Taiga REST API

Table of Contents

1. General notes

NOTE

About Taiga instance and URLs used in this document

All API calls used in the documentation are refered to Taiga.io instance, so if you use another instance remember to change the url.

For example, if you have installed Taiga on your own PC, you must perform the tests using *http://localhost:8000/api/v1* instead of *https://taiga.io/api/v1*.

1.1. Authentication

1.1.1. Standard token authentication

To authenticate requests an http header called "Authorization" should be added. Its format should be:

```
Authorization: Bearer ${AUTH_TOKEN}
```

This token can be received through the login API

To provide an example, the following can be used within a Bash script running on Ubuntu - customise as appropriate for your system configuration.

- Install `jq` (a command-line JSON processor):

```
$ sudo apt-get install jq
```

- Bash snippet:

```
#!/bin/bash
# Request username and password for connecting to Taiga
read -p "Username or email: " USERNAME
read -s -p "Password: " PASSWORD

# Get AUTH_TOKEN
USER_AUTH_DETAIL=$( curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
        "type": "normal",
        "username": "'${USERNAME}'",
        "password": "'${PASSWORD}'"
    }' \
  https://api.taiga.io/api/v1/auth 2>/dev/null )

AUTH_TOKEN=$( echo ${USER_AUTH_DETAIL} | jq -r '.auth_token' )

# Exit if AUTH_TOKEN is not available
if [ -z ${AUTH_TOKEN} ]; then
    echo "Error: Incorrect username and/or password supplied"
    exit 1
else
    echo "auth_token is ${AUTH_TOKEN}"
fi

# Proceed to use API calls as desired
...
```

- If unable to install `jq`, it is possible (but not recommended) to use `grep` and `cut` to extract the value of `auth_token` from the JSON user auth detail object - use the following line instead:

```
AUTH_TOKEN=$( echo ${USER_AUTH_DETAIL} | grep -Po '"auth_token":.*?[^\\]",' | cut -d\"
-f4 )
```

1.1.2. Application token authentication

This kind of tokes are designed for allowing external apps use the Taiga API, they are associated to an existing user and an Application. They can be manually created via the django ADMIN or programatically created via API.

They work in the same way than standard Taiga authentication tokens but the "Authorization" header change slightly. Its format should be:

```
Authorization: Application ${AUTH_TOKEN}
```

The process for obtaining a valid token consists in:

- Checking if there is an existing application token for the requesting user
- Requesting an authorization code for the requesting user if it doesn't exist yet
- Validating the authorization code to obtain the final token
- Decyphering the token

Checking if there is an existing application token for the requesting user

A GET request must be done to the applications resource including the application id in the url and specifying the token endpoint:

```
curl -X GET \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer {AUTH_TOKEN}" \
    https://api.taiga.io/api/v1/applications/5c8515c2-4fc4-11e5-9a5e-68f72800aadd/token
```

The API will answer with info about the application and the token:

```
{
    "user": 4,
    "id": null,
    "application": {
        "id": "a60c3208-5234-11e5-96df-68f72800aadd",
        "name": "Testing application",
        "web": "http://taiga.io",
        "description": "Testing external app",
        "icon_url": "https://tree.taiga.io/images/beta.png"
    },
    "auth_code": null,
    "next_url": "http://tree.taiga.io/redirect?auth_code=None"
}
```

If id and auth_code are null it means there is no application token generated and you need to authorize one. If they are not null you can jump to the validation step.

Requesting an authorization code for the requesting user if it doesn't exist yet

The request should include:

- application: the application id for the requested token

- state: an unguessable random string. It is used to protect against cross-site request forgery attacks. The API will include this value when the validation process is completed so the final app can verify it matches the original one.

```
curl -X POST \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer
eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjo0fQ:1ZX33b:QnAN3EcuChLoRVf3CdybWEi2OEg" \
    -d '{
          "application": "a60c3208-5234-11e5-96df-68f72800aadd",
          "state": "random-state"
      }' \
    https://api.taiga.io/api/v1/application-tokens/authorize
```

The API answer will be something like:

```
{
    "auth_code": "c8bfacba-5236-11e5-b8f6-68f72800aadd",
    "state": "random-state",
    "next_url": "asd?auth_code=c8bfacba-5236-11e5-b8f6-68f72800aadd"
}
```

The obtained auth_code must be validated as described in the validation step.

Validating the authorization code to obtain the final token

Now the external app must validate the auth_code obtained in the previous steps with a request including:

- application: the application id for the requested token
- state: an unguessable random string. It is used to protect against cross-site request forgery attacks. The API will include this value when the validation process is completed so the final app can verify it matches the original one.
- auth_code: the authorization code received on previous the steps.

```
curl -X POST \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer
eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjo0fQ:1ZX33b:QnAN3EcuChLoRVf3CdybWEi2OEg" \
    -d '{
          "application": "a60c3208-5234-11e5-96df-68f72800aadd",
          "auth_code": "21ce08c4-5237-11e5-a8a3-68f72800aadd",
          "state": "random-state"
      }' \
https://api.taiga.io/api/v1/application-tokens/validate
```

The API answer will be something like:

```
{
    "cyphered_token": "eyJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiQTEyOEtXIn0.E-Ee1cRgG0JEd90yJu-
Dgl_vwKHTHdPy2YHRbCsMvfiJx0OvR12E8g.kGwJPnWQJecFPEae.ebQtpRNPbKh6FBS-
LSUhw1xNARl0Q5loCO4fAk00LHFqcDpAwba7LHeR3MPx9T9LfA.KM-Id_041g8OdWaseGyV8g"
}
```

Decyphering the token

The token is cyphered using JWE with A128KW as algorythm and A256GCM as encryption. Both parts (Taiga and the external application requesting the token) must know about the encryption key used in the process (in Taiga it's an attribute of the application model).

