:project-header:view-switcher-before-task-list`	Project view switcher before list
`template:search:task:footer`	Task in results: footer
`template:task:layout:top`	Task layout top (after page header)
`template:task:details:top`	Task summary top
`template:task:details:bottom`	Task summary bottom
`template:task:details:first-column`	Task summary first column
`template:task:details:second-column`	Task summary second column
`template:task:details:third-column`	Task summary third column
`template:task:details:fourth-column`	Task summary fourth column
`template:task:dropdown`	Task dropdown menu in listing pages
`template:task:sidebar:actions`	Sidebar on task page (section actions)
`template:task:sidebar:information`	Sidebar on task page (section informati
`template:task-file:images:dropdown`	Task image attachment dropdown
`template:task-file:images:before-thumbnail-description`	Task image attachment desciption
`template:task:form:first-column`	1st column in task form
`template:task:form:second-column`	2nd column in task form
`template:task:form:third-column`	3nd column in task form
`template:task:form:bottom-before-buttons`	before form submit/cancel buttons
`template:task:show:top`	Show task page: top
`template:task:show:bottom`	Show task page: bottom
`template:task:show:before-description`	Show task page: before description
`template:task:show:before-tasklinks`	Show task page: before tasklinks
`template:task:show:before-subtasks`	Show task page: before subtasks
`template:task:show:before-external-links`	Show task page: before external links
`template:task:show:before-internal-links`	Show task page: before internal links
`template:task:show:before-timetracking`	Show task page: before timetracking

Continued on

Table 1 – continued from previous page

Hook	Description
`template:task:show:before-attachments`	Show task page: before attachments
`template:task:show:before-comments`	Show task page: before comments
`template:subtask:form:create`	Create Subtask form
`template:subtask:form:edit`	Edit Subtask form
`template:subtask:table:header:before-timetracking`	Subtask table header before Time Track
`template:subtask:table:rows`	Column on row of subtasks table
`template:user:authentication:form`	"Edit authentication" form in user profi
`template:user:create-remote:form`	"Create remote user" form
`template:user:external`	"External authentication" page in user p
`template:user:integrations`	Integration page in user profile
`template:user:sidebar:actions`	Sidebar in user profile (section actions)
`template:user:sidebar:information`	Sidebar in user profile (section informa
`template:user:show:profile:info`	User profile information

4.3 Using Events

Kanboard use internally the Symfony EventDispatcher component to manage internal events.

4.3.1 Event Listening

```
$this->on('app.bootstrap', function($container) {
    // Do something
});
```

- The first argument is the event name (string)
- The second argument is a PHP callable function (closure or class method)

4.3.2 Adding a new event

To add a new event, you have to call the method `register()` of the class `Kanboard\Core\Event\EventManager`:

```
$this->eventManager->register('my.event.name', 'My new event description');
```

These events can be used by other components of Kanboard like automatic actions.

4.4 Metadata

You can attach metadata for each project, task, user or for the whole application. Metadata are custom fields, it's a key/value table.

For example your plugin can store external information for a task or new settings for a project. Basically that allow you to extend the default fields without having to create new tables.

4.4.1 Attach metadata to tasks and remove them

```
// Return a dictionary of metadata (keys/values) for the $task_id
$this->taskMetadataModel->getAll($task_id);

// Get a value only for a task
$this->taskMetadataModel->get($task_id, 'my_plugin_variable', 'default_value');

// Return true if the metadata my_plugin_variable exists
$this->taskMetadataModel->exists($task_id, 'my_plugin_variable');

// Create or update metadata for the task
$this->taskMetadataModel->save($task_id, ['my_plugin_variable' => 'something']);

// Remove a metadata from a project
$this->projectMetadataModel->remove($project_id, my_plugin_variable);
```

4.4.2 Metadata types

- TaskMetadata: `$this->taskMetadataModel`
- ProjectMetadata: `$this->projectMetadataModel`
- UserMetadata: `$this->userMetadataModel`
- Settings/Config: `$this->configModel`

Note: Always prefix the metadata name with your plugin name

4.5 Add Notification Types with Plugins

You can send notifications to almost any system by adding a new type. There are two kinds of notifications: project and user.

- Project: Notifications configured at the project level
- User: Notifications sent individually and configured at the user profile

4.5.1 Register a new notification type

In your plugin registration file call the method `setType()`:

```
$this->userNotificationTypeModel->setType('irc', t('IRC'),
↪'\Kanboard\Plugin\IRC\Notification\IrcHandler');
$this->projectNotificationTypeModel->setType('irc', t('IRC'),
↪'\Kanboard\Plugin\IRC\Notification\IrcHandler');
```

Your handler can be registered for user or project notification. You don't necessarily need to support both.

When your handler is registered, the end-user can choose to receive the new notification type or not.

4.5.2 Notification Handler

Your notification handler must implement the interface `Kanboard\Core\Notification\NotificationInterface`:

```
interface NotificationInterface
{
    /**
     * Send notification to a user
     *
     * @access public
     * @param  array     $user
     * @param  string    $event_name
     * @param  array     $event_data
     */
    public function notifyUser(array $user, $event_name, array $event_data);

    /**
     * Send notification to a project
     *
     * @access public
     * @param  array     $project
     * @param  string    $event_name
     * @param  array     $event_data
     */
    public function notifyProject(array $project, $event_name, array $event_data);
}
```

4.5.3 Example of notification plugins

- Slack
- Hipchat
- Jabber

4.6 Plugin Overrides

4.6.1 Override HTTP Content Security Policy

If you would like to replace the default HTTP Content Security Policy header, you can use the method `setContentSecurityPolicy()`:

```
<?php

namespace Kanboard\Plugin\Csp;

use Kanboard\Core\Plugin\Base;

class Plugin extends Base
{
    public function initialize()
    {
        $this->setContentSecurityPolicy(array('script-src' => 'something'));
    }
}
```

4.6.2 Template Overrides

Any templates defined in the core can be overridden. For example, you can redefine the default layout or change email notifications.

Example of template override:

```
$this->template->setTemplateOverride('header', 'theme:layout/header');
```

The first argument is the original template name and the second argument the template to use as replacement.

You can still use the original template using the "kanboard:" prefix:

```
<?= $this->render('kanboard:header') ?>
```

4.6.3 Formatter Overrides

Here an example to override formatter objects in Kanboard:

```
class MyFormatter extends UserAutoCompleteFormatter
{
    public function format()
    {
        $users = parent::format();

        foreach ($users as &$user) {
            $user['label'] = 'something'; // Do something useful here
        }

        return $users;
    }
}

class Plugin extends Base
{
    public function initialize()
    {
        $this->container['userAutoCompleteFormatter'] = $this->container->
↪factory(function ($c) {
            return new MyFormatter($c);
        });
    }
}
```

4.7 Custom Routes

When URL rewriting is enabled, you can define custom routes from your plugins.

4.7.1 Define new routes

Routes are handled by the class `Kanboard\Core\Http\Route`.

New routes can be added by using the method `addRoute($path, $controller, $action, $plugin)`, here an example:

```
$this->route->addRoute('/my/custom/route', 'myController', 'myAction', 'myplugin');
```

When the end-user go to the URL `/my/custom/route`, the method `Kanboard\Plugin\Myplugin\Controller\MyController::myAction()` will be executed.

The first character of the controller and the plugin name will converted in uppercase with the function `ucfirst()`.

You can also define routes with variables:

```
$this->route->addRoute('/my/route/:my_variable', 'myController', 'myAction', 'myplugin
↪');
```

The colon prefix `:`, define a variable. For example `:my_variable` declare a new variable named `my_variable`.

To fetch the value of the variable you can use the method `getStringParam()` or `getIntegerParam()` from the class `Kanboard\Core\Http\Request`:

If we have the URL `/my/route/foobar`, the value of `my_variable` is `foobar`:

```
$this->request->getStringParam('my_variable'); // Return foobar
```

4.7.2 Generate links based on the routing table

From templates, you have to use the helper `Kanboard\Helper\Url`.

Generate a HTML link

```
<?= $this->url->link('My link', 'mycontroller', 'myaction', array('plugin' =>
↪'myplugin')) ?>
```

Will generate this HTML:

```
<a href="/my/custom/route">My link</a>
```

Generate only the attribute `href`:

```
<?= $this->url->href('My link', 'mycontroller', 'myaction', array('plugin' =>
↪'myplugin')) ?>
```

HTML output:

```
/my/custom/route
```

HTML output when URL rewriting is not enabled:

```
?controller=mycontroller&amp;action=myaction&amp;plugin=myplugin
```

Generate redirect link:

From a controller, if you need to perform a redirection:

```
$this->url->to('mycontroller', 'myaction', array('plugin' => 'myplugin'));
```

Generate:

```
?controller=mycontroller&action=myaction&plugin=myplugin
```

4.8 Registering new helpers

Helper skeleton:

```
<?php

namespace Kanboard\Plugin\MyPlugin\Helper;

use Kanboard\Core\Base;

class MyHelper extends Base
{
    public function doSomething()
    {
        return 'foobar';
    }
}
```

Register your helper class:

```
$this->helper->register('myHelper', '\Kanboard\Plugin\MyPlugin\Helper\MyHelper');
```

Using your helper from a template:

```
<p>
    <?= $this->myHelper->doSomething() ?>
</p>
```

Using your helper from another class:

```
$this->helper->myHelper->doSomething();
```

4.9 Plugin Schema Migrations

Kanboard executes database migrations automatically for you. Migrations must be stored in a folder **Schema** and the filename must be the same as the database driver:

```
Schema
├── Mysql.php
├── Postgres.php
└── Sqlite.php
```

Each file contains all migrations, here an example for Sqlite:

```
<?php

namespace Kanboard\Plugin\Something\Schema;

const VERSION = 1;

function version_1($pdo)
{
    $pdo->exec('CREATE TABLE IF NOT EXISTS something (
        "id" INTEGER PRIMARY KEY,
        "project_id" INTEGER NOT NULL,
        "something" TEXT,
        FOREIGN KEY(project_id) REFERENCES projects(id) ON DELETE CASCADE
    )');
}
```

- The constant `VERSION` is the last version of your schema
- Each function is a migration `version_1()`, `version_2()`, etc.
- A `PDO` instance is passed as first argument
- Everything is executed inside a transaction, if something doesn't work a rollback is performed and the error is displayed to the user

Kanboard will compare the version defined in your schema and the version stored in the database. If the versions are different, Kanboard will execute one by one each migration until to reach the last version.

4.10 Avatar Providers

4.10.1 Registration

```
$this->avatarManager->register(new CustomAvatarProvider());
```

4.10.2 Interface

The provider must implements the interface `Kanboard\Core\User\Avatar\AvatarProviderInterface`:

Method	Description
`render(array $user, $size)`	Render HTML
`isActive(array $user)`	Returns a boolean if the provider is able to render something

The `$user` argument is a dictionary that contains these keys:

```
[
    'id' => 123,
    'username' => 'admin',
    'name' => 'Administrator',
    'email' => 'me@localhost',
]
```

4.11 Mail Transport

By default Kanboard supports 3 standards mail transports:

- Mail (PHP mail function)
- Smtp
- Sendmail command

With the plugin API you can add a driver for any email provider. For example, your plugin can add a mail transport for a provider that uses an HTTP API.

4.11.1 Implementation

Your plugin must implement the interface `Kanboard\Core\Mail\ClientInterface` and extends from `Kanboard\Core\Base`.

The only method you need to implement is `sendEmail()`:

```
interface ClientInterface
{
    /**
     * Send a HTML email
     *
     * @access public
     * @param  string $recipientEmail
     * @param  string $recipientName
     * @param  string $subject
     * @param  string $html
     * @param  string $authorName
     * @param  string $authorEmail
     */
    public function sendEmail($recipientEmail, $recipientName, $subject, $html,
↪$authorName, $authorEmail = '');
}
```

To register your new mail transport, use the method `setTransport($transport, $class)` from the class `Kanboard\Core\Mail\Client`:

```
$this->emailClient->setTransport('myprovider',
↪'\Kanboard\Plugin\MyProvider\MyEmailHandler');
```

The second argument contains the absolute name space of your concrete class.

4.11.2 Examples of mail transport plugins

- Sendgrid
- Mailgun
- Postmark

4.12 Adding Automatic Actions

Adding a new automatic action is pretty simple.

4.12.1 Creating a new action

Your automatic action must inherit of the class `Kanboard\Action\Base`. Several abstract methods must be implemented by yourself:

Method	Description
`getDescription()`	Description visible in the user interface
`getCompatibleEvents()`	Get the list of compatible events
`getActionRequiredParameters()`	Get the required parameter for the action (defined by the user)
`getEventRequiredParameters()`	Get the required parameter for the event
`doAction(array $data)`	Execute the action, must return true on success
`hasRequiredCondition(array $data)`	Check if the event data meet the action condition

Your automatic action is identified in Kanboard by using the absolute class name with the name space included.

4.12.2 Adding new events

The list of application events is available in the class `Kanboard\Core\Event\EventManager::getAll()`. However, if your plugin fires new events, you can register these events like that:

```
$this->actionManager->getAction('\Kanboard\Plugin\MyPlugin\MyActionName')->addEvent(
↪'my.event', 'My event description');
```

You can extend the list of compatible events of existing actions by using the same method.

4.12.3 Registering the action

You have to call the method `register()` from the class `Kanboard\Core\Action\ActionManager`:

```
<?php

namespace Kanboard\Plugin\AutomaticAction;

use Kanboard\Core\Plugin\Base;
use Kanboard\Plugin\AutomaticAction\Action\TaskRename;

class Plugin extends Base
{
    public function initialize()
    {
        $this->actionManager->register(new TaskRename($this->container));
    }
}
```

4.12.4 Example

- Automatic Action example

4.13 Authentication Architecture

Kanboard provides a flexible and pluggable authentication architecture.

By default, user authentication can be done with multiple methods:

- Username and password authentication (Local database and LDAP)
- OAuth2 authentication
- Reverse-Proxy authentication
- Cookie based authentication (Remember Me)

More over, after a successful authentication, a Two-Factor post authentication can be done. Kanboard supports natively the TOTP standard.

4.13.1 Authentication Interfaces

To have a pluggable system, authentication drivers must implement a set of interfaces:

Interface	Role
AuthenticationProviderInterface	Base interface for other authentication interfaces
PreAuthenticationProviderInterface	The user is already authenticated when reaching the application, web servers usually define some environment variables
PasswordAuthenticationProviderInterface	Authentication methods that uses the username and password provided in the login form
OAuthAuthenticationProviderInterface	OAuth2 providers
PostAuthenticationProviderInterface	Two-Factor auhentication drivers, ask for confirmation code
SessionCheckProviderInterface	Providers that are able to check if the user session is valid

Examples of authentication providers:

- The default Database method implements `PasswordAuthenticationProviderInterface` and `SessionCheckProviderInterface`
- The Reverse-Proxy method implements `PreAuthenticationProviderInterface` and `SessionCheckProviderInterface`
- The Google method implements `OAuthAuthenticationProviderInterface`
- The LDAP method implements `PasswordAuthenticationProviderInterface`
- The RememberMe cookie method implements `PreAuthenticationProviderInterface`
- The Two-Factor TOTP method implements `PostAuthenticationProviderInterface`

4.13.2 Authentication Workflow

For each HTTP request:

1. If the user session is already open, execute registered providers that implements `SessionCheckProviderInterface`
2. Execute all providers that implements `PreAuthenticationProviderInterface`
3. If the end-user submit the login form, providers that implements `PasswordAuthenticationProviderInterface` are executed
4. If the end-user wants to use OAuth2, the selected provider will be executed
5. After a successful authentication, the last registered `PostAuthenticationProviderInterface` will be used
6. Synchronize user information if necessary

This workflow is managed by the class `Kanboard\Core\Security\AuthenticationManager`.

Events triggered:

- `AuthenticationManager::EVENT_SUCCESS`: Successful authentication
- `AuthenticationManager::EVENT_FAILURE`: Failed authentication

Each time a failure event occurs, the counter of failed logins is incremented.

The user account can be locked down for the configured period of time and a captcha can be shown to avoid brute force attacks.

4.13.3 User Provider Interface

When the authentication is successful, the `AuthenticationManager` will ask the user information to your driver by calling the method `getUser()`. This method must return an object that implements the interface `Kanboard\Core\User\UserProviderInterface`.

This class abstract the information gathered from another system.

Examples:

- `DatabaseUserProvider` provides information for an internal user
- `LdapUserProvider` for a LDAP user
- `ReverseProxyUserProvider` for a Reverse-Proxy user
- `GoogleUserProvider` represents a Google user

Methods for User Provider Interface:

- `isUserCreationAllowed()`: Return true to allow automatic user creation
- `getExternalIdColumn()`: Get external id column name (google_id, github_id, gitlab_id…)
- `getInternalId()`: Get internal database id
- `getExternalId()`: Get external id (Unique id)
- `getRole()`: Get user role
- `getUsername()`: Get username
- `getName()`: Get user full name
- `getEmail()`: Get user email address
- `getExternalGroupIds()`: Get external group ids, automatically sync group membership if present
- `getExtraAttributes()`: Get extra attributes to set for the user during the local sync

It's not mandatory to return a value for each method.

4.13.4 User Local Synchronization

User information can be automatically synced with the local database.

- If the method `getInternalId()` return a value no synchronization is performed
- The methods `getExternalIdColumn()` and `getExternalId()` must return a value to sync the user
- Properties that returns an empty string won't be synced

4.14 Authentication Providers

New authentication backends can be written with very few lines of code.

4.14.1 Provider Registration

In the method `initialize()` of your plugin, call the method `register()` of the class `AuthenticationManager`:

```
public function initialize()
{
    $this->authenticationManager->register(new ReverseProxyLdapAuth($this->
↪container));
}
```

The object provided to the method `register()` must implement one of the pre-defined authentication interfaces.

Those interfaces are defined in the namepsace `Kanboard\Core\Security`:

- `Kanboard\Core\Security\PreAuthenticationProviderInterface`
- `Kanboard\Core\Security\PostAuthenticationProviderInterface`
- `Kanboard\Core\Security\PasswordAuthenticationProviderInterface`
- `Kanboard\Core\Security\OAuthAuthenticationProviderInterface`

The only requirement is to implement the interfaces, you class can be written the way you want and located anywhere on the disk.

4.14.2 User Provider

When the authentication is successful, your driver must return an object that represents the user. This object must implement the interface `Kanboard\Core\User\UserProviderInterface`.

4.14.3 Example of authentication plugins

- Authentication providers included in Kanboard
- Reverse-Proxy Authentication with LDAP support
- SMS Two-Factor Authentication

4.15 Authorization Architecture

Kanboard supports multiple roles at the application level and at the project level.

4.15.1 Authorization Workflow

For each HTTP request:

1. Authorize or not access to the resource based on the application access list
2. If the resource is for a project (board, task. . .):
   1. Fetch user role for this project
   2. Grant/Denied access based on the project access map

4.15.2 Extending Access Map

The Access List (ACL) is based on the controller class name and the method name. The list of access is handled by the class `Kanboard\Core\Security\AccessMap`.

There are two access map: one for the application and another one for projects.

- Application access map: `$this->applicationAccessMap`
- Project access map: `$this->projectAccessMap`

Examples to define a new policy from your plugin:

```
// All methods of the class MyController:
$this->projectAccessMap->add('MyController', '*', Role::PROJECT_MANAGER);

// Specific methods:
$this->projectAccessMap->add('MyOtherController', array('create', 'save'),
↪Role::PROJECT_MEMBER);
```

Roles are defined in the class `Kanboard\Core\Security\Role`.

The Authorization class (`Kanboard\Core\Security\Authorization`) will check the access for each page.

4.16 Custom Group Providers

Kanboard is able to load groups from an external system. This feature is mainly used for project permissions.

Project managers can allow access to a project for a group. The end-user will use an auto-complete box and search for a group.

Each time a group query is executed, all registered group providers are executed.

4.16.1 Group Provider Workflow

1. The end-user start to type the group name in the auto-complete field
2. The `GroupManager` class will execute the query across all registered group providers
3. Results are merged and returned to the user interface

4. After selecting a group, the information of the group are synced to the local database if necessary

4.16.2 Group Provider Interface

Interface to implement: `Kanboard\Core\Group\GroupProviderInterface`.

Classes that implements this interface abstract the group information, there are only 3 methods:

- `getInternalId()`: Get internal database id, return 0 otherwise
- `getExternalId()`: Get external unique id
- `getName()`: Get group name

Kanboard will use the external id to sync with the local database.

4.16.3 Group Backend Provider Interface

Interface to implement: `Kanboard\Core\Group\GroupBackendProviderInterface`.

This interface requires only one method: `find($input)`. The argument `$input` is the text entered from the user interface.

This method must return a list of `GroupProviderInterface`, this is the result of the search.

4.16.4 Backend Registration from Plugins

In the method `initialize()` of your plugin register your custom backend like that:

```
$groupManager->register(new MyCustomLdapBackendGroupProvider($this->container));
```

4.16.5 Examples

- Group providers included in Kanboard (LDAP and Database)

4.17 External Link Providers

This functionality allows you to link a task to additional items stored on another system.

For example, you can link a task to:

- Traditional web page
- Attachment (PDF documents stored on the web, archive…)
- Any ticketing system (bug tracker, customer support ticket…)

Each item has a type, a URL, a dependency type and a title.

By default, Kanboard includes two kinds of providers:

- Web Link: You copy and paste a link and Kanboard will fetch the page title automatically
- Attachment: Link to anything that is not a web page

4.17.1 Workflow

1. The end-user copy and paste the URL to the form and submit
2. If the link type is "auto", Kanboard will loop through all providers registered until there is a match
3. Then, the link provider returns a object that implements the interface `ExternalLinkInterface`
4. A form is shown to the user with all pre-filled data before to save the link

4.17.2 Interfaces

To implement a new link provider from a plugin, you need to create 2 classes that implement those interfaces:

- `Kanboard\Core\ExternalLink\ExternalLinkProviderInterface`
- `Kanboard\Core\ExternalLink\ExternalLinkInterface`

ExternalLinkProviderInterface

Method	Usage
`getName()`	Get provider name (label)
`getType()`	Get link type (will be saved in the database)
`getDependencies()`	Get a dictionary of supported dependency types by the provider
`setUserTextInput($input)`	Set text entered by the user
`match()`	Return true if the provider can parse correctly the user input
`getLink()`	Get the link found with the properties

ExternalLinkInterface

Method	Usage
`getTitle()`	Get link title
`getUrl()`	Get link URL
`setUrl($url)`	Set link URL

4.17.3 Register a new link provider

In your `Plugin.php`, just call the method `register()` from the object `ExternalLinkManager`:

```
<?php

namespace Kanboard\Plugin\MyExternalLink;

use Kanboard\Core\Plugin\Base;

class Plugin extends Base
{
    public function initialize()
    {
        $this->externalLinkManager->register(new MyLinkProvider());
    }
}
```

4.17.4 Examples

- Kanboard includes the default providers "WebLink" and "Attachment"

4.18 External Task Providers

Kanboard can be used to manage tasks stored in another system. For example, an external system can be a bug tracker or any kind of ticketing software. In this way, you can use Kanboard to manage external tasks in the same way as native tasks.

4.18.1 Workflow

Creation:

1. The end-user select an alternative task provider during the task creation
2. The external task provider expose a form to the user to be able to fetch the external task
3. The external task is retrieved from the other system
4. A customized form is shown to the user

Visualization:

When the task detail page is opened, Kanboard will load asynchronously the remote task. This information might be cached by the plugin to improve the loading time.

Modification:

Optionally, the plugin can offer a custom form to save extra information to the external system.

4.18.2 Interfaces

External task providers must implements at least two interfaces:

- `Kanboard\Core\ExternalTask\ExternalTaskProviderInterface`
- `Kanboard\Core\ExternalTask\ExternalTaskInterface`

ExternalTaskProviderInterface

Method	Usage
`getName()`	Get provider name (label)
`fetch()`	Retrieve task from external system or cache
`save($uri, array $formValues, array &$formErrors)`	Save external task to another system
`getImportFormTemplate()`	Get task import template name
`getCreationFormTemplate()`	Get creation form template
`getModificationFormTemplate()`	Get modification form template
`getViewTemplate()`	Get task view template name
`buildTaskUri(array $formValues)`	Build external task URI based on import form values

ExternalTaskInterface

Method	Usage
`getUri()`	Return Uniform Resource Identifier for the task
`getFormValues()`	Return a dict to populate the task form

4.18.3 Exceptions

The plugin may raise an exception if something goes wrong:

- `Kanboard\Core\ExternalTask\ExternalTaskException`: Generic error related to the external system
- `Kanboard\Core\ExternalTask\AccessForbiddenException`: Access not allowed by the external system
- `Kanboard\Core\ExternalTask\NotFoundException`: External task not found

4.18.4 Provider Registration

```
class Plugin extends Base
{
    public function initialize()
    {
        $this->externalTaskManager->register(new MyExternalTaskProvider());
    }
}
```

4.19 LDAP Library

To facilitate LDAP integration, Kanboard has its own LDAP library. This library can perform common operations.

4.19.1 Client

Class: `Kanboard\Core\Ldap\Client`

To connect to your LDAP server easily, use this method:

```
use Kanboard\Core\Ldap\Client as LdapClient;
use Kanboard\Core\Ldap\ClientException as LdapException;

try {
    $client = LdapClient::connect();

    // Get native LDAP resource
    $resource = $client->getConnection();

    // ...

} catch (LdapException $e) {
```

(continues on next page)

(continued from previous page)

```
    // ...
}
```

4.19.2 LDAP Queries

Classes:

- `Kanboard\Core\Ldap\Query`
- `Kanboard\Core\Ldap\Entries`
- `Kanboard\Core\Ldap\Entry`

Example to query the LDAP directory:

```
$query = new Query($client)
$query->execute('ou=People,dc=kanboard,dc=local', 'uid=my_user', array('cn', 'mail'));

if ($query->hasResult()) {
    $entries = $query->getEntries(); // Return an instance of Entries
}
```

Read one entry:

```
$firstEntry = $query->getEntries()->getFirstEntry();
$email = $firstEntry->getFirstValue('mail');
$name = $firstEntry->getFirstValue('cn', 'Default Name');
```

Read multiple entries:

```
foreach ($query->getEntries()->getAll() as $entry) {
    $emails = $entry->getAll('mail'); // Fetch all emails
    $dn = $entry->getDn(); // Get LDAP DN of this user

    // Check if a value is present for an attribute
    if ($entry->hasValue('mail', 'user2@localhost')) {
        // ...
    }
}
```

4.19.3 User Helper

Class: `Kanboard\Core\Ldap\User`

Fetch a single user in one line:

```
// Return an instance of LdapUserProvider
$user = User::getUser($client, 'my_username');
```

4.19.4 Group Helper

Class: `Kanboard\Core\Ldap\Group`

Fetch groups in one line:

```
// Define LDAP filter
$filter = '(&(objectClass=group)(sAMAccountName=My group*))';

// Return a list of LdapGroupProvider
$groups = Group::getGroups($client, $filter);
```

CHAPTER 5

API Reference

Documentation of API endpoints.

5.1 Introduction

5.1.1 User and application API

There are two types of API access:

Application API

- Access to the API with the user "jsonrpc" and the token available on the settings page
- Access to all procedures
- No permission checked
- There is no user session on the server
- No access to procedures that starts with "My…" (example: "getMe" or "getMyProjects")
- Example of possible clients: tools to migrate/import data, create tasks from another system, etc…

User API

- Access to the API with the user credentials (username and password)
- You can also generate a personal access token instead of your password
- Application role and project permissions are checked for each procedure
- A user session is created on the server
- Example of possible clients: native mobile/desktop application, command line utility, etc…

5.1.2 Security

- Always use HTTPS with a valid certificate (avoid clear text communication)
- If you develop a mobile application, it's your responsability to store securely the user credentials on the device
- After 3 authentication failures on the user API, the end-user have to unlock his account by using the login form

Warning: Since Kanboard v1.2.8, people with two-factor authentication enabled must use API keys.

5.1.3 Protocol

Kanboard uses the protocol Json-RPC to interact with external programs.