- A python snippet for decyphering the token:

```
from jwkest.jwk import SYMKey
from jwkest.jwe import JWE
key ="this-is-the-secret-key"
cyphered_token="eyJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiQTEyOEtXIn0.H5jWzzXQISSh_QPCO5mWhT0EI9RRV
45xA7vbWoxeBIjiCL3qwAmlzg.bBWVKwGTkta5y99c.AryfFtrlmWgyZ4lwXw_JiSVmkn9YF6Xwlh8nVDku0BLW8
kvaxNy3XRbbb17MtZ7mg.pDkpgDwffCyCy4sYNQI6zA"
sym_key = SYMKey(key=key, alg="A128KW")
token=JWE().decrypt(cyphered_token, keys=[sym_key])
print(token)
```

When decyphering it correctly you will obtain a json containing the application token that can be used in the Authorization headers

```
{
    "token": "95db1710-5238-11e5-a86e-68f72800aadd"
}
```

1.2. OCC - Optimistic concurrency control

In taiga multiple operations can be happening at the same time for an element so every modifying request should include a valid version parameter. You can think about two different users updating the same user story, there are two possible scenarios here:

- They are updating the same attributes on the element. In this situation the API will accept the first request and deny the second one because the version parameter will be considered as invalid.
- They are updating different attributes on the element. In this situation the API is smart enough for accepting both requests, the second one would have an invalid version but the changes are not affecting modified attributes so they can be applied safely

The version parameter is considered valid if it contains the current version for the element, it will be incremented automatically if the modification is successful.

1.3. Pagination

By default the API will always return paginated results and includes the following headers in the response:

- x-paginated: boolean indicating if pagination is being used for the request
- x-paginated-by: number of results per page
- x-pagination-count: total number of results
- x-pagination-current: current page
- x-pagination-next: next results
- x-pagination-prev: previous results

Disabling pagination can be accomplished by setting an extra http header:

```
x-disable-pagination: True
```

1.4. Internationalization

The API returns some content translated, you can specify the language with an extra http header:

```
Accept-Language: {LanguageId}
```

The LanguageId can be chosen from the value list of available languages. You can get them using the locales API.

1.5. Throttling

If the api is configured with throttling you have to take care on responses with 429 (Too many requests) status code, that mean you reach the throttling limit.

2. Endpoints Summary

2.1. Auth

URL	Method	Functionality
/api/v1/auth	POST	Login
/api/v1/auth/register	POST	Register user

2.2. Applications

URL	Method	Functionality
/api/v1/applications/{applicationId}	GET	Get application
/api/v1/applications/{applicationId}/token	GET	Get application token

2.3. Application Tokens

URL	Method	Functionality
/api/v1/application-tokens	GET	List application tokens
/api/v1/application-tokens/{applicationTokenId}	GET	Get application token
/api/v1/application-tokens/{applicationTokenId}	DELETE	Delete application token
/api/v1/application-tokens/authorize	POST	Authorize application token

URL	Method	Functionality
/api/v1/application-tokens/validate	POST	Validate application token

2.4. Resolver

URL	Method	Functionality
/api/v1/resolver	GET	Resolve references and slugs

2.5. Searches

URL	Method	Functionality
/api/v1/search	GET	Search in a project

2.6. User storage

URL	Method	Functionality
/api/v1/user-storage	GET	List user storage data
/api/v1/user-storage	POST	Create user storage data
/api/v1/user-storage/{key}	GET	Get user storage data
/api/v1/user-storage/{key}	PUT	Modify user storage data
/api/v1/user-storage/{key}	PATCH	Modify partially an user storage data
/api/v1/user-storage/{key}	DELETE	Delete user storage data

2.7. Project templates

URL	Method	Functionality
/api/v1/project-templates	GET	List project templates
/api/v1/project-templates	POST	Create project template
/api/v1/project-templates/{projectTemplateId}	GET	Get project template
/api/v1/project-templates/{projectTemplateId}	PUT	Modify project template

URL	Method	Functionality
/api/v1/project-templates/{projectTemplateId}	PATCH	Modify partially an project template
/api/v1/project-templates/{projectTemplateId}	DELETE	Delete an project template

2.8. Projects

URL	Method	Functionality
/api/v1/projects	GET	List projects
/api/v1/projects	POST	Create project
/api/v1/projects/{projectId}	GET	Get project
/api/v1/projects/by_slug?slug={projectSlug}	GET	Get project
/api/v1/projects/{projectId}	PUT	Modify project
/api/v1/projects/{projectId}	PATCH	Modify partially a project
/api/v1/projects/{projectId}	DELETE	Delete a project
/api/v1/projects/bulk_update_order	POST	Update projects order for logged in user
/api/v1/projects/{projectId}/modules	GET	Get project modules configuration
/api/v1/projects/{projectId}/modules	PATCH	Modify partially a project modules configuration
/api/v1/projects/{projectId}/stats	GET	Get project stats
/api/v1/projects/{projectId}/issues_stats	GET	Get project issue stats
/api/v1/projects/{projectId}/issue_filters_data	GET	Get project issue filters data
/api/v1/projects/{projectId}/tags_colors	GET	Get project tags colors
/api/v1/projects/{projectId}/like	POST	Like a project
/api/v1/projects/{projectId}/unlike	POST	Unlike a project
/api/v1/projects/{projectId}/fans	GET	Get project fans

URL	Method	Functionality
/api/v1/projects/{projectId}/watch	POST	Watch a project
/api/v1/projects/{projectId}/unwatch	POST	Unwatch a project
/api/v1/projects/{projectId}/watchers	GET	Get project watchers
/api/v1/projects/{projectId}/create_template	POST	Create project template
/api/v1/projects/{projectId}/leave	POST	Leave project

2.9. Memberships/Invitations

URL	Method	Functionality
/api/v1/memberships	GET	List memberships
/api/v1/memberships	POST	Create membership
/api/v1/memberships/bulk_create	POST	Create a bulk of memberships
/api/v1/memberships/{membershipId}	GET	Get membership
/api/v1/memberships/{membershipId}	PUT	Modify membership
/api/v1/memberships/{membershipId}	PATCH	Modify partially a membership
/api/v1/memberships/{membershipId}	DELETE	Delete a membership
/api/v1/memberships/{membershipId}/resend_invitation	POST	Resend invitation
/api/v1/invitations/{invitationUuid}	POST	Get invitation by anonymous user