JSON-RPC is a remote procedure call protocol encoded in JSON. Almost the same thing as XML-RPC but with the JSON format.

We use the version 2 of the protocol. You must call the API with a `POST` HTTP request.

Kanboard support batch requests, so you can make multiple API calls in a single HTTP request. It's particularly useful for mobile clients with higher network latency.

5.2 API Authentication

Warning: Since Kanboard v1.2.8, people with two-factor authentication enabled must use API keys.

5.2.1 API endpoint

URL: `https://YOUR_SERVER/jsonrpc.php`

5.2.2 Default method (HTTP Basic)

Application credentials

- Username: `jsonrpc`
- Password: API token on the settings page

User credentials

- Username: username
- Password: user password or personal access token

The API use the HTTP Basic Authentication Scheme described in the RFC2617.

5.2.3 Custom HTTP header

You can use an alternative HTTP header for the authentication if your server have a very specific configuration.

- The header name can be anything, for example `X-API-Auth`.
- The header value is the `username:password` encoded in Base64.

Configuration:

1. Define your custom header in your `config.php`: `define('API_AUTHENTICATION_HEADER', 'X-API-Auth');`
2. Encode the credentials in Base64, example with PHP `base64_encode('jsonrpc:19ffd9709d03ce50675c3a43d1c4`
3. Test with curl:

```
curl \
-H 'X-API-Auth:␣
↪anNvbnJwYzoxOWZmZDk3MDlkMDNjZTUwNjc1YzNhNDNkMWM0OWMxYWMyMDdmNGJjNDVmMDZjNWIyNzAxZmJkZjg5Mjk=
↪' \
-d '{"jsonrpc": "2.0", "method": "getAllProjects", "id": 1}' \
http://localhost/kanboard/jsonrpc.php
```

5.2.4 Authentication error

If the credentials are wrong, you will receive a `401 Not Authorized` and the corresponding JSON response.

5.2.5 Authorization error

If the connected user is not allowed to access to the resource, you will receive a `403 Forbidden`.

5.3 API Examples

5.3.1 Example with cURL

With the special user `jsonrpc`:

```
curl \
-u "jsonrpc:19ffd9709d03ce50675c3a43d1c49c1ac207f4bc45f06c5b2701fbdf8929" \
-d '{"jsonrpc": "2.0", "method": "getAllProjects", "id": 1}' \
http://localhost/kanboard/jsonrpc.php
```

With a normal user:

```
curl \
-u "username:secret" \
-d '{"jsonrpc": "2.0", "method": "getMyProjects", "id": 1}' \
http://localhost/kanboard/jsonrpc.php
```

Response from the server:

```
{
    "jsonrpc":"2.0",
    "id":1,
    "result":[
        {
            "id":"1",
            "name":"API test",
            "is_active":"1",
            "token":"6bd0932fe7f4b5e6e4bc3c72800bfdef36a2c5de2f38f756dfb5bd632ebf",
            "last_modified":"1403392631"
        }
    ]
}
```

5.3.2 Example with Python

There is an official Python client for Kanboard:

```
pip install kanboard
```

Here an example to create a project and a task:

```
import kanboard

kb = kanboard.Client('http://localhost/jsonrpc.php', 'jsonrpc', 'your_api_token')

project_id = kb.create_project(name='My project')
kb.add_project_user(project_id=project_id, user_id=123, role='project-manager')

task_id = kb.create_task(project_id=project_id, title='My task title')
```

5.3.3 Example with Ruby

This example can be used with Kanboard configured with Reverse-Proxy authentication and the API configured with a custom authentication header:

```
require 'faraday'

conn = Faraday.new(:url => 'https://kanboard.example.com') do |faraday|
    faraday.response :logger
    faraday.headers['X-API-Auth'] = 'XXX'       # base64_encode('jsonrpc:API_KEY')
    faraday.basic_auth(ENV['user'], ENV['pw']) # user/pass to get through basic auth
    faraday.adapter Faraday.default_adapter    # make requests with Net::HTTP
end

response = conn.post do |req|
    req.url '/jsonrpc.php'
    req.headers['Content-Type'] = 'application/json'
    req.body = '{ "jsonrpc": "2.0", "id": 1, "method": "getAllProjects" }'
end

puts response.body
```

5.3.4 Example with Java

This is a basic example using Spring. For proper usage see this link.

```
import java.io.UnsupportedEncodingException;
import java.util.Base64;

import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.web.client.RestTemplate;

public class ProjectService {

    public void getAllProjects() throws UnsupportedEncodingException {

        RestTemplate restTemplate = new RestTemplate();

        String url = "http://localhost/kanboard/jsonrpc.php";
        String requestJson = "{\"jsonrpc\": \"2.0\", \"method\": \"getAllProjects\", \
↪"id\": 1}";
        String user = "jsonrpc";
        String apiToken =
↪"19ffd9709d03ce50675c3a43d1c49c1ac207f4bc45f06c5b2701fbdf8929";

        // encode api token
        byte[] xApiAuthTokenBytes = String.join(":", user, apiToken).getBytes("utf-8
↪");
        String xApiAuthToken = Base64.getEncoder().encodeToString(xApiAuthTokenBytes);

        // consume request
        HttpHeaders headers = new HttpHeaders();
        headers.add("X-API-Auth", xApiAuthToken);
        headers.setContentType(MediaType.APPLICATION_JSON);
        HttpEntity<String> entity = new HttpEntity<String>(requestJson, headers);
        String answer = restTemplate.postForObject(url, entity, String.class);
        System.out.println(answer);
    }
}
```

5.4 Application API Procedures

5.4.1 getVersion

- Purpose: **Get the application version**
- Parameters: none
- Result: **version** (Example: 1.0.12, master)

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getVersion",
```

(continues on next page)

(continued from previous page)

```
    "id": 1661138292
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1661138292,
    "result": "1.0.13"
}
```

5.4.2 getTimezone

- Purpose: **Get the timezone of the connected user**
- Parameters: none
- Result on success: **Timezone** (Example: UTC, Europe/Paris)
- Result on failure: **Default timezone** (UTC)

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getTimezone",
    "id": 1661138292
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1661138292,
    "result": "Europe\/Paris"
}
```

5.4.3 getDefaultTaskColors

- Purpose: **Get all default task colors**
- Parameters: None
- Result on success: **Color properties**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getDefaultTaskColors",
    "id": 2108929212
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2108929212,
    "result": {
        "yellow": {
            "name": "Yellow",
            "background": "rgb(245, 247, 196)",
            "border": "rgb(223, 227, 45)"
        },
        "blue": {
            "name": "Blue",
            "background": "rgb(219, 235, 255)",
            "border": "rgb(168, 207, 255)"
        },
        "green": {
            "name": "Green",
            "background": "rgb(189, 244, 203)",
            "border": "rgb(74, 227, 113)"
        },
        "purple": {
            "name": "Purple",
            "background": "rgb(223, 176, 255)",
            "border": "rgb(205, 133, 254)"
        },
        "red": {
            "name": "Red",
            "background": "rgb(255, 187, 187)",
            "border": "rgb(255, 151, 151)"
        },
        "orange": {
            "name": "Orange",
            "background": "rgb(255, 215, 179)",
            "border": "rgb(255, 172, 98)"
        },
        "grey": {
            "name": "Grey",
            "background": "rgb(238, 238, 238)",
            "border": "rgb(204, 204, 204)"
        },
        "brown": {
            "name": "Brown",
            "background": "#d7ccc8",
            "border": "#4e342e"
        },
        "deep_orange": {
            "name": "Deep Orange",
            "background": "#ffab91",
            "border": "#e64a19"
        },
        "dark_grey": {
            "name": "Dark Grey",
            "background": "#cfd8dc",
            "border": "#455a64"
        },
        "pink": {
            "name": "Pink",
            "background": "#f48fb1",
```

(continues on next page)

(continued from previous page)

```
            "border": "#d81b60"
        },
        "teal": {
            "name": "Teal",
            "background": "#80cbc4",
            "border": "#00695c"
        },
        "cyan": {
            "name": "Cyan",
            "background": "#b2ebf2",
            "border": "#00bcd4"
        },
        "lime": {
            "name": "Lime",
            "background": "#e6ee9c",
            "border": "#afb42b"
        },
        "light_green": {
            "name": "Light Green",
            "background": "#dcedc8",
            "border": "#689f38"
        },
        "amber": {
            "name": "Amber",
            "background": "#ffe082",
            "border": "#ffa000"
        }
    }
}
```

5.4.4 getDefaultTaskColor

- Purpose: **Get default task color**
- Parameters: None
- Result on success: **color_id**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getDefaultTaskColor",
    "id": 1144775215
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1144775215,
    "result": "yellow"
}
```

5.4.5 getColorList

- Purpose: **Get the list of task colors**
- Parameters: none
- Result on success: **Dictionary of color_id => color_name**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getColorList",
    "id": 1677051386
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1677051386,
    "result": {
        "yellow": "Yellow",
        "blue": "Blue",
        "green": "Green",
        "purple": "Purple",
        "red": "Red",
        "orange": "Orange",
        "grey": "Grey",
        "brown": "Brown",
        "deep_orange": "Deep Orange",
        "dark_grey": "Dark Grey",
        "pink": "Pink",
        "teal": "Teal",
        "cyan": "Cyan",
        "lime": "Lime",
        "light_green": "Light Green",
        "amber": "Amber"
    }
}
```

5.4.6 getApplicationRoles

- Purpose: **Get the application roles**
- Parameters: none
- Result: **Dictionary of role => role_name**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getApplicationRoles",
    "id": 317154243
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 317154243,
    "result": {
        "app-admin": "Administrator",
        "app-manager": "Manager",
        "app-user": "User"
    }
}
```

5.4.7 getProjectRoles

- Purpose: **Get the project roles**
- Parameters: none
- Result: **Dictionary of role => role_name**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectRoles",
    "id": 8981960
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 8981960,
    "result": {
        "project-manager": "Project Manager",
        "project-member": "Project Member",
        "project-viewer": "Project Viewer"
    }
}
```

5.5 User API Specific Procedures

5.5.1 getMe

- Purpose: **Get logged user session**
- Parameters: None
- Result on success: **user session data**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getMe",
```

(continues on next page)

(continued from previous page)

```
    "id": 1718627783
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1718627783,
    "result": {
        "id": 2,
        "username": "user",
        "role": "app-user",
        "is_ldap_user": false,
        "name": "",
        "email": "",
        "google_id": null,
        "github_id": null,
        "notifications_enabled": "0",
        "timezone": null,
        "language": null,
        "disable_login_form": "0",
        "twofactor_activated": false,
        "twofactor_secret": null,
        "token": "",
        "notifications_filter": "4"
    }
}
```

5.5.2 getMyDashboard

- Purpose: **Get the dashboard of the logged user without pagination**
- Parameters: None
- Result on success: **Dashboard information**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getMyDashboard",
    "id": 447898718
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1563664593,
    "result": {
        "projects": [
            {
                "id": "2",
                "name": "my project",
```

(continues on next page)

(continued from previous page)

```
                "is_active": "1",
                "token": "",
                "last_modified": "1438205337",
                "is_public": "0",
                "is_private": "1",
                "default_swimlane": "Default swimlane",
                "show_default_swimlane": "1",
                "description": null,
                "identifier": "",
                "columns": [
                    {
                        "id": "5",
                        "title": "Backlog",
                        "position": "1",
                        "project_id": "2",
                        "task_limit": "0",
                        "description": "",
                        "nb_tasks": 0
                    },
                    {
                        "id": "6",
                        "title": "Ready",
                        "position": "2",
                        "project_id": "2",
                        "task_limit": "0",
                        "description": "",
                        "nb_tasks": 0
                    },
                    {
                        "id": "7",
                        "title": "Work in progress",
                        "position": "3",
                        "project_id": "2",
                        "task_limit": "0",
                        "description": "",
                        "nb_tasks": 0
                    },
                    {
                        "id": "8",
                        "title": "Done",
                        "position": "4",
                        "project_id": "2",
                        "task_limit": "0",
                        "description": "",
                        "nb_tasks": 0
                    }
                ],
                "url": {
                    "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&
↪project_id=2",
                    "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&
↪action=show&project_id=2",
                    "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&
↪project_id=2"
                }
            }
        ],
```

(continues on next page)

(continued from previous page)

```
        "tasks": [
            {
                "id": "1",
                "title": "new title",
                "date_due": "0",
                "date_creation": "1438205336",
                "project_id": "2",
                "color_id": "yellow",
                "time_spent": "0",
                "time_estimated": "0",
                "project_name": "my project",
                "url": "http:\/\/127.0.0.1:8000\/?controller=task&action=show&task_
↪id=1&project_id=2"
            }
        ],
        "subtasks": []
    }
}
```

5.5.3 getMyActivityStream

- Purpose: **Get the last 100 events for the logged user**
- Parameters: None
- Result on success: **List of events**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getMyActivityStream",
    "id": 1132562181
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1132562181,
    "result": [
        {
            "id": "1",
            "date_creation": "1438205054",
            "event_name": "task.create",
            "creator_id": "2",
            "project_id": "2",
            "task_id": "1",
            "author_username": "user",
            "author_name": "",
            "email": "",
            "task": {
                "id": "1",
                "reference": "",
```

(continues on next page)

(continued from previous page)

```
                "title": "my user title",
                "description": "",
                "date_creation": "1438205054",
                "date_completed": null,
                "date_modification": "1438205054",
                "date_due": "0",
                "date_started": null,
                "time_estimated": "0",
                "time_spent": "0",
                "color_id": "yellow",
                "project_id": "2",
                "column_id": "5",
                "owner_id": "0",
                "creator_id": "2",
                "position": "1",
                "is_active": "1",
                "score": "0",
                "category_id": "0",
                "swimlane_id": "0",
                "date_moved": "1438205054",
                "recurrence_status": "0",
                "recurrence_trigger": "0",
                "recurrence_factor": "0",
                "recurrence_timeframe": "0",
                "recurrence_basedate": "0",
                "recurrence_parent": null,
                "recurrence_child": null,
                "category_name": null,
                "swimlane_name": null,
                "project_name": "my project",
                "default_swimlane": "Default swimlane",
                "column_title": "Backlog",
                "assignee_username": null,
                "assignee_name": null,
                "creator_username": "user",
                "creator_name": ""
            },
            "changes": [],
            "author": "user",
            "event_title": "user created the task #1",
            "event_content": "\n<p class=\"activity-title\">\n    user created the␣
↪task <a href=\"\/?controller=task&amp;action=show&amp;task_id=1&amp;project_id=2\"␣
↪class=\"\" title=\"\" >#1<\/a><\/p>\n<p class=\"activity-description\">\n    <em>my␣
↪user title<\/em>\n<\/p>"
        }
    ]
}
```

5.5.4 createMyPrivateProject

- Purpose: **Create a private project for the logged user**
- Parameters:
  - **name** (string, required)
  - **description** (string, optional)

- Result on success: **project_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createMyPrivateProject",
    "id": 1271580569,
    "params": [
        "my project"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1271580569,
    "result": 2
}
```

5.5.5 getMyProjectsList

- Purpose: **Get projects of the connected user**
- Parameters: None
- Result on success: **dictionary of project_id => project_name**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getMyProjectsList",
    "id": 987834805
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 987834805,
    "result": {
        "2": "my project"
    }
}
```

5.5.6 getMyOverdueTasks

- Purpose: **Get my overdue tasks**
- Result on success: **List of tasks**
- Result on failure: **false**

Request example to fetch all tasks on the board:

```
{
    "jsonrpc": "2.0",
    "method": "getMyOverdueTasks",
    "id": 133280317
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": [
        {
            "id": "1",
            "title": "Task #1",
            "date_due": "1409961789",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        },
        {
            "id": "2",
            "title": "Test",
            "date_due": "1409962115",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        }
    ]
}
```

5.5.7 getMyProjects

- Purpose: **Get projects of connected user with full details**
- Parameters:
  - **none**
- Result on success: **List of projects with details**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getmyProjects",
    "id": 2134420212
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2134420212,
    "result": [
        {
            "id": "1",
            "name": "API test",
            "is_active": "1",
            "token": "",
            "last_modified": "1436119570",
            "is_public": "0",
            "is_private": "0",
            "default_swimlane": "Default swimlane",
            "show_default_swimlane": "1",
            "description": null,
            "identifier": "",
            "url": {
                "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&
↪project_id=1",
                "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&
↪action=show&project_id=1",
                "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&
↪project_id=1"
            }
        }
    ]
}
```

5.6 User API Procedures

5.6.1 createUser

- Purpose: **Create a new user**
- Parameters:
  - **username** Must be unique (string, required)
  - **password** Must have at least 6 characters (string, required)
  - **name** (string, optional)
  - **email** (string, optional)
  - **role** (string, optional, example: app-admin, app-manager, app-user)
- Result on success: **user_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createUser",
    "id": 1518863034,
    "params": {
```

(continues on next page)

(continued from previous page)

```
        "username": "biloute",
        "password": "123456"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1518863034,
    "result": 22
}
```

5.6.2 createLdapUser

- Purpose: **Create a new user authentified by LDAP**
- Parameters:
  - **username** (string, required)
- Result on success: **user_id**
- Result on failure: **false**

The user will only be created if he is found on the LDAP server. This method works only with LDAP authentication configured in proxy or anonymous mode.

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createLdapUser",
    "id": 1518863034,
    "params": {
        "username": "my_ldap_user",
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1518863034,
    "result": 22
}
```

5.6.3 getUser

- Purpose: **Get user information**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **user properties**

- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getUser",
    "id": 1769674781,
    "params": {
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1769674781,
    "result": {
        "id": "1",
        "username": "biloute",
        "password": "$2y$10$dRs6pPoBu935RpmsrhmbjevJH5MgZ7Kr9QrnVINwwyZ3.MOwqg.0m",
        "role": "app-user",
        "is_ldap_user": "0",
        "name": "",
        "email": "",
        "google_id": null,
        "github_id": null,
        "notifications_enabled": "0"
    }
}
```

5.6.4 getUserByName

- Purpose: **Get user information**
- Parameters:
  - **username** (string, required)
- Result on success: **user properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getUserByName",
    "id": 1769674782,
    "params": {
        "username": "biloute"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1769674782,
    "result": {
        "id": "1",
        "username": "biloute",
        "password": "$2y$10$dRs6pPoBu935RpmsrhmbjevJH5MgZ7Kr9QrnVINwwyZ3.MOwqg.0m",
        "role": "app-user",
        "is_ldap_user": "0",
        "name": "",
        "email": "",
        "google_id": null,
        "github_id": null,
        "notifications_enabled": "0"
    }
}
```

5.6.5 getAllUsers

- Purpose: **Get all available users**
- Parameters:
  - **none**
- Result on success: **List of users**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllUsers",
    "id": 1438712131
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1438712131,
    "result": [
        {
            "id": "1",
            "username": "biloute",
            "name": "",
            "email": "",
            "role": "app-user",
            "is_ldap_user": "0",
            "notifications_enabled": "0",
            "google_id": null,
            "github_id": null
        }
    ]
}
```

5.6.6 updateUser

- Purpose: **Update a user**
- Parameters:
  - **id** (integer)
  - **username** (string, optional)
  - **name** (string, optional)
  - **email** (string, optional)
  - **role** (string, optional, example: app-admin, app-manager, app-user)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateUser",
    "id": 322123657,
    "params": {
        "id": 1,
        "role": "app-manager"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 322123657,
    "result": true
}
```

5.6.7 removeUser

- Purpose: **Remove a user**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeUser",
    "id": 2094191872,
    "params": {
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2094191872,
    "result": true
}
```

5.6.8 disableUser

- Purpose: **Disable a user**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "disableUser",
    "id": 2094191872,
    "params": {
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2094191872,
    "result": true
}
```

5.6.9 enableUser

- Purpose: **Enable a user**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "enableUser",
    "id": 2094191872,
    "params": {
```

(continues on next page)

(continued from previous page)

```
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2094191872,
    "result": true
}
```

5.6.10 isActiveUser

- Purpose: **Check if a user is active**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "isActiveUser",
    "id": 2094191872,
    "params": {
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2094191872,
    "result": true
}
```

5.7 Group API Procedures

5.7.1 createGroup

- Purpose: **Create a new group**
- Parameters:
  - **name** (string, required)
  - **external_id** (string, optional)
- Result on success: **link_id**

- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createGroup",
    "id": 1416806551,
    "params": [
        "My Group B",
        "1234"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1416806551,
    "result": 2
}
```

5.7.2 updateGroup

- Purpose: **Update a group**
- Parameters:
  - **group_id** (integer, required)
  - **name** (string, optional)
  - **external_id** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateGroup",
    "id": 866078030,
    "params": {
        "group_id": "1",
        "name": "ABC",
        "external_id": "something"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 866078030,
    "result": true
}
```

5.7.3 removeGroup

- Purpose: **Remove a group**
- Parameters:
  - **group_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeGroup",
    "id": 566000661,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 566000661,
    "result": true
}
```

5.7.4 getGroup

- Purpose: **Get one group**
- Parameters:
  - **group_id** (integer, required)
- Result on success: **Group dictionary**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getGroup",
    "id": 1968647622,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1968647622,
    "result": {
```

(continues on next page)

(continued from previous page)

```
        "id": "1",
        "external_id": "",
        "name": "My Group A"
    }
}
```

5.7.5 getAllGroups

- Purpose: **Get all groups**
- Parameters: none
- Result on success: **list of groups**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllGroups",
    "id": 546070742
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 546070742,
    "result": [
        {
            "id": "1",
            "external_id": "",
            "name": "My Group A"
        },
        {
            "id": "2",
            "external_id": "1234",
            "name": "My Group B"
        }
    ]
}
```

5.8 Group Member API Procedures

5.8.1 getMemberGroups

- Purpose: **Get all groups for a given user**
- Parameters:
  - **user_id** (integer, required)
- Result on success: **List of groups**

- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getMemberGroups",
    "id": 1987176726,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1987176726,
    "result": [
        {
            "id": "1",
            "name": "My Group A"
        }
    ]
}
```

5.8.2 getGroupMembers

- Purpose: **Get all members of a group**
- Parameters:
    - **group_id** (integer, required)
- Result on success: **List of users**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getGroupMembers",
    "id": 1987176726,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1987176726,
    "result": [
        {
            "group_id": "1",
            "user_id": "1",
```

(continues on next page)

(continued from previous page)

```
                "id": "1",
                "username": "admin",
                "is_ldap_user": "0",
                "name": null,
                "email": null,
                "notifications_enabled": "0",
                "timezone": null,
                "language": null,
                "disable_login_form": "0",
                "notifications_filter": "4",
                "nb_failed_login": "0",
                "lock_expiration_date": "0",
                "is_project_admin": "0",
                "gitlab_id": null,
                "role": "app-admin"
            }
    ]
}
```

5.8.3 addGroupMember

- Purpose: **Add a user to a group**
- Parameters:
  - **group_id** (integer, required)
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "addGroupMember",
    "id": 1589058273,
    "params": [
        1,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1589058273,
    "result": true
}
```

5.8.4 removeGroupMember

- Purpose: **Remove a user from a group**

- Parameters:
  - **group_id** (integer, required)
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeGroupMember",
    "id": 1730416406,
    "params": [
        1,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1730416406,
    "result": true
}
```

5.8.5 isGroupMember

- Purpose: **Check if a user is member of a group**
- Parameters:
  - **group_id** (integer, required)
  - **user_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "isGroupMember",
    "id": 1052800865,
    "params": [
        1,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1052800865,
    "result": false
}
```

5.9 Project API Procedures

5.9.1 createProject

- Purpose: **Create a new project**
- Parameters:
  - **name** (string, required)
  - **description** (string, optional)
  - **owner_id** (integer, optional)
  - **identifier** (alphanumeric string, optional)
  - **start_date** ISO8601 format (string, optional)
  - **end_date** ISO8601 format (string, optional)
- Result on success: **project_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createProject",
    "id": 1797076613,
    "params": {
        "name": "PHP client"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1797076613,
    "result": 2
}
```

5.9.2 getProjectById

- Purpose: **Get project information**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **project properties**

- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectById",
    "id": 226760253,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 226760253,
    "result": {
        "id": "1",
        "name": "API test",
        "is_active": "1",
        "token": "",
        "last_modified": "1436119135",
        "is_public": "0",
        "is_private": "0",
        "default_swimlane": "Default swimlane",
        "show_default_swimlane": "1",
        "description": "test",
        "identifier": "",
        "url": {
            "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&project_
↪id=1",
            "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&action=show&
↪project_id=1",
            "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&project_
↪id=1"
        }
    }
}
```

5.9.3 getProjectByName

- Purpose: **Get project information**
- Parameters:
  - **name** (string, required)
- Result on success: **project properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectByName",
```

(continues on next page)

(continued from previous page)

```
    "id": 1620253806,
    "params": {
        "name": "Test"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1620253806,
    "result": {
        "id": "1",
        "name": "Test",
        "is_active": "1",
        "token": "",
        "last_modified": "1436119135",
        "is_public": "0",
        "is_private": "0",
        "default_swimlane": "Default swimlane",
        "show_default_swimlane": "1",
        "description": "test",
        "identifier": "",
        "url": {
            "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&project_
↪id=1",
            "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&action=show&
↪project_id=1",
            "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&project_
↪id=1"
        }
    }
}
```

5.9.4 getProjectByIdentifier

- Purpose: **Get project information**
- Parameters:
  - **identifier** (alphanumeric string, required)
- Result on success: **project properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectByIdentifier",
    "id": 1620253806,
    "params": {
        "identifier": "TEST"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1620253806,
    "result": {
        "id": "1",
        "name": "Test",
        "is_active": "1",
        "token": "",
        "last_modified": "1436119135",
        "is_public": "0",
        "is_private": "0",
        "default_swimlane": "Default swimlane",
        "show_default_swimlane": "1",
        "description": "test",
        "identifier": "TEST",
        "url": {
            "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&project_
↪id=1",
            "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&action=show&
↪project_id=1",
            "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&project_
↪id=1"
        }
    }
}
```

5.9.5 getProjectByEmail

- Purpose: **Get project information**
- Parameters:
  - **email** (string, required)
- Result on success: **project properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectByEmail",
    "id": 1620253806,
    "params": {
        "email": "my_project@my_domain.tld"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1620253806,
    "result": {
        "id": "1",
```

(continues on next page)

(continued from previous page)

```
        "name": "Test",
        "is_active": "1",
        "token": "",
        "last_modified": "1436119135",
        "is_public": "0",
        "is_private": "0",
        "default_swimlane": "Default swimlane",
        "show_default_swimlane": "1",
        "description": "test",
        "identifier": "",
        "email": "my_project@my_domain.tld",
        "url": {
            "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&project_
↪id=1",
            "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&action=show&
↪project_id=1",
            "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&project_
↪id=1"
        }
    }
}
```

5.9.6 getAllProjects

- Purpose: **Get all available projects**
- Parameters:
  - **none**
- Result on success: **List of projects**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllProjects",
    "id": 2134420212
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2134420212,
    "result": [
        {
            "id": "1",
            "name": "API test",
            "is_active": "1",
            "token": "",
            "last_modified": "1436119570",
            "is_public": "0",
            "is_private": "0",
            "default_swimlane": "Default swimlane",
```

(continues on next page)

(continued from previous page)

```
            "show_default_swimlane": "1",
            "description": null,
            "identifier": "",
            "url": {
                "board": "http:\/\/127.0.0.1:8000\/?controller=board&action=show&
↪project_id=1",
                "calendar": "http:\/\/127.0.0.1:8000\/?controller=calendar&
↪action=show&project_id=1",
                "list": "http:\/\/127.0.0.1:8000\/?controller=listing&action=show&
↪project_id=1"
            }
        }
    ]
}
```

5.9.7 updateProject

- Purpose: **Update a project**
- Parameters:
  - **project_id** (integer, required)
  - **name** (string, optional)
  - **description** (string, optional)
  - **owner_id** (integer, optional)
  - **identifier** (string, optional)
  - **start_date** ISO8601 format (string, optional)
  - **end_date** ISO8601 format (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateProject",
    "id": 1853996288,
    "params": {
        "project_id": 1,
        "name": "PHP client update"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1853996288,
    "result": true
}
```