2.10. Milestones

URL	Method	Functionality
/api/v1/milestones	GET	List milestones
/api/v1/milestones	POST	Create milestone

URL	Method	Functionality
/api/v1/milestones/{milestoneId}	GET	Get milestone
/api/v1/milestones/{milestoneId}	PUT	Modify milestone
/api/v1/milestones/{milestoneId}	PATCH	Modify partially a milestone
/api/v1/milestones/{milestoneId}	DELETE	Delete a milestone
/api/v1/milestones/{milestoneId}/stats	GET	Get a milestone stats
/api/v1/milestones/{milestoneId}/watch	POST	Watch a milestone
/api/v1/milestones/{milestoneId}/unwatch	POST	Stop watching a milestone
/api/v1/milestones/{milestoneId}/watchers	GET	Get milestone watchers

2.11. User stories

URL	Method	Functionality
/api/v1/userstories	GET	List user stories
/api/v1/userstories	POST	Create user story
/api/v1/userstories/{userStoryId}	GET	Get user story
/api/v1/userstories/by_ref?ref={userStoryRef}&project={userStoryId}	GET	Get user story
/api/v1/userstories/{userStoryId}	PUT	Modify user story
/api/v1/userstories/{userStoryId}	PATCH	Modify partially a user story
/api/v1/userstories/{userStoryId}	DELETE	Delete a user story
/api/v1/userstories/bulk_create	POST	Create user stories un bulk mode
/api/v1/userstories/bulk_update_backlog_order	POST	Update user stories order for backlog in bulk mode
/api/v1/userstories/bulk_update_kanban_order	POST	Update user stories order for kanban in bulk mode
/api/v1/userstories/bulk_update_sprint_order	POST	Update user stories order for sprint in bulk mode
/api/v1/userstories/{userStoryId}/upvote	POST	Add star to a user story

URL	Method	Functionality
/api/v1/userstories/{userStoryId}/downvote	POST	Remove star from user story
/api/v1/userstories/{userStoryId}/voters	GET	Get user story voters
/api/v1/userstories/{userStoryId}/watch	POST	Watch a user story
/api/v1/userstories/{userStoryId}/unwatch	POST	Unwatch a user story
/api/v1/userstories/{userStoryId}/watchers	GET	Get user story watchers
/api/v1/userstories/attachments	GET	List user story attachments
/api/v1/userstories/attachments	POST	Create user story attachments
/api/v1/userstories/attachments/{userStoryAttachmentId}	GET	Get user story attachments
/api/v1/userstories/attachments/{userStoryAttachmentId}	PUT	Modify user story attachments
/api/v1/userstories/attachments/{userStoryAttachmentId}	PATCH	Modify partially a user story attachments
/api/v1/userstories/attachments/{userStoryAttachmentId}	DELETE	Delete a user story attachments

2.12. User story status

URL	Method	Functionality
/api/v1/userstory-statuses	GET	List user story status
/api/v1/userstory-statuses	POST	Create user story status
/api/v1/userstory-statuses/{userStoryStatusId}	GET	Get user story status
/api/v1/userstory-statuses/{userStoryStatusId}	PUT	Modify user story status
/api/v1/userstory-statuses/{userStoryStatusId}	PATCH	Modify partially a user story status
/api/v1/userstory-statuses/{userStoryStatusId}	DELETE	Delete a user story status

URL	Method	Functionality
/api/v1/userstory-statuses/bulk_update_order	POST	Update user story statuses order in bulk mode

2.13. Points

URL	Method	Functionality
/api/v1/points	GET	List points
/api/v1/points	POST	Create point
/api/v1/points/{pointId}	GET	Get point
/api/v1/points/{pointId}	PUT	Modify point
/api/v1/points/{pointId}	PATCH	Modify partially a point
/api/v1/points/{pointId}	DELETE	Delete a point
/api/v1/points/bulk_update_order	POST	Update points order in bulk mode

2.14. User story custom attribute

URL	Method	Functionality
/api/v1/userstory-custom-attributes	GET	List user story custom attributes
/api/v1/userstory-custom-attributes	POST	Create user story custom attribute
/api/v1/userstory-custom-attributes/{userStoryCustomAttributeId}	GET	Get user story custom attribute
/api/v1/userstory-custom-attributes/{userStoryCustomAttributeId}	PUT	Modify user story custom attribute
/api/v1/userstory-custom-attributes/{userStoryCustomAttributeId}	PATCH	Modify partially a user story custom attribute
/api/v1/userstory-custom-attributes/{userStoryCustomAttributeId}	DELETE	Delete a user story custom attribute

URL	Method	Functionality
/api/v1/userstory-custom-attributes/bulk_update_order	POST	Update user story custom attributes order in bulk mode

2.15. User story custom attributes values

URL	Method	Functionality
/api/v1/userstories/custom-attributes-values/{userStoryId}	GET	Get user story custom attributes values
/api/v1/userstories/custom-attributes-values/{userStoryId}	PUT	Modify user story custom attributes values
/api/v1/userstories/custom-attributes-values/{userStoryId}	PATCH	Modify partially a user story custom attributes values

2.16. Tasks

URL	Method	Functionality
/api/v1/tasks	GET	List tasks
/api/v1/tasks	POST	Create task
/api/v1/tasks/{taskId}	GET	Get task
/api/v1/tasks/by_ref?ref={taskRef}&project={projectId}	GET	Get task
/api/v1/tasks/{taskId}	PUT	Modify task
/api/v1/tasks/{taskId}	PATCH	Modify partially a task
/api/v1/tasks/{taskId}	DELETE	Delete a task
/api/v1/tasks/bulk_create	POST	Create tasks on bulk mode
/api/v1/tasks/{taskId}/upvote	POST	Add star to a task
/api/v1/tasks/{taskId}/downvote	POST	Remove star from task
/api/v1/tasks/{taskId}/voters	GET	Get task voters
/api/v1/tasks/{taskId}/watch	POST	Watch a task
/api/v1/tasks/{taskId}/unwatch	POST	Unwatch a task
/api/v1/tasks/{taskId}/watchers	GET	Get task watchers
/api/v1/tasks/attachments	GET	List task attachments