5.9.8 removeProject

- Purpose: **Remove a project**
- Parameters: **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeProject",
    "id": 46285125,
    "params": {
        "project_id": "2"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 46285125,
    "result": true
}
```

5.9.9 enableProject

- Purpose: **Enable a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "enableProject",
    "id": 1775494839,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1775494839,
    "result": true
}
```

5.9.10 disableProject

- Purpose: **Disable a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "disableProject",
    "id": 1734202312,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1734202312,
    "result": true
}
```

5.9.11 enableProjectPublicAccess

- Purpose: **Enable public access for a given project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "enableProjectPublicAccess",
    "id": 103792571,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 103792571,
```

(continues on next page)

(continued from previous page)

```
    "result": true
}
```

5.9.12 disableProjectPublicAccess

- Purpose: **Disable public access for a given project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "disableProjectPublicAccess",
    "id": 942472945,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 942472945,
    "result": true
}
```

5.9.13 getProjectActivity

- Purpose: **Get activity stream for a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **List of events**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectActivity",
    "id": 942472945,
    "params": {
        "project_id": 1
    }
}
```

5.9.14 getProjectActivities

- Purpose: **Get Activityfeed for Project(s)**
- Parameters:
  - **project_ids** (integer array, required)
- Result on success: **List of events**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectActivities",
    "id": 942472945,
    "params": {
        "project_ids": [1,2]
    }
}
```

5.10 Project Permission API Procedures

5.10.1 getProjectUsers

- Purpose: **Get all members of a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **Dictionary of user_id => user name**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectUsers",
    "id": 1601016721,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1601016721,
    "result": {
        "1": "admin"
    }
}
```

5.10.2 getAssignableUsers

- Purpose: **Get users that can be assigned to a task for a project** (all members except viewers)
- Parameters:
  - **project_id** (integer, required)
  - **prepend_unassigned** (boolean, optional, default is false)
- Result on success: **Dictionary of user_id => user name**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAssignableUsers",
    "id": 658294870,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 658294870,
    "result": {
        "1": "admin"
    }
}
```

5.10.3 addProjectUser

- Purpose: **Grant access to a project for a user**
- Parameters:
  - **project_id** (integer, required)
  - **user_id** (integer, required)
  - **role** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "addProjectUser",
    "id": 1294688355,
    "params": [
        "1",
        "1",
        "project-viewer"
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1294688355,
    "result": true
}
```

5.10.4 addProjectGroup

- Purpose: **Grant access to a project for a group**
- Parameters:
  - **project_id** (integer, required)
  - **group_id** (integer, required)
  - **role** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "addProjectGroup",
    "id": 1694959089,
    "params": [
        "1",
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1694959089,
    "result": true
}
```

5.10.5 removeProjectUser

- Purpose: **Revoke user access to a project**
- Parameters:
  - **project_id** (integer, required)
  - **user_id** (integer, required)
- Result on success: **true**

- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeProjectUser",
    "id": 645233805,
    "params": [
        1,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 645233805,
    "result": true
}
```

5.10.6 removeProjectGroup

- Purpose: **Revoke group access to a project**
- Parameters:
  - **project_id** (integer, required)
  - **group_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeProjectGroup",
    "id": 557146966,
    "params": [
        1,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 557146966,
    "result": true
}
```

5.10.7 changeProjectUserRole

- Purpose: **Change role of a user for a project**
- Parameters:
  - **project_id** (integer, required)
  - **user_id** (integer, required)
  - **role** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "changeProjectUserRole",
    "id": 193473170,
    "params": [
        "1",
        "1",
        "project-viewer"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 193473170,
    "result": true
}
```

5.10.8 changeProjectGroupRole

- Purpose: **Change role of a group for a project**
- Parameters:
  - **project_id** (integer, required)
  - **group_id** (integer, required)
  - **role** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "changeProjectGroupRole",
    "id": 2114673298,
    "params": [
        "1",
```

(continues on next page)

(continued from previous page)

```
        "1",
        "project-viewer"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2114673298,
    "result": true
}
```

5.10.9 getProjectUserRole

- Purpose: **Get the role of a user for a given project**
- Parameters:
  - **project_id** (integer, required)
  - **user_id** (integer, required)
- Result on success: **role name**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectUserRole",
    "id": 2114673298,
    "params": [
        "2",
        "3"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2114673298,
    "result": "project-viewer"
}
```

5.11 Project Metadata API Procedures

5.11.1 getProjectMetadata

- Purpose: **Get Project metadata**
- Parameters:

  - **project_id** (integer, required)

- Result on success: **dictionary of metadata**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectMetadata",
    "id": 1797076613,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1797076613,
    "result": {
        "key1": "value1"
    }
}
```

5.11.2 getProjectMetadataByName

- Purpose: **Fetch single metadata value**
- Parameters:
  - **project_id** (integer, required)
  - **name** (string, required)
- Result on success: **mixed**
- Result on failure: **empty string**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectMetadataByName",
    "id": 1797076613,
    "params": {
        "project_id": 1,
        "name": "key1"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1797076613,
    "result": "value1"
}
```

5.11.3 saveProjectMetadata

- Purpose: **Add or update metadata**
- Parameters:
  - **project_id** (integer, required)
  - **values** (dict, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "saveProjectMetadata",
    "id": 1797076613,
    "params": {
        "project_id": 1,
        "values": {
            "key1": "value1"
        }
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1797076613,
    "result": true
}
```

5.11.4 removeProjectMetadata

- Purpose: **Remove a project metadata**
- Parameters:
  - **project_id** (integer, required)
  - **name** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeProjectMetadata",
    "id": 1797076613,
    "params": {
        "project_id": 1,
        "name": "my key"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1797076613,
    "result": true
}
```

5.12 Board API Procedures

5.12.1 getBoard

- Purpose: **Get all necessary information to display a board**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **board properties**
- Result on failure: **empty list**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getBoard",
    "id": 827046470,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 827046470,
    "result": [
        {
            "id": 0,
            "name": "Default swimlane",
            "columns": [
                {
                    "id": "1",
                    "title": "Backlog",
                    "position": "1",
                    "project_id": "1",
                    "task_limit": "0",
                    "description": "",
                    "tasks": [],
                    "nb_tasks": 0,
                    "score": 0
                },
                {
                    "id": "2",
                    "title": "Ready",
```

(continues on next page)

(continued from previous page)

```
                    "position": "2",
                    "project_id": "1",
                    "task_limit": "0",
                    "description": "",
                    "tasks": [
                        {
                            "nb_comments":"0",
                            "nb_files":"0",
                            "nb_subtasks":"0",
                            "nb_completed_subtasks":"0",
                            "nb_links":"0",
                            "id":"2",
                            "reference":"",
                            "title":"Test",
                            "description":"",
                            "date_creation":"1430870507",
                            "date_modification":"1430870507",
                            "date_completed":null,
                            "date_due":"0",
                            "color_id":"yellow",
                            "project_id":"1",
                            "column_id":"2",
                            "swimlane_id":"0",
                            "owner_id":"0",
                            "creator_id":"1",
                            "position":"1",
                            "is_active":"1",
                            "score":"0",
                            "category_id":"0",
                            "date_moved":"1430870507",
                            "recurrence_status":"0",
                            "recurrence_trigger":"0",
                            "recurrence_factor":"0",
                            "recurrence_timeframe":"0",
                            "recurrence_basedate":"0",
                            "recurrence_parent":null,
                            "recurrence_child":null,
                            "assignee_username":null,
                            "assignee_name":null
                        }
                    ],
                    "nb_tasks": 1,
                    "score": 0
                },
                {
                    "id": "3",
                    "title": "Work in progress",
                    "position": "3",
                    "project_id": "1",
                    "task_limit": "0",
                    "description": "",
                    "tasks": [
                        {
                            "nb_comments":"0",
                            "nb_files":"0",
                            "nb_subtasks":"1",
                            "nb_completed_subtasks":"0",
```

(continues on next page)

(continued from previous page)

```
                            "nb_links":"0",
                            "id":"1",
                            "reference":"",
                            "title":"Task with comment",
                            "description":"",
                            "date_creation":"1430783188",
                            "date_modification":"1430783188",
                            "date_completed":null,
                            "date_due":"0",
                            "color_id":"red",
                            "project_id":"1",
                            "column_id":"3",
                            "swimlane_id":"0",
                            "owner_id":"1",
                            "creator_id":"0",
                            "position":"1",
                            "is_active":"1",
                            "score":"0",
                            "category_id":"0",
                            "date_moved":"1430783191",
                            "recurrence_status":"0",
                            "recurrence_trigger":"0",
                            "recurrence_factor":"0",
                            "recurrence_timeframe":"0",
                            "recurrence_basedate":"0",
                            "recurrence_parent":null,
                            "recurrence_child":null,
                            "assignee_username":"admin",
                            "assignee_name":null
                        }
                    ],
                    "nb_tasks": 1,
                    "score": 0
                },
                {
                    "id": "4",
                    "title": "Done",
                    "position": "4",
                    "project_id": "1",
                    "task_limit": "0",
                    "description": "",
                    "tasks": [],
                    "nb_tasks": 0,
                    "score": 0
                }
            ],
            "nb_columns": 4,
            "nb_tasks": 2
        }
    ]
}
```

5.13 Column API Procedures

5.13.1 getColumns

- Purpose: **Get all columns information for a given project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **columns properties**
- Result on failure: **empty list**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getColumns",
    "id": 887036325,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 887036325,
    "result": [
        {
            "id": "1",
            "title": "Backlog",
            "position": "1",
            "project_id": "1",
            "task_limit": "0"
        },
        {
            "id": "2",
            "title": "Ready",
            "position": "2",
            "project_id": "1",
            "task_limit": "0"
        },
        {
            "id": "3",
            "title": "Work in progress",
            "position": "3",
            "project_id": "1",
            "task_limit": "0"
        }
    ]
}
```

5.13.2 getColumn

- Purpose: **Get a single column**

- Parameters:
  - **column_id** (integer, required)
- Result on success: **column properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getColumn",
    "id": 1242049935,
    "params": [
        2
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1242049935,
    "result": {
        "id": "2",
        "title": "Youpi",
        "position": "2",
        "project_id": "1",
        "task_limit": "5"
    }
}
```

5.13.3 changeColumnPosition

- Purpose: **Change the column position**
- Parameters:
  - **project_id** (integer, required)
  - **column_id** (integer, required)
  - **position** (integer, required, must be >= 1)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "changeColumnPosition",
    "id": 99275573,
    "params": [
        1,
        2,
        3
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 99275573,
    "result": true
}
```

5.13.4 updateColumn

- Purpose: **Update column properties**
- Parameters:
  - **column_id** (integer, required)
  - **title** (string, required)
  - **task_limit** (integer, optional)
  - **description** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateColumn",
    "id": 480740641,
    "params": [
        2,
        "Boo",
        5
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 480740641,
    "result": true
}
```

5.13.5 addColumn

- Purpose: **Add a new column**
- Parameters:
  - **project_id** (integer, required)
  - **title** (string, required)
  - **task_limit** (integer, optional)
  - **description** (string, optional)

- Result on success: **column_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "addColumn",
    "id": 638544704,
    "params": [
        1,
        "Boo"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 638544704,
    "result": 5
}
```

5.13.6 removeColumn

- Purpose: **Remove a column**
- Parameters:
  - **column_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeColumn",
    "id": 1433237746,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": true
}
```

5.14 Swimlane API Procedures

5.14.1 getActiveSwimlanes

- Purpose: **Get the list of enabled swimlanes of a project (include default swimlane if enabled)**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **List of swimlanes**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getActiveSwimlanes",
    "id": 934789422,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 934789422,
    "result": [
        {
            "id": 0,
            "name": "Default swimlane"
        },
        {
            "id": "2",
            "name": "Swimlane A"
        }
    ]
}
```

5.14.2 getAllSwimlanes

- Purpose: **Get the list of all swimlanes of a project (enabled or disabled) and sorted by position**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **List of swimlanes**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
```

(continues on next page)

(continued from previous page)

```
    "method": "getAllSwimlanes",
    "id": 509791576,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 509791576,
    "result": [
        {
            "id": "1",
            "name": "Another swimlane",
            "position": "1",
            "is_active": "1",
            "project_id": "1"
        },
        {
            "id": "2",
            "name": "Swimlane A",
            "position": "2",
            "is_active": "1",
            "project_id": "1"
        }
    ]
}
```

5.14.3 getSwimlane

- Purpose: **Get the a swimlane by id**
- Parameters:
  - **swimlane_id** (integer, required)
- Result on success: **swimlane properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getSwimlane",
    "id": 131071870,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
```

(continues on next page)

(continued from previous page)

```
    "id": 131071870,
    "result": {
        "id": "1",
        "name": "Swimlane 1",
        "position": "1",
        "is_active": "1",
        "project_id": "1"
    }
}
```

5.14.4 getSwimlaneById

- Purpose: **Get the a swimlane by id**
- Parameters:
  - **swimlane_id** (integer, required)
- Result on success: **swimlane properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getSwimlaneById",
    "id": 131071870,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 131071870,
    "result": {
        "id": "1",
        "name": "Swimlane 1",
        "position": "1",
        "is_active": "1",
        "project_id": "1"
    }
}
```

5.14.5 getSwimlaneByName

- Purpose: **Get the a swimlane by name**
- Parameters:
  - **project_id** (integer, required)
  - **name** (string, required)

- Result on success: **swimlane properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getSwimlaneByName",
    "id": 824623567,
    "params": [
        1,
        "Swimlane 1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 824623567,
    "result": {
        "id": "1",
        "name": "Swimlane 1",
        "position": "1",
        "is_active": "1",
        "project_id": "1"
    }
}
```

5.14.6 changeSwimlanePosition

- Purpose: **Move up the swimlane position** (only for active swimlanes)
- Parameters:
  - **project_id** (integer, required)
  - **swimlane_id** (integer, required)
  - **position** (integer, required, must be >= 1)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "changeSwimlanePosition",
    "id": 99275573,
    "params": [
        1,
        2,
        3
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 99275573,
    "result": true
}
```

5.14.7 updateSwimlane

- Purpose: **Update swimlane properties**
- Parameters:
  - **project_id** (integer, required)
  - **swimlane_id** (integer, required)
  - **name** (string, required)
  - **description** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateSwimlane",
    "id": 87102426,
    "params": [
        "1",
        "1",
        "Another swimlane"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 87102426,
    "result": true
}
```

5.14.8 addSwimlane

- Purpose: **Add a new swimlane**
- Parameters:
  - **project_id** (integer, required)
  - **name** (string, required)
  - **description** (string, optional)
- Result on success: **swimlane_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "addSwimlane",
    "id": 849940086,
    "params": [
        1,
        "Swimlane 1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 849940086,
    "result": 1
}
```

5.14.9 removeSwimlane

- Purpose: **Remove a swimlane**
- Parameters:
  - **project_id** (integer, required)
  - **swimlane_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeSwimlane",
    "id": 1433237746,
    "params": [
        2,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": true
}
```

5.14.10 disableSwimlane

- Purpose: **Disable a swimlane**

- Parameters:
  - **project_id** (integer, required)
  - **swimlane_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "disableSwimlane",
    "id": 1433237746,
    "params": [
        2,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": true
}
```

5.14.11 enableSwimlane

- Purpose: **Enable a swimlane**
- Parameters:
  - **project_id** (integer, required)
  - **swimlane_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "enableSwimlane",
    "id": 1433237746,
    "params": [
        2,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": true
}
```

5.15 Task API Procedures

5.15.1 createTask

- Purpose: **Create a new task**
- Parameters:
  - **title** (string, required)
  - **project_id** (integer, required)
  - **color_id** (string, optional)
  - **column_id** (integer, optional)
  - **owner_id** (integer, optional)
  - **creator_id** (integer, optional)
  - **date_due**: ISO8601 format (string, optional)
  - **description** Markdown content (string, optional)
  - **category_id** (integer, optional)
  - **score** (integer, optional)
  - **swimlane_id** (integer, optional)
  - **priority** (integer, optional)
  - **recurrence_status** (integer, optional)
  - **recurrence_trigger** (integer, optional)
  - **recurrence_factor** (integer, optional)
  - **recurrence_timeframe** (integer, optional)
  - **recurrence_basedate** (integer, optional)
  - **reference** (string, optional)
  - **tags** ([]string, optional)
  - **date_started**: ISO8601 format (string, optional)
- Result on success: **task_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createTask",
    "id": 1176509098,
    "params": {
        "owner_id": 1,
        "creator_id": 0,
        "date_due": "",
        "description": "",
        "category_id": 0,
        "score": 0,
        "title": "Test",
        "project_id": 1,
        "color_id": "green",
        "column_id": 2,
        "recurrence_status": 0,
        "recurrence_trigger": 0,
        "recurrence_factor": 0,
        "recurrence_timeframe": 0,
        "recurrence_basedate": 0
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1176509098,
    "result": 3
}
```

5.15.2 getTask

- Purpose: **Get task by the unique id**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **task properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getTask",
    "id": 700738119,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 700738119,
    "result": {
        "id": "1",
        "title": "Task #1",
        "description": "",
        "date_creation": "1409963206",
        "color_id": "blue",
        "project_id": "1",
        "column_id": "2",
        "owner_id": "1",
        "position": "1",
        "is_active": "1",
        "date_completed": null,
        "score": "0",
        "date_due": "0",
        "category_id": "0",
        "creator_id": "0",
        "date_modification": "1409963206",
        "reference": "",
        "date_started": null,
        "time_spent": "0",
        "time_estimated": "0",
        "swimlane_id": "0",
        "date_moved": "1430875287",
        "recurrence_status": "0",
        "recurrence_trigger": "0",
        "recurrence_factor": "0",
        "recurrence_timeframe": "0",
        "recurrence_basedate": "0",
        "recurrence_parent": null,
        "recurrence_child": null,
        "url": "http:\/\/127.0.0.1:8000\/?controller=task&action=show&task_id=1&
↪project_id=1",
        "color": {
            "name": "Yellow",
            "background": "rgb(245, 247, 196)",
            "border": "rgb(223, 227, 45)"
        }
    }
}
```

5.15.3 getTaskByReference

- Purpose: **Get task by the external reference**
- Parameters:
  - **project_id** (integer, required)
  - **reference** (string, required)
- Result on success: **task properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getTaskByReference",
    "id": 1992081213,
    "params": {
        "project_id": 1,
        "reference": "TICKET-1234"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1992081213,
    "result": {
        "id": "5",
        "title": "Task with external ticket number",
        "description": "[Link to my ticket](http:\/\/my-ticketing-system\/1234)",
        "date_creation": "1434227446",
        "color_id": "yellow",
        "project_id": "1",
        "column_id": "1",
        "owner_id": "0",
        "position": "4",
        "is_active": "1",
        "date_completed": null,
        "score": "0",
        "date_due": "0",
        "category_id": "0",
        "creator_id": "0",
        "date_modification": "1434227446",
        "reference": "TICKET-1234",
        "date_started": null,
        "time_spent": "0",
        "time_estimated": "0",
        "swimlane_id": "0",
        "date_moved": "1434227446",
        "recurrence_status": "0",
        "recurrence_trigger": "0",
        "recurrence_factor": "0",
        "recurrence_timeframe": "0",
        "recurrence_basedate": "0",
        "recurrence_parent": null,
        "recurrence_child": null,
        "url": "http:\/\/127.0.0.1:8000\/?controller=task&action=show&task_id=5&
↪project_id=1"
    }
}
```

5.15.4 getAllTasks

- Purpose: **Get all available tasks**
- Parameters:
  - **project_id** (integer, required)

  - **status_id**: The value 1 for active tasks and 0 for inactive (integer, required)

- Result on success: **List of tasks**
- Result on failure: **false**

Request example to fetch all tasks on the board:

```
{
    "jsonrpc": "2.0",
    "method": "getAllTasks",
    "id": 133280317,
    "params": {
        "project_id": 1,
        "status_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": [
        {
            "id": "1",
            "title": "Task #1",
            "description": "",
            "date_creation": "1409961789",
            "color_id": "blue",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "position": "1",
            "is_active": "1",
            "date_completed": null,
            "score": "0",
            "date_due": "0",
            "category_id": "0",
            "creator_id": "0",
            "date_modification": "1409961789",
            "reference": "",
            "date_started": null,
            "time_spent": "0",
            "time_estimated": "0",
            "swimlane_id": "0",
            "date_moved": "1430783191",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "priority": "0",
            "external_provider": null,
            "external_uri": null,
            "url": "http:\/\/127.0.0.1:8000\/?controller=task&action=show&task_id=1&
↪project_id=1",
```

(continues on next page)

(continued from previous page)

```
            "color": {
                "name": "Blue",
                "background": "rgb(219, 235, 255)",
                "border": "rgb(168, 207, 255)"
            }
        },
        {
            "id": "2",
            "title": "Test",
            "description": "",
            "date_creation": "1409962115",
            "color_id": "green",
            "project_id": "1",
            "column_id": "2",
            "owner_id": "1",
            "position": "2",
            "is_active": "1",
            "date_completed": null,
            "score": "0",
            "date_due": "0",
            "category_id": "0",
            "creator_id": "0",
            "date_modification": "1409962115",
            "reference": "",
            "date_started": null,
            "time_spent": "0",
            "time_estimated": "0",
            "swimlane_id": "0",
            "date_moved": "1430783191",
            "recurrence_status": "0",
            "recurrence_trigger": "0",
            "recurrence_factor": "0",
            "recurrence_timeframe": "0",
            "recurrence_basedate": "0",
            "recurrence_parent": null,
            "recurrence_child": null,
            "priority": "0",
            "external_provider": null,
            "external_uri": null,
            "url": "http:\/\/127.0.0.1:8000\/?controller=task&action=show&task_id=2&
↪project_id=1",
            "color": {
                "name": "Green",
                "background": "rgb(189, 244, 203)",
                "border": "rgb(74, 227, 113)"
            }
        }
    ]
}
```

5.15.5 getOverdueTasks

- Purpose: **Get all overdue tasks**
- Result on success: **List of tasks**

- Result on failure: **false**

Request example to fetch all tasks on the board:

```
{
    "jsonrpc": "2.0",
    "method": "getOverdueTasks",
    "id": 133280317
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": [
        {
            "id": "1",
            "title": "Task #1",
            "date_due": "1409961789",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        },
        {
            "id": "2",
            "title": "Test",
            "date_due": "1409962115",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        }
    ]
}
```

5.15.6 getOverdueTasksByProject

- Purpose: **Get all overdue tasks for a special project**
- Result on success: **List of tasks**
- Result on failure: **false**

Request example to fetch all tasks on the board:

```
{
    "jsonrpc": "2.0",
    "method": "getOverdueTasksByProject",
    "id": 133280317,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": [
        {
            "id": "1",
            "title": "Task #1",
            "date_due": "1409961789",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        },
        {
            "id": "2",
            "title": "Test",
            "date_due": "1409962115",
            "project_id": "1",
            "project_name": "Test",
            "assignee_username":"admin",
            "assignee_name": null
        }
    ]
}
```

5.15.7 updateTask

- Purpose: **Update a task**
- Parameters:
  - **id** (integer, required)
  - **title** (string, optional)
  - **color_id** (string, optional)
  - **owner_id** (integer, optional)
  - **date_due**: ISO8601 format (string, optional)
  - **description** Markdown content (string, optional)
  - **category_id** (integer, optional)
  - **score** (integer, optional)
  - **priority** (integer, optional)
  - **recurrence_status** (integer, optional)
  - **recurrence_trigger** (integer, optional)
  - **recurrence_factor** (integer, optional)
  - **recurrence_timeframe** (integer, optional)
  - **recurrence_basedate** (integer, optional)
  - **reference** (string, optional)
  - **tags** ([]string, optional)

  - **date_started**: ISO8601 format (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example to change the task color:

```
{
    "jsonrpc": "2.0",
    "method": "updateTask",
    "id": 1406803059,
    "params": {
        "id": 1,
        "color_id": "blue"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1406803059,
    "result": true
}
```

5.15.8 openTask

- Purpose: **Set a task to the status open**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "openTask",
    "id": 1888531925,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1888531925,
    "result": true
}
```

5.15.9 closeTask

- Purpose: **Set a task to the status close**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "closeTask",
    "id": 1654396960,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1654396960,
    "result": true
}
```

5.15.10 removeTask

- Purpose: **Remove a task**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeTask",
    "id": 1423501287,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1423501287,
```

(continues on next page)

(continued from previous page)

```
    "result": true
}
```

5.15.11 moveTaskPosition

- Purpose: **Move a task to another column, position or swimlane inside the same board**
- Parameters:
  - **project_id** (integer, required)
  - **task_id** (integer, required)
  - **column_id** (integer, required)
  - **position** (integer, required)
  - **swimlane_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "moveTaskPosition",
    "id": 117211800,
    "params": {
        "project_id": 1,
        "task_id": 1,
        "column_id": 2,
        "position": 1,
        "swimlane_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 117211800,
    "result": true
}
```

5.15.12 moveTaskToProject

- Purpose: **Move a task to another project**
- Parameters:
  - **task_id** (integer, required)
  - **project_id** (integer, required)
  - **swimlane_id** (integer, optional)
  - **column_id** (integer, optional)

  - **category_id** (integer, optional)
  - **owner_id** (integer, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "moveTaskToProject",
    "id": 15775829,
    "params": [
        4,
        5
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 15775829,
    "result": true
}
```

5.15.13 duplicateTaskToProject

- Purpose: **Move a task to another column or another position**
- Parameters:
  - **task_id** (integer, required)
  - **project_id** (integer, required)
  - **swimlane_id** (integer, optional)
  - **column_id** (integer, optional)
  - **category_id** (integer, optional)
  - **owner_id** (integer, optional)
- Result on success: **task_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "duplicateTaskToProject",
    "id": 1662458687,
    "params": [
        5,
        7
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1662458687,
    "result": 6
}
```

5.15.14 searchTasks

- Purpose: **Find tasks by using the search engine**
- Parameters:
  - **project_id** (integer, required)
  - **query** (string, required)
- Result on success: **list of tasks**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "searchTasks",
    "id": 1468511716,
    "params": {
        "project_id": 2,
        "query": "assignee:nobody"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1468511716,
    "result": [
        {
            "nb_comments": "0",
            "nb_files": "0",
            "nb_subtasks": "0",
            "nb_completed_subtasks": "0",
            "nb_links": "0",
            "nb_external_links": "0",
            "is_milestone": null,
            "id": "3",
            "reference": "",
            "title": "T3",
            "description": "",
            "date_creation": "1461365164",
            "date_modification": "1461365164",
            "date_completed": null,
            "date_started": null,
            "date_due": "0",
            "color_id": "yellow",
```

(continues on next page)

(continued from previous page)

```
                "project_id": "2",
                "column_id": "5",
                "swimlane_id": "0",
                "owner_id": "0",
                "creator_id": "0"
            }
    ]
}
```

5.16 Task Metadata API Procedures

5.16.1 getTaskMetadata

- Purpose: **Get all metadata related to a task by task unique id**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **list of metadata**
- Result on failure: **empty array**

Request example to fetch all the metada of a task:

```
{
    "jsonrpc": "2.0",
    "method": "getTaskMetadata",
    "id": 133280317,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": [
        {
            "metaKey1": "metaValue1",
            "metaKey2": "metaValue2"
        }
    ]
}
```

5.16.2 getTaskMetadataByName

- Purpose: **Get metadata related to a task by task unique id and metakey (name)**
- Parameters:
  - **task_id** (integer, required)

  - **name** (string, required)
- Result on success: **metadata value**
- Result on failure: **empty string**

Request example to fetch metada of a task by name:

```
{
    "jsonrpc": "2.0",
    "method": "getTaskMetadataByName",
    "id": 133280317,
    "params": [
        1,
        "metaKey1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": "metaValue1"
}
```

5.16.3 saveTaskMetadata

- Purpose: **Save/update task metadata**
- Parameters:
  - **task_id** (integer, required)
  - **values** (array, required)
- Result on success: **true**
- Result on failure: **false**