URL	Method	Functionality
/api/v1/tasks/attachments	POST	Create task attachments
/api/v1/tasks/attachments/{taskAttachmentId}	GET	Get task attachments
/api/v1/tasks/attachments/{taskAttachmentId}	PUT	Modify task attachments
/api/v1/tasks/attachments/{taskAttachmentId}	PATCH	Modify partially a task attachments
/api/v1/tasks/attachments/{taskAttachmentId}	DELETE	Delete a task attachments

2.17. Task status

URL	Method	Functionality
/api/v1/task-statuses	GET	List task statuses
/api/v1/task-statuses	POST	Create task status
/api/v1/task-statuses/{taskStatusId}	GET	Get task status
/api/v1/task-statuses/{taskStatusId}	PUT	Modify task status
/api/v1/task-statuses/{taskStatusId}	PATCH	Modify partially a task status
/api/v1/task-statuses/{taskStatusId}	DELETE	Delete a task status
/api/v1/task-statuses/bulk_update_order	POST	Update task statuses order in bulk mode

2.18. Task custom attribute

URL	Method	Functionality
/api/v1/task-custom-attributes	GET	List task custom attributes
/api/v1/task-custom-attributes	POST	Create task custom attribute
/api/v1/task-custom-attributes/{taskCustomAttributeId}	GET	Get task custom attribute

URL	Method	Functionality
/api/v1/task-custom-attributes/{taskCustomAttributeId}	PUT	Modify task custom attribute
/api/v1/task-custom-attributes/{taskCustomAttributeId}	PATCH	Modify partially a task custom attribute
/api/v1/task-custom-attributes/{taskCustomAttributeId}	DELETE	Delete a task custom attribute
/api/v1/task-custom-attributes/bulk_update_order	POST	Update task custom attributes order in bulk mode

2.19. Task custom attributes values

URL	Method	Functionality
/api/v1/tasks/custom-attributes-values/{taskId}	GET	Get task custom attributes values
/api/v1/tasks/custom-attributes-values/{taskId}	PUT	Modify task custom attributes values
/api/v1/tasks/custom-attributes-values/{taskId}	PATCH	Modify partially a task custom attributes values

2.20. Issues

URL	Method	Functionality
/api/v1/issues	GET	List issues
/api/v1/issues	POST	Create issue
/api/v1/issues/{issueId}	GET	Get issue
/api/v1/issues/by_ref?ref={issueRef}&project={projectId}	GET	Get issue
/api/v1/issues/{issueId}	PUT	Modify issue
/api/v1/issues/{issueId}	PATCH	Modify partially an issue
/api/v1/issues/{issueId}	DELETE	Delete an issue
/api/v1/issues/bulk_create	POST	Create issues un bulk mode
/api/v1/issues/{issueId}/upvote	POST	Add a vote to an issue

URL	Method	Functionality
/api/v1/issues/{issueId}/downvote	POST	Remove your vote to an issue
/api/v1/issues/{issueId}/voters	GET	Get issue voters list
/api/v1/issues/{issueId}/watch	POST	Watch an issue
/api/v1/issues/{issueId}/unwatch	POST	Unwatch an issue
/api/v1/issues/{issueId}/watchers	GET	Get issue watchers
/api/v1/issues/attachments	GET	List issue attachments
/api/v1/issues/attachments	POST	Create issue attachments
/api/v1/issues/attachments/{issueAttachmentId}	GET	Get issue attachments
/api/v1/issues/attachments/{issueAttachmentId}	PUT	Modify issue attachments
/api/v1/issues/attachments/{issueAttachmentId}	PATCH	Modify partially an issue attachments
/api/v1/issues/attachments/{issueAttachmentId}	DELETE	Delete an issue attachments

2.21. Issue status

URL	Method	Functionality
/api/v1/issue-statuses	GET	List issue statuses
/api/v1/issue-statuses	POST	Create issue status
/api/v1/issue-statuses/{issueStatusId}	GET	Get issue status
/api/v1/issue-statuses/{issueStatusId}	PUT	Modify issue status
/api/v1/issue-statuses/{issueStatusId}	PATCH	Modify partially a issue status
/api/v1/issue-statuses/{issueStatusId}	DELETE	Delete a issue status
/api/v1/issue-statuses/bulk_update_order	POST	Update issue statuses order in bulk mode

2.22. Issue types

URL	Method	Functionality
/api/v1/issue-types	GET	List issue types
/api/v1/issue-types	POST	Create issue type
/api/v1/issue-types/{issueTypeId}	GET	Get issue type
/api/v1/issue-types/{issueTypeId}	PUT	Modify issue type
/api/v1/issue-types/{issueTypeId}	PATCH	Modify partially a issue type
/api/v1/issue-types/{issueTypeId}	DELETE	Delete a issue type
/api/v1/issue-types/bulk_update_order	POST	Update issue types order in bulk mode

2.23. Priorities

URL	Method	Functionality
/api/v1/priorities	GET	List priorities
/api/v1/priorities	POST	Create priority
/api/v1/priorities/{priorityId}	GET	Get priority
/api/v1/priorities/{priorityId}	PUT	Modify priority
/api/v1/priorities/{priorityId}	PATCH	Modify partially a priority
/api/v1/priorities/{priorityId}	DELETE	Delete a priority
/api/v1/priorities/bulk_update_order	POST	Update priorities order in bulk mode

2.24. Severities

URL	Method	Functionality
/api/v1/severities	GET	List severities
/api/v1/severities	POST	Create severity
/api/v1/severities/{severityId}	GET	Get severity
/api/v1/severities/{severityId}	PUT	Modify severity
/api/v1/severities/{severityId}	PATCH	Modify partially a severity
/api/v1/severities/{severityId}	DELETE	Delete a severity

URL	Method	Functionality
/api/v1/severities/bulk_update_order	POST	Update severities order in bulk mode

2.25. Issue custom attribute

URL	Method	Functionality
/api/v1/issue-custom-attributes	GET	List issue custom attributes
/api/v1/issue-custom-attributes	POST	Create issue custom attribute
/api/v1/issue-custom-attributes/{issueCustomAttributeId}	GET	Get issue custom attribute
/api/v1/issue-custom-attributes/{issueCustomAttributeId}	PUT	Modify issue custom attribute
/api/v1/issue-custom-attributes/{issueCustomAttributeId}	PATCH	Modify partially a issue custom attribute
/api/v1/issue-custom-attributes/{issueCustomAttributeId}	DELETE	Delete a issue custom attribute
/api/v1/issue-custom-attributes/bulk_update_order	POST	Update issue custom attributes order in bulk mode

2.26. Issue custom attributes values

URL	Method	Functionality
/api/v1/issues/custom-attributes-values/{issueId}	GET	Get issue custom attributes values
/api/v1/issues/custom-attributes-values/{issueId}	PUT	Modify issue custom attributes values
/api/v1/issues/custom-attributes-values/{issueId}	PATCH	Modify partially a issue custom attributes values

2.27. Wiki pages

URL	Method	Functionality
/api/v1/wiki	GET	List wiki pages
/api/v1/wiki	POST	Create wiki page
/api/v1/wiki/{wikiId}	GET	Get wiki page
/api/v1/wiki/by_slug?slug={wikiPageSlug}&project={projectId}	GET	Get wiki page
/api/v1/wiki/{wikiPageId}	PUT	Modify wiki page
/api/v1/wiki/{wikiPageId}	PATCH	Modify partially an wiki page
/api/v1/wiki/{wikiPageId}	DELETE	Delete an wiki page
/api/v1/wiki/{wikiPageId}/watch	POST	Watch a wiki page
/api/v1/wiki/{wikiPageId}/unwatch	POST	Stop watching a wiki page
/api/v1/wiki/{wikiPageId}/watchers	GET	Get wiki page watchers
/api/v1/wiki/attachments	GET	List wiki page attachments
/api/v1/wiki/attachments	POST	Create wiki page attachments
/api/v1/wiki/attachments/{wikiPageAttachmentId}	GET	Get wiki page attachments
/api/v1/wiki/attachments/{wikiPageAttachmentId}	PUT	Modify wiki page attachments
/api/v1/wiki/attachments/{wikiPageAttachmentId}	PATCH	Modify partially an wiki page attachments
/api/v1/wiki/attachments/{wikiPageAttachmentId}	DELETE	Delete an wiki page attachments

2.28. Wiki links

URL	Method	Functionality
/api/v1/wiki-links	GET	List wiki links
/api/v1/wiki-links	POST	Create wiki link
/api/v1/wiki-links/{wikiLinkId}	GET	Get wiki link
/api/v1/wiki-links/{wikiLinkId}	PUT	Modify wiki link
/api/v1/wiki-links/{wikiLinkId}	PATCH	Modify partially an wiki link

URL	Method	Functionality
/api/v1/wiki-links/{wikiLinkId}	DELETE	Delete an wiki link

2.29. History

URL	Method	Functionality
/api/v1/history/userstory/{usId}	GET	Get user story history
/api/v1/history/userstory/{usId}/delete_comment?id={commentId}	POST	Delete user story comment
/api/v1/history/userstory/{usId}/undelete_comment?id={commentId}	POST	Undelete user story comment
/api/v1/history/issue/{issueId}	GET	Get issue history
/api/v1/history/issue/{usId}/delete_comment?id={commentId}	POST	Delete issue comment
/api/v1/history/issue/{usId}/undelete_comment?id={commentId}	POST	Undelete issue comment
/api/v1/history/task/<taskId>	GET	Get task history
/api/v1/history/task/{usId}/delete_comment?id={commentId}	POST	Delete task comment
/api/v1/history/task/{usId}/undelete_comment?id={commentId}	POST	Undelete task comment
/api/v1/history/wiki/{wikiId}	GET	Get wiki history
/api/v1/history/wiki/{usId}/delete_comment?id={commentId}	POST	Delete wiki comment
/api/v1/history/wiki/{usId}/undelete_comment?id={commentId}	POST	Undelete wiki comment

2.30. Users

URL	Method	Functionality
/api/v1/users	GET	List users
/api/v1/users/{userId}	GET	Get user
/api/v1/users/me	GET	Get myself

URL	Method	Functionality
/api/v1/users/{userId}	PUT	Modify user
/api/v1/users/{userId}	PATCH	Modify partially a user
/api/v1/users/{userId}/stats	GET	Get user stats
/api/v1/users/{userId}/watched	GET	Get user watched content
/api/v1/users/{userId}/liked	GET	Get user liked content
/api/v1/users/{userId}/voted	GET	Get user voted content
/api/v1/users/{userId}	DELETE	Delete a user
/api/v1/users/{userId}/contacts	GET	Get user contacts
/api/v1/users/cancel	POST	Cancel user
/api/v1/users/change_avatar	POST	Change avatar
/api/v1/users/remove_avatar	POST	Remove avatar
/api/v1/users/change_email	POST	Change email
/api/v1/users/change_password	POST	Change password
/api/v1/users/password_recovery	POST	Password recovery
/api/v1/users/change_password_from_recovery	POST	Change password from recovery

2.31. Notify policies

URL	Method	Functionality
/api/v1/notify-policies	GET	List notify policies
/api/v1/notify-policies/{policyId}	GET	Get notify policy
/api/v1/notify-policies/{policyId}	PUT	Modify notify policy
/api/v1/notify-policies/{policyId}	PATCH	Modify partially a notify policy

2.32. Feedback

URL	Method	Functionality
/api/v1/feedback	POST	Send feedback

2.33. Export/Import

URL	Method	Functionality
/api/v1/exporter/{projectId}	GET	Export a project dump
/api/v1/importer/load_dump	POST	Import a project dump