Request example to add/update metada of a task:

```
{
    "jsonrpc": "2.0",
    "method": "saveTaskMetadata",
    "id": 133280317,
    "params": [
        1,
        {
            "metaName" : "metaValue"
        }
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
```

(continues on next page)

(continued from previous page)

```
    "result": true
}
```

5.16.4 removeTaskMetadata

- Purpose: **Remove task metadata by name**
- Parameters:
  - **task_id** (integer, required)
  - **name** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example to remove metada of a task by name:

```
{
    "jsonrpc": "2.0",
    "method": "removeTaskMetadata",
    "id": 133280317,
    "params": [
        1,
        "metaKey1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133280317,
    "result": true
}
```

5.17 Subtask API Procedures

5.17.1 createSubtask

- Purpose: **Create a new subtask**
- Parameters:
  - **task_id** (integer, required)
  - **title** (integer, required)
  - **user_id** (int, optional)
  - **time_estimated** (int, optional)
  - **time_spent** (int, optional)
  - **status** (int, optional)
- Result on success: **subtask_id**

- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createSubtask",
    "id": 2041554661,
    "params": {
        "task_id": 1,
        "title": "Subtask #1"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2041554661,
    "result": 45
}
```

5.17.2 getSubtask

- Purpose: **Get subtask information**
- Parameters:
  - **subtask_id** (integer)
- Result on success: **subtask properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getSubtask",
    "id": 133184525,
    "params": {
        "subtask_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 133184525,
    "result": {
        "id": "1",
        "title": "Subtask #1",
        "status": "0",
        "time_estimated": "0",
        "time_spent": "0",
        "task_id": "1",
        "user_id": "0"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

5.17.3 getAllSubtasks

- Purpose: **Get all available subtasks**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **List of subtasks**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllSubtasks",
    "id": 2087700490,
    "params": {
        "task_id": 1
    }
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2087700490,
    "result": [
        {
            "id": "1",
            "title": "Subtask #1",
            "status": "0",
            "time_estimated": "0",
            "time_spent": "0",
            "task_id": "1",
            "user_id": "0",
            "username": null,
            "name": null,
            "status_name": "Todo"
        }
    ]
}
```

5.17.4 updateSubtask

- Purpose: **Update a subtask**
- Parameters:
  - **id** (integer, required)
  - **task_id** (integer, required)
  - **title** (integer, optional)

  - **user_id** (integer, optional)
  - **time_estimated** (integer, optional)
  - **time_spent** (integer, optional)
  - **status** (integer, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateSubtask",
    "id": 191749979,
    "params": {
        "id": 1,
        "task_id": 1,
        "status": 1,
        "time_spent": 5,
        "user_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 191749979,
    "result": true
}
```

5.17.5 removeSubtask

- Purpose: **Remove a subtask**
- Parameters:
  - **subtask_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeSubtask",
    "id": 1382487306,
    "params": {
        "subtask_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1382487306,
    "result": true
}
```

5.18 Subtask Time Tracking API procedures

5.18.1 hasSubtaskTimer

- Purpose: **Check if a timer is started for the given subtask and user**
- Parameters:
  - **subtask_id** (integer, required)
  - **user_id** (integer, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"hasSubtaskTimer","id":1786995697,"params":[2,4]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1786995697
}
```

5.18.2 setSubtaskStartTime

- Purpose: **Start subtask timer for a user**
- Parameters:
  - **subtask_id** (integer, required)
  - **user_id** (integer, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"setSubtaskStartTime","id":1168991769,"params":[2,4]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1168991769
}
```

5.18.3 setSubtaskEndTime

- Purpose: **Stop subtask timer for a user**
- Parameters:
  - **subtask_id** (integer, required)
  - **user_id** (integer, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"setSubtaskEndTime","id":1026607603,"params":[2,4]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1026607603
}
```

5.18.4 getSubtaskTimeSpent

- Purpose: **Get time spent on a subtask for a user**
- Parameters:
  - **subtask_id** (integer, required)
  - **user_id** (integer, optional)
- Result on success: **number of hours**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"getSubtaskTimeSpent","id":738527378,"params":[2,4]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": 1.5,
    "id": 738527378
}
```

5.19 Comment API Procedures

5.19.1 createComment

- Purpose: **Create a new comment**
- Parameters:
  - **task_id** (integer, required)
  - **user_id** (integer, required)
  - **content** Markdown content (string, required)
- Result on success: **comment_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createComment",
    "id": 1580417921,
    "params": {
        "task_id": 1,
        "user_id": 1,
        "content": "Comment #1"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1580417921,
    "result": 11
}
```

5.19.2 getComment

- Purpose: **Get comment information**
- Parameters:
  - **comment_id** (integer, required)
- Result on success: **comment properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getComment",
    "id": 867839500,
    "params": {
        "comment_id": 1
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 867839500,
    "result": {
        "id": "1",
        "task_id": "1",
        "user_id": "1",
        "date_creation": "1410881970",
        "comment": "Comment #1",
        "username": "admin",
        "name": null
    }
}
```

5.19.3 getAllComments

- Purpose: **Get all available comments**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **List of comments**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllComments",
    "id": 148484683,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 148484683,
    "result": [
        {
            "id": "1",
            "date_creation": "1410882272",
            "task_id": "1",
            "user_id": "1",
            "comment": "Comment #1",
            "username": "admin",
            "name": null
        }
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

5.19.4 updateComment

- Purpose: **Update a comment**
- Parameters:
  - **id** (integer, required)
  - **content** Markdown content (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateComment",
    "id": 496470023,
    "params": {
        "id": 1,
        "content": "Comment #1 updated"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1493368950,
    "result": true
}
```

5.19.5 removeComment

- Purpose: **Remove a comment**
- Parameters:
  - **comment_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeComment",
    "id": 328836871,
    "params": {
        "comment_id": 1
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 328836871,
    "result": true
}
```

5.20 Automatic Actions API Procedures

5.20.1 getAvailableActions

- Purpose: **Get list of available automatic actions**
- Parameters: none
- Result on success: **list of actions**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAvailableActions",
    "id": 1217735483
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1217735483,
    "result": {
        "\Kanboard\Action\TaskLogMoveAnotherColumn": "Add a comment logging moving␣
↪the task between columns",
        "\Kanboard\Action\TaskAssignColorUser": "Assign a color to a specific user",
        "\Kanboard\Action\TaskAssignColorColumn": "Assign a color when the task is␣
↪moved to a specific column",
        "\Kanboard\Action\TaskAssignCategoryColor": "Assign automatically a category␣
↪based on a color",
        "\Kanboard\Action\TaskAssignColorCategory": "Assign automatically a color␣
↪based on a category",
        "\Kanboard\Action\TaskAssignSpecificUser": "Assign the task to a specific user
↪",
        "\Kanboard\Action\TaskAssignCurrentUser": "Assign the task to the person who␣
↪does the action",
        "\Kanboard\Action\TaskUpdateStartDate": "Automatically update the start date",
        "\Kanboard\Action\TaskAssignUser": "Change the assignee based on an external␣
↪username",
        "\Kanboard\Action\TaskAssignCategoryLabel": "Change the category based on an␣
↪external label",
```

(continues on next page)

(continued from previous page)

```
        "\Kanboard\Action\TaskClose": "Close a task",
        "\Kanboard\Action\CommentCreation": "Create a comment from an external␣
↪provider",
        "\Kanboard\Action\TaskCreation": "Create a task from an external provider",
        "\Kanboard\Action\TaskDuplicateAnotherProject": "Duplicate the task to␣
↪another project",
        "\Kanboard\Action\TaskMoveColumnAssigned": "Move the task to another column␣
↪when assigned to a user",
        "\Kanboard\Action\TaskMoveColumnUnAssigned": "Move the task to another column␣
↪when assignee is cleared",
        "\Kanboard\Action\TaskMoveAnotherProject": "Move the task to another project",
        "\Kanboard\Action\TaskOpen": "Open a task"
    }
}
```

5.20.2 getAvailableActionEvents

- Purpose: **Get list of available events for actions**
- Parameters: none
- Result on success: **list of events**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAvailableActionEvents",
    "id": 2116665643
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2116665643,
    "result": {
        "bitbucket.webhook.commit": "Bitbucket commit received",
        "task.close": "Closing a task",
        "github.webhook.commit": "Github commit received",
        "github.webhook.issue.assignee": "Github issue assignee change",
        "github.webhook.issue.closed": "Github issue closed",
        "github.webhook.issue.commented": "Github issue comment created",
        "github.webhook.issue.label": "Github issue label change",
        "github.webhook.issue.opened": "Github issue opened",
        "github.webhook.issue.reopened": "Github issue reopened",
        "gitlab.webhook.commit": "Gitlab commit received",
        "gitlab.webhook.issue.closed": "Gitlab issue closed",
        "gitlab.webhook.issue.opened": "Gitlab issue opened",
        "task.move.column": "Move a task to another column",
        "task.open": "Open a closed task",
        "task.assignee_change": "Task assignee change",
        "task.create": "Task creation",
        "task.create_update": "Task creation or modification",
```

(continues on next page)

(continued from previous page)

```
        "task.update": "Task modification"
    }
}
```

5.20.3 getCompatibleActionEvents

- Purpose: **Get list of events compatible with an action**
- Parameters:
  - **action_name** (string, required)
- Result on success: **list of events**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getCompatibleActionEvents",
    "id": 899370297,
    "params": [
        "\Kanboard\Action\TaskClose"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 899370297,
    "result": {
        "bitbucket.webhook.commit": "Bitbucket commit received",
        "github.webhook.commit": "Github commit received",
        "github.webhook.issue.closed": "Github issue closed",
        "gitlab.webhook.commit": "Gitlab commit received",
        "gitlab.webhook.issue.closed": "Gitlab issue closed",
        "task.move.column": "Move a task to another column"
    }
}
```

5.20.4 getActions

- Purpose: **Get list of actions for a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **list of actions properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getActions",
    "id": 1433237746,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": [
        {
            "id" : "13",
            "project_id" : "2",
            "event_name" : "task.move.column",
            "action_name" : "\Kanboard\Action\TaskAssignSpecificUser",
            "params" : {
                "column_id" : "5",
                "user_id" : "1"
            }
        }
    ]
}
```

5.20.5 createAction

- Purpose: **Create an action**
- Parameters:
  - **project_id** (integer, required)
  - **event_name** (string, required)
  - **action_name** (string, required)
  - **params** (key/value parameters, required)
- Result on success: **action_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createAction",
    "id": 1433237746,
    "params": {
        "project_id" : "2",
        "event_name" : "task.move.column",
        "action_name" : "\Kanboard\Action\TaskAssignSpecificUser",
        "params" : {
            "column_id" : "3",
```

(continues on next page)

(continued from previous page)

```
            "user_id" : "2"
        }
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1433237746,
    "result": 14
}
```

5.20.6 removeAction

- Purpose: **Remove an action**
- Parameters:
  - **action_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeAction",
    "id": 1510741671,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1510741671,
    "result": true
}
```

5.21 Category API Procedures

5.21.1 createCategory

- Purpose: **Create a new category**
- Parameters:
- **project_id** (integer, required)
  - **name** (string, required, must be unique for the given project)

- Result on success: **category_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createCategory",
    "id": 541909890,
    "params": {
        "name": "Super category",
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 541909890,
    "result": 4
}
```

5.21.2 getCategory

- Purpose: **Get category information**
- Parameters:
  - **category_id** (integer, required)
- Result on success: **category properties**
- Result on failure: **null**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getCategory",
    "id": 203539163,
    "params": {
        "category_id": 1
    }
}
```

Response example:

```
{

    "jsonrpc": "2.0",
    "id": 203539163,
    "result": {
        "id": "1",
        "name": "Super category",
        "project_id": "1"
    }
}
```

5.21.3 getAllCategories

- Purpose: **Get all available categories**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **List of categories**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllCategories",
    "id": 1261777968,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1261777968,
    "result": [
        {
            "id": "1",
            "name": "Super category",
            "project_id": "1"
        }
    ]
}
```

5.21.4 updateCategory

- Purpose: **Update a category**
- Parameters:
  - **id** (integer, required)
  - **name** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateCategory",
    "id": 570195391,
    "params": {
        "id": 1,
        "name": "Renamed category"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 570195391,
    "result": true
}
```

5.21.5 removeCategory

- Purpose: **Remove a category**
- Parameters:
  - **category_id** (integer)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeCategory",
    "id": 88225706,
    "params": {
        "category_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 88225706,
    "result": true
}
```

5.22 External Task Link API Procedures

5.22.1 getExternalTaskLinkTypes

- Purpose: **Get all registered external link providers**
- Parameters: **none**
- Result on success: **dict**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"getExternalTaskLinkTypes","id":477370568}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": {
        "auto": "Auto",
        "attachment": "Attachment",
        "file": "Local File",
        "weblink": "Web Link"
    },
    "id": 477370568
}
```

5.22.2 getExternalTaskLinkProviderDependencies

- Purpose: **Get available dependencies for a given provider**
- Parameters:
  - **providerName** (string, required)
- Result on success: **dict**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"getExternalTaskLinkProviderDependencies","id":124790226,
↪"params":["weblink"]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": {
        "related": "Related"
    },
    "id": 124790226
}
```

5.22.3 createExternalTaskLink

- Purpose: **Create a new external link**
- Parameters:
  - **task_id** (integer, required)
  - **url** (string, required)
  - **dependency** (string, required)
  - **type** (string, optional)
  - **title** (string, optional)
- Result on success: **link_id**

- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"createExternalTaskLink","id":924217495,"params":[9,"http:\/
↪\/localhost\/document.pdf","related","attachment"]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": 1,
    "id": 924217495
}
```

5.22.4 updateExternalTaskLink

- Purpose: **Update external task link**
- Parameters:
  - **task_id** (integer, required)
  - **link_id** (integer, required)
  - **title** (string, required)
  - **url** (string, required)
  - **dependency** (string, optional)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc":"2.0",
    "method":"updateExternalTaskLink",
    "id":1123562620,
    "params": {
        "task_id":9,
        "link_id":1,
        "title":"New title"
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1123562620
}
```

5.22.5 getExternalTaskLinkById

- Purpose: **Get an external task link**

- Parameters:
  - **task_id** (integer, required)
  - **link_id** (integer, required)
- Result on success: **dict**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"getExternalTaskLinkById","id":2107066744,"params":[9,1]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": {
        "id": "1",
        "link_type": "attachment",
        "dependency": "related",
        "title": "document.pdf",
        "url": "http:\/\/localhost\/document.pdf",
        "date_creation": "1466965256",
        "date_modification": "1466965256",
        "task_id": "9",
        "creator_id": "0"
    },
    "id": 2107066744
}
```