2.34. Webhooks

URL	Method	Functionality
/api/v1/webhooks	GET	List webhooks
/api/v1/webhooks	POST	Create webhook
/api/v1/webhooks/{webhookId}	GET	Get webhook
/api/v1/webhooks/{webhookId}	PUT	Modify webhook
/api/v1/webhooks/{webhookId}	PATCH	Modify partially an webhook
/api/v1/webhooks/{webhookId}	DELETE	Delete an webhook
/api/v1/webhooks/{webhookId}/test	POST	Test webhook
/api/v1/webhooklogs	GET	List webhooks logs
/api/v1/webhooklogs/{webhookLogId}	GET	Get webhook log
/api/v1/webhooklogs/{webhookLogId}/resend	POST	Resend webhook log request

2.35. Timelines

URL	Method	Functionality
/api/v1/timeline/user/{userId}	GET	List user timeline
/api/v1/timeline/profile/{userId}	GET	List profile timeline
/api/v1/timeline/project/{projectId}	GET	List project timeline

2.36. Locales

URL	Method	Functionality
/api/v1/locales	GET	List locales

3. Auth

3.1. Normal login

To login a user send a POST request containing the following data:

- **type** with value "normal"
- **username** (required): this field also supports the user email
- **password** (required)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
        "type": "normal",
        "username": "'${USERNAME}'",
        "password": "'${PASSWORD}'"
    }' \
  https://api.taiga.io/api/v1/auth
```

When the login is successful, the HTTP response is a 200 OK and the response body is a JSON user auth detail object

3.2. Github login

To login a user via GitHub send a POST request containing the following data:

- **type** with value "github"
- **code** (required): your github authentication code
- **token**: generated when creating a project's membership (for accept invitations to projects)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
        "type": "github",
        "code": "'${GITHUB_CODE}'"
    }' \
  https://api.taiga.io/api/v1/auth
```

When the login is successful, the HTTP response is a 200 OK and the response body is a JSON user auth detail object

NOTE

Get GitHub autorized code

To get the GitHub code you have to follow the first step *Redirect users to request GitHub access* described in GitHub Documentation for Developers - API - OAuth - Web Application Flow.

Taiga needs privileges to get the user email from Github so you have to use the scope user:email.

3.3. Public registry

To register a user without invitation send a POST request containing the following data:

- **type** with value "public"
- **username** (required)
- **password** (required)
- **email** (required)
- **full_name** (required)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
        "type": "public",
        "username": "'${USERNAME}'",
        "password": "'${PASSWORD}'",
        "email": "'${EMAIL}'",
        "full_name": "'${FULL_NAME}'"
    }' \
  https://api.taiga.io/api/v1/auth/register
```

When the registration is successful, the HTTP response is a 201 CREATED and the response body is a JSON user auth detail object

3.4. Private registry

To add a user into a project via invitation send a POST request containing the following data:

- **type** with value "private"
- **existing** (required): indicates if the user is member or not

token (required): generated when creating a project's membership

- **username** (required)
- **password** (required)
- **email** (required only if the user doesn't exist in the platform)
- **full_name** (required only if the user doesn't exist in the platform)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{
        "type": "private",
        "existing": false,
        "token": "e8ea658e-6655-11e4-9d95-b499ba562790",
        "username": "'${USERNAME}'",
        "password": "'${PASSWORD}'",
        "email": "'${EMAIL}'",
        "full_name": "'${FULL_NAME}'"
    }' \
  https://api.taiga.io/api/v1/auth/register
```

When the registration is successful, the HTTP response is a 201 CREATED and the response body is a JSON user auth detail object

4. Applications

4.1. Get

To get an application send a GET request specifying the application id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/applications/a60c3208-5234-11e5-96df-68f72800aadd
```

The HTTP response is a 200 OK and the response body is a JSON application object

4.2. Get token

To get an application token send a GET request specifying the application id in the url

-

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/applications/a60c3208-5234-11e5-96df-68f72800aadd/token
```

The HTTP response is a 200 OK and the response body is a JSON application token object

5. Application tokens

5.1. List

To list the application tokens for an authenticated user send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/application-tokens
```

The HTTP response is a 200 OK and the response body is a JSON list of application token objects

5.2. Get

To get an application token send a GET request specifying the application token id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/application-tokens/1
```

The HTTP response is a 200 OK and the response body is a JSON application token object

5.3. Delete

To delete application tokens send a DELETE specifying the application token id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/application-tokens/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

5.4. Authorize

To request an authorization code send a POST request with the following data:

- **application**: the application id for the requested token
- **state**: an unguessable random string. It is used to protect against cross-site request forgery attacks. The API will include this value when the validation process is completed so the final app can verify it matches the original one.

```
curl -X POST \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer
eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjo0fQ:1ZX33b:QnAN3EcuChLoRVf3CdybWEi2OEg" \
    -d '{
          "application": "a60c3208-5234-11e5-96df-68f72800aadd",
          "state": "random-state"
      }' \
    https://api.taiga.io/api/v1/application-tokens/authorize
```

When the creation is successful, the HTTP response is a 200 and the response body is a JSON authorization code object

5.5. Validate

To validate an authorization code send a POST request with the following data:

- **application**: the application id for the requested token
- **state**: an unguessable random string. It is used to protect against cross-site request forgery attacks. The API will include this value when the validation process is completed so the final app can verify it matches the original one.
- **auth_code**: the authorization code received on previous the steps.

```
curl -X POST \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer
eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjo0fQ:1ZX33b:QnAN3EcuChLoRVf3CdybWEi2OEg" \
    -d '{
          "application": "a60c3208-5234-11e5-96df-68f72800aadd",
          "auth_code": "21ce08c4-5237-11e5-a8a3-68f72800aadd",
          "state": "random-state"
      }' \
https://api.taiga.io/api/v1/application-tokens/validate
```

When the creation is successful, the HTTP response is a 200 and the response body is a JSON cyphered token object

6. Resolver

6.1. Projects

To resolve the id of a project send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga
```

The response body is a JSON object containing the project id

```
{
    "project": 1
}
```

6.2. User stories

To resolve the id of a user story send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved
- **us** (required): the user story ref trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&us=1
```

The response body is a JSON object containing the project and the user story ids

```
{
    "us": 26,
    "project": 1
}
```

6.3. Issues

To resolve the id of an issue send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved
- **issue** (required): the issue ref trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&issue=1485
```

The response body is a JSON object containing the project and the issue ids

```
{
    "issue": 5209,
    "project": 1
}
```

6.4. Tasks

To resolve the id of a task send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved
- **task** (required): the task ref trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&task=915
```

The response body is a JSON object containing the project and the task ids

```
{
    "task": 1336,
    "project": 1
}
```

6.5. Milestones

To resolve the id of a milestone send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved
- **milestone** (required): the milestone slug trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&milestone=sprint-0
```

The response body is a JSON object containing the project and the milestone ids

```
{
    "milestone": 1,
    "project": 1
}
```

6.6. Wiki pages

To resolve the id of a wiki page send a GET request with the following parameters:

- **project** (required): the project slug trying to be resolved
- **wikipage**(required): the wiki-page slug trying to be resolved

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&wikipage=home
```

The response body is a JSON object containing the project and the wiki page ids

```
{
    "wikipage": 2,
    "project": 1
}
```

6.7. Multiple resolution

To resolve the multiple ids you can send a GET mixing parameters from the previous examples:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&task=915\&us=1\&wikipage=home
```

The response body is a JSON object containing the project and the task ids

```
{
    "project": 1,
    "task": 1336,
    "us": 26,
    "wikipage": 2
}
```

6.8. By ref value

To resolve an object if we don't know its type we have to use `ref` GET parameter:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/resolver?project=taiga\&ref=915
```

The response body is a JSON object containing the project and the story, task or issue id.

```
{
    "project": 1,
    "task": 1336,
}
```

IMPORTANT	*Incompatibility between GET params* Be careful because `ref` is inconpatible with `us`, `task` and `issue` parameters in requests with multiple resolutions.

7. Searches

7.1. Search

To search send a GET request with the following get parameters:

- **project** (required): project id
- **text**: string

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/search?project=1\&text=design
```

The HTTP response is a 200 OK and the response body is a JSON list of search results detail objects

8. User storage

8.1. List

To list user storage data send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/user-storage
```

The HTTP response is a 200 OK and the response body is a JSON list of user storage data objects

8.2. Create

To create user storage data send a POST request with the following data:

- **key**: string
- **value**: string

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "key": "favorite-forest",
        "value": "Taiga"
    }' \
  https://api.taiga.io/api/v1/user-storage
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON user storage data object

8.3. Get

To get a user storage data send a GET request specifying the user storage key in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/user-storage/favorite-forest
```

The HTTP response is a 200 OK and the response body is a JSON user storage data object

8.4. Edit

To edit user storage data send a PUT or a PATCH specifying the user storage key in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "value": "Russian Taiga"
      }' \
  https://api.taiga.io/api/v1/user-storage/favorite-forest
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON user storage data object

8.5. Delete

To delete user storage data send a DELETE specifying the user storage key in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/user-storage/favorite-forest
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

9. Project templates

9.1. List

To list project templates send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/project-templates
```

The HTTP response is a 200 OK and the response body is a JSON list of project template detail objects

9.2. Create

To create project templates send a POST request with the following data:

- **name** (required): string
- **slug**: slug

- **description** (required): string
- **default_owner_role** (required):
- **is_backlog_activated**: boolean
- **is_kanban_activated**: boolean
- **is_wiki_activated**: boolean
- **is_issues_activated**: boolean
- **videoconferences**: (talky|appear-in)
- **videoconferences_extra_data**: string
- **default_options**: a json with a list of objects with:
  - **points**: slug
  - **us_status**: slug
  - **task_status**: slug
  - **issue_status**: slug
  - **issue_type**: slug
  - **priority**: slug
  - **severity**: slug
- **us_statuses**: a json with a list of objects with:
  - **name**: string
  - **slug**: slug
  - **is_closed**: boolean
  - **color**: #rgb color
  - **wip_limit**: integer or none
  - **order**: integer
- **points**: a json with a list of objects with:
  - **name**: string
  - **value**: integer or none
  - **order**: integer
- **task_statuses**: a json with a list of objects with:
  - **name**: string
  - **slug**: slug
  - **is_closed**: boolean
  - **color**: #rgb color

    - **order**: integer
- **issue_statuses**: a json with a list of objects with:
    - **name**: string
    - **slug**: slug
    - **is_closed**: boolean
    - **color**: #rgb color
    - **order**: integer
- **issue_types**: a json with a list of objects with:
    - **name**: string
    - **color**: #rgb color
    - **order**: integer
- **priorities**: a json with a list of objects with:
    - **name**: string
    - **color**: #rgb color
    - **order**: integer
- **severities**: a json with a list of objects with:
    - **name**: string
    - **color**: #rgb color
    - **order**: integer
- **roles**: a json with a list of objects with:
    - **name**: string
    - **slug**: slug
    - **permissions**: list of permissions
    - **order**: integer
    - **computable**: boolean

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "default_options": {
            "us_status": "New",
            "points": "?",
            "priority": "Normal",
            "severity": "Normal",
            "task_status": "New",
```

```
            "issue_type": "Bug",
            "issue_status": "New"
        },
        "us_statuses": [
            {
                "wip_limit": null,
                "color": "#999999",
                "name": "New",
                "order": 1,
                "is_closed": false
            },
            {
                "wip_limit": null,
                "color": "#f57900",
                "name": "Ready",
                "order": 2,
                "is_closed": false
            },
            {
                "wip_limit": null,
                "color": "#729fcf",
                "name": "In progress",
                "order": 3,
                "is_closed": false
            },
            {
                "wip_limit": null,
                "color": "#4e9a06",
                "name": "Ready for test",
                "order": 4,
                "is_closed": false
            },
            {
                "wip_limit": null,
                "color": "#cc0000",
                "name": "Done",
                "order": 5,
                "is_closed": true
            }
        ],
        "points": [
            {
                "value": null,
                "name": "?",
                "order": 1
            },
            {
                "value": 0.0,
```