5.22.6 getAllExternalTaskLinks

- Purpose: **Get all external links attached to a task**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **list of external links**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"getAllExternalTaskLinks","id":2069307223,"params":[9]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "id": "1",
            "link_type": "attachment",
            "dependency": "related",
            "title": "New title",
            "url": "http:\/\/localhost\/document.pdf",
            "date_creation": "1466965256",
```

(continues on next page)

(continued from previous page)

```
            "date_modification": "1466965256",
            "task_id": "9",
            "creator_id": "0",
            "creator_name": null,
            "creator_username": null,
            "dependency_label": "Related",
            "type": "Attachment"
        }
    ],
    "id": 2069307223
}
```

5.22.7 removeExternalTaskLink

- Purpose: **Remove an external link**
- Parameters:
  - **task_id** (integer, required)
  - **link_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"removeExternalTaskLink","id":552055660,"params":[9,1]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 552055660
}
```

5.23 Internal Task Link API Procedures

5.23.1 createTaskLink

- Purpose: **Create a link between two tasks**
- Parameters:
  - **task_id** (integer, required)
  - **opposite_task_id** (integer, required)
  - **link_id** (integer, required)
- Result on success: **task_link_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createTaskLink",
    "id": 509742912,
    "params": [
        2,
        3,
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 509742912,
    "result": 1
}
```

5.23.2 updateTaskLink

- Purpose: **Update task link**
- Parameters:
  - **task_link_id** (integer, required)
  - **task_id** (integer, required)
  - **opposite_task_id** (integer, required)
  - **link_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateTaskLink",
    "id": 669037109,
    "params": [
        1,
        2,
        4,
        2
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 669037109,
    "result": true
}
```

5.23.3 getTaskLinkById

- Purpose: **Get a task link**
- Parameters:
  - **task_link_id** (integer, required)
- Result on success: **task link properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getTaskLinkById",
    "id": 809885202,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 809885202,
    "result": {
        "id": "1",
        "link_id": "1",
        "task_id": "2",
        "opposite_task_id": "3"
    }
}
```

5.23.4 getAllTaskLinks

- Purpose: **Get all links related to a task**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **list of task link**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllTaskLinks",
    "id": 810848359,
    "params": [
        2
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 810848359,
    "result": [
        {
            "id": "1",
            "task_id": "3",
            "label": "relates to",
            "title": "B",
            "is_active": "1",
            "project_id": "1",
            "task_time_spent": "0",
            "task_time_estimated": "0",
            "task_assignee_id": "0",
            "task_assignee_username": null,
            "task_assignee_name": null,
            "column_title": "Backlog"
        }
    ]
}
```

5.23.5 removeTaskLink

- Purpose: **Remove a link between two tasks**
- Parameters:
  - **task_link_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeTaskLink",
    "id": 473028226,
    "params": [
        1
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 473028226,
    "result": true
}
```

5.24 Link API Procedures

5.24.1 getAllLinks

- Purpose: **Get the list of possible relations between tasks**
- Parameters: none
- Result on success: **List of links**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllLinks",
    "id": 113057196
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 113057196,
    "result": [
        {
            "id": "1",
            "label": "relates to",
            "opposite_id": "0"
        },
        {
            "id": "2",
            "label": "blocks",
            "opposite_id": "3"
        },
        {
            "id": "3",
            "label": "is blocked by",
            "opposite_id": "2"
        },
        {
            "id": "4",
            "label": "duplicates",
            "opposite_id": "5"
        },
        {
            "id": "5",
            "label": "is duplicated by",
            "opposite_id": "4"
        },
        {
            "id": "6",
            "label": "is a child of",
            "opposite_id": "7"
        },
        {
            "id": "7",
```

(continues on next page)

(continued from previous page)

```
            "label": "is a parent of",
            "opposite_id": "6"
        },
        {
            "id": "8",
            "label": "targets milestone",
            "opposite_id": "9"
        },
        {
            "id": "9",
            "label": "is a milestone of",
            "opposite_id": "8"
        },
        {
            "id": "10",
            "label": "fixes",
            "opposite_id": "11"
        },
        {
            "id": "11",
            "label": "is fixed by",
            "opposite_id": "10"
        }
    ]
}
```

5.24.2 getOppositeLinkId

- Purpose: **Get the opposite link id of a task link**
- Parameters:
  - **link_id** (integer, required)
- Result on success: **link_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getOppositeLinkId",
    "id": 407062448,
    "params": [
        2
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 407062448,
    "result": "3"
}
```

5.24.3 getLinkByLabel

- Purpose: **Get a link by label**
- Parameters:
  - **label** (integer, required)
- Result on success: **link properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getLinkByLabel",
    "id": 1796123316,
    "params": [
        "blocks"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1796123316,
    "result": {
        "id": "2",
        "label": "blocks",
        "opposite_id": "3"
    }
}
```

5.24.4 getLinkById

- Purpose: **Get a link by id**
- Parameters:
  - **link_id** (integer, required)
- Result on success: **link properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getLinkById",
    "id": 1190238402,
    "params": [
        4
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1190238402,
    "result": {
        "id": "4",
        "label": "duplicates",
        "opposite_id": "5"
    }
}
```

5.24.5 createLink

- Purpose: **Create a new task relation**
- Parameters:
  - **label** (integer, required)
  - **opposite_label** (integer, optional)
- Result on success: **link_id**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createLink",
    "id": 1040237496,
    "params": [
        "foo",
        "bar"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1040237496,
    "result": 13
}
```

5.24.6 updateLink

- Purpose: **Update a link**
- Parameters:
  - **link_id** (integer, required)
  - **opposite_link_id** (integer, required)
  - **label** (string, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "updateLink",
    "id": 2110446926,
    "params": [
        "14",
        "12",
        "boo"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2110446926,
    "result": true
}
```

5.24.7 removeLink

- Purpose: **Remove a link**
- Parameters:
  - **link_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeLink",
    "id": 2136522739,
    "params": [
        "14"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 2136522739,
    "result": true
}
```

5.25 Project File API Procedures

5.25.1 createProjectFile

- Purpose: **Create and upload a new project attachment**
- Parameters:
  - **project_id** (integer, required)
  - **filename** (integer, required)
  - **blob** File content encoded in base64 (string, required)
- Result on success: **file_id**
- Result on failure: **false**
- Note: **The maximum file size depends of your PHP configuration, this method should not be used to upload large files**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createProjectFile",
    "id": 94500810,
    "params": [
        1,
        "My file",
        "cGxhaW4gdGV4dCBmaWxl"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 94500810,
    "result": 1
}
```

5.25.2 getAllProjectFiles

- Purpose: **Get all files attached to a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **list of files**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllProjectFiles",
```

(continues on next page)

(continued from previous page)

```
    "id": 1880662820,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1880662820,
    "result": [
        {
            "id": "1",
            "name": "My file",
            "path": "1\/1\/0db4d0a897a4c852f6e12f0239d4805f7b4ab596",
            "is_image": "0",
            "project_id": "1",
            "date": "1432509941",
            "user_id": "0",
            "size": "15",
            "username": null,
            "user_name": null
        }
    ]
}
```

5.25.3 getProjectFile

- Purpose: **Get file information**
- Parameters:
  - **project_id** (integer, required)
  - **file_id** (integer, required)
- Result on success: **file properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getProjectFile",
    "id": 318676852,
    "params": [
        "42",
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
```

(continues on next page)

(continued from previous page)

```
    "id": 318676852,
    "result": {
        "id": "1",
        "name": "My file",
        "path": "1\/1\/0db4d0a897a4c852f6e12f0239d4805f7b4ab596",
        "is_image": "0",
        "project_id": "1",
        "date": "1432509941",
        "user_id": "0",
        "size": "15"
    }
}
```

5.25.4 downloadProjectFile

- Purpose: **Download project file contents (encoded in base64)**
- Parameters:
  - **project_id** (integer, required)
  - **file_id** (integer, required)
- Result on success: **base64 encoded string**
- Result on failure: **empty string**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "downloadProjectFile",
    "id": 235943344,
    "params": [
        "1",
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 235943344,
    "result": "cGxhaW4gdGV4dCBmaWxl"
}
```

5.25.5 removeProjectFile

- Purpose: **Remove a file associated to a project**
- Parameters:
  - **project_id** (integer, required)
  - **file_id** (integer, required)
- Result on success: **true**

- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeProjectFile",
    "id": 447036524,
    "params": [
        "1",
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 447036524,
    "result": true
}
```

5.25.6 removeAllProjectFiles

- Purpose: **Remove all files associated to a project**
- Parameters:
  - **project_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeAllProjectFiles",
    "id": 593312993,
    "params": {
        "project_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 593312993,
    "result": true
}
```

5.26 Task File API Procedures

5.26.1 createTaskFile

- Purpose: **Create and upload a new task attachment**
- Parameters:
  - **project_id** (integer, required)
  - **task_id** (integer, required)
  - **filename** (integer, required)
  - **blob** File content encoded in base64 (string, required)
- Result on success: **file_id**
- Result on failure: **false**
- Note: **The maximum file size depends of your PHP configuration, this method should not be used to upload large files**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "createTaskFile",
    "id": 94500810,
    "params": [
        1,
        1,
        "My file",
        "cGxhaW4gdGV4dCBmaWxl"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 94500810,
    "result": 1
}
```

5.26.2 getAllTaskFiles

- Purpose: **Get all files attached to task**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **list of files**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getAllTaskFiles",
    "id": 1880662820,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 1880662820,
    "result": [
        {
            "id": "1",
            "name": "My file",
            "path": "1\/1\/0db4d0a897a4c852f6e12f0239d4805f7b4ab596",
            "is_image": "0",
            "task_id": "1",
            "date": "1432509941",
            "user_id": "0",
            "size": "15",
            "username": null,
            "user_name": null
        }
    ]
}
```

5.26.3 getTaskFile

- Purpose: **Get file information**
- Parameters:
  - **file_id** (integer, required)
- Result on success: **file properties**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "getTaskFile",
    "id": 318676852,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
```

(continues on next page)

(continued from previous page)

```
    "id": 318676852,
    "result": {
        "id": "1",
        "name": "My file",
        "path": "1\/1\/0db4d0a897a4c852f6e12f0239d4805f7b4ab596",
        "is_image": "0",
        "task_id": "1",
        "date": "1432509941",
        "user_id": "0",
        "size": "15"
    }
}
```

5.26.4 downloadTaskFile

- Purpose: **Download file contents (encoded in base64)**
- Parameters:
  - **file_id** (integer, required)
- Result on success: **base64 encoded string**
- Result on failure: **empty string**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "downloadTaskFile",
    "id": 235943344,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 235943344,
    "result": "cGxhaW4gdGV4dCBmaWxl"
}
```

5.26.5 removeTaskFile

- Purpose: **Remove file**
- Parameters:
  - **file_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeTaskFile",
    "id": 447036524,
    "params": [
        "1"
    ]
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 447036524,
    "result": true
}
```

5.26.6 removeAllTaskFiles

- Purpose: **Remove all files associated to a task**
- Parameters:
  - **task_id** (integer, required)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{
    "jsonrpc": "2.0",
    "method": "removeAllTaskFiles",
    "id": 593312993,
    "params": {
        "task_id": 1
    }
}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "id": 593312993,
    "result": true
}
```

5.27 Tags API Procedures

5.27.1 getAllTags

- Purpose: **Get all tags**
- Parameters: none

- Result on success: **List of tags**
- Result on failure: **false|null**

Request example:

```
{"jsonrpc":"2.0","method":"getAllTags","id":45253426}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "id": "1",
            "name": "another tag",
            "project_id": "33"
        }
    ],
    "id": 45253426
}
```

5.27.2 getTagsByProject

- Purpose: **Get all tags for a given project**
- Parameters:
  - **project_id** (integer)
- Result on success: **List of tags**
- Result on failure: **false|null**

Request example:

```
{"jsonrpc":"2.0","method":"getTagsByProject","id":1217591720,"params":[33]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "id": "1",
            "name": "some tag",
            "project_id": "33"
        }
    ],
    "id": 1217591720
}
```

5.27.3 createTag

- Purpose: **Create a new tag**
- Parameters:
  - **project_id** (integer)

    - **tag** (string)

- Result on success: **tag_id**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"createTag","id":1775436017,"params":[33,"some tag"]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": 1,
    "id": 1775436017
}
```

5.27.4 updateTag

- Purpose: **Rename a tag**
- Parameters:
    - **tag_id** (integer)
    - **tag** (string)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"updateTag","id":2037516512,"params":["1","another tag"]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 2037516512
}
```

5.27.5 removeTag

- Purpose: **removeTag**
- Parameters:
    - **tag_id** (integer)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"removeTag","id":907581298,"params":["1"]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 907581298
}
```

5.27.6 setTaskTags

- Purpose: **Assign/Create/Update tags for a task**
- Parameters:
  - **project_id** (integer)
  - **task_id** (integer)
  - **tags** List of tags ([]string)
- Result on success: **true**
- Result on failure: **false**

Request example:

```
{"jsonrpc":"2.0","method":"setTaskTags","id":1524522873,"params":[39,17,["tag1","tag2
↪"]]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1524522873
}
```

5.27.7 getTaskTags

- Purpose: **Get assigned tags to a task**
- Parameters:
  - **task_id** (integer)
- Result on success: **Dictionary of tags**
- Result on failure: **false|null**

Request example:

```
{"jsonrpc":"2.0","method":"getTaskTags","id":1667157705,"params":[17]}
```

Response example:

```
{
    "jsonrpc": "2.0",
    "result": {
        "1": "tag1",
```

(continues on next page)

(continued from previous page)

```
        "2": "tag2"
    },
    "id": 1667157705
}
```