```
                "name": "0",
                "order": 2
            },
            {
                "value": 0.5,
                "name": "1/2",
                "order": 3
            },
            {
                "value": 1.0,
                "name": "1",
                "order": 4
            },
            {
                "value": 2.0,
                "name": "2",
                "order": 5
            },
            {
                "value": 3.0,
                "name": "3",
                "order": 6
            },
            {
                "value": 5.0,
                "name": "5",
                "order": 7
            },
            {
                "value": 8.0,
                "name": "8",
                "order": 8
            },
            {
                "value": 10.0,
                "name": "10",
                "order": 9
            },
            {
                "value": 15.0,
                "name": "15",
                "order": 10
            },
            {
                "value": 20.0,
                "name": "20",
                "order": 11
            },
```

```
            {
                "value": 40.0,
                "name": "40",
                "order": 12
            }
        ],
        "task_statuses": [
            {
                "color": "#999999",
                "name": "New",
                "order": 1,
                "is_closed": false
            },
            {
                "color": "#729fcf",
                "name": "In progress",
                "order": 2,
                "is_closed": false
            },
            {
                "color": "#f57900",
                "name": "Ready for test",
                "order": 3,
                "is_closed": true
            },
            {
                "color": "#4e9a06",
                "name": "Closed",
                "order": 4,
                "is_closed": true
            },
            {
                "color": "#cc0000",
                "name": "Needs Info",
                "order": 5,
                "is_closed": false
            }
        ],
        "issue_statuses": [
            {
                "color": "#999999",
                "name": "New",
                "order": 1,
                "is_closed": false
            },
            {
                "color": "#729fcf",
                "name": "In progress",
```

```
                "order": 2,
                "is_closed": false
            },
            {
                "color": "#f57900",
                "name": "Ready for test",
                "order": 3,
                "is_closed": true
            },
            {
                "color": "#4e9a06",
                "name": "Closed",
                "order": 4,
                "is_closed": true
            },
            {
                "color": "#cc0000",
                "name": "Needs Info",
                "order": 5,
                "is_closed": false
            },
            {
                "color": "#d3d7cf",
                "name": "Rejected",
                "order": 6,
                "is_closed": true
            },
            {
                "color": "#75507b",
                "name": "Postponed",
                "order": 7,
                "is_closed": false
            }
        ],
        "issue_types": [
            {
                "color": "#cc0000",
                "name": "Bug",
                "order": 1
            },
            {
                "color": "#729fcf",
                "name": "Question",
                "order": 2
            },
            {
                "color": "#4e9a06",
                "name": "Enhancement",
```

```
                "order": 3
            }
        ],
        "priorities": [
            {
                "color": "#999999",
                "name": "Low",
                "order": 1
            },
            {
                "color": "#4e9a06",
                "name": "Normal",
                "order": 3
            },
            {
                "color": "#CC0000",
                "name": "High",
                "order": 5
            }
        ],
        "severities": [
            {
                "color": "#999999",
                "name": "Wishlist",
                "order": 1
            },
            {
                "color": "#729fcf",
                "name": "Minor",
                "order": 2
            },
            {
                "color": "#4e9a06",
                "name": "Normal",
                "order": 3
            },
            {
                "color": "#f57900",
                "name": "Important",
                "order": 4
            },
            {
                "color": "#CC0000",
                "name": "Critical",
                "order": 5
            }
        ],
        "roles": [
```

```
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "add_milestone", "modify_milestone",
                    "delete_milestone", "view_milestones", "view_project",
                    "add_task", "modify_task", "delete_task", "view_tasks",
                    "add_us", "modify_us", "delete_us", "view_us",
                    "add_wiki_page", "modify_wiki_page", "delete_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 10,
                "computable": true,
                "slug": "ux",
                "name": "UX"
            },
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "add_milestone", "modify_milestone",
                    "delete_milestone", "view_milestones", "view_project",
                    "add_task", "modify_task", "delete_task", "view_tasks",
                    "add_us", "modify_us", "delete_us", "view_us",
                    "add_wiki_page", "modify_wiki_page", "delete_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 20,
                "computable": true,
                "slug": "design",
                "name": "Design"
            },
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "add_milestone", "modify_milestone",
                    "delete_milestone", "view_milestones", "view_project",
                    "add_task", "modify_task", "delete_task", "view_tasks",
                    "add_us", "modify_us", "delete_us", "view_us",
                    "add_wiki_page", "modify_wiki_page", "delete_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 30,
                "computable": true,
                "slug": "front",
                "name": "Front"
            },
```

```
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "add_milestone", "modify_milestone",
                    "delete_milestone", "view_milestones", "view_project",
                    "add_task", "modify_task", "delete_task", "view_tasks",
                    "add_us", "modify_us", "delete_us", "view_us",
                    "add_wiki_page", "modify_wiki_page", "delete_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 40,
                "computable": true,
                "slug": "back",
                "name": "Back"
            },
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "add_milestone", "modify_milestone",
                    "delete_milestone", "view_milestones", "view_project",
                    "add_task", "modify_task", "delete_task", "view_tasks",
                    "add_us", "modify_us", "delete_us", "view_us",
                    "add_wiki_page", "modify_wiki_page", "delete_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 50,
                "computable": false,
                "slug": "product-owner",
                "name": "Product Owner"
            },
            {
                "permissions": [
                    "add_issue", "modify_issue", "delete_issue",
                    "view_issues", "view_milestones", "view_project",
                    "view_tasks", "view_us", "modify_wiki_page",
                    "view_wiki_pages", "add_wiki_link", "delete_wiki_link",
                    "view_wiki_links"
                ],
                "order": 60,
                "computable": false,
                "slug": "stakeholder",
                "name": "Stakeholder"
            }
        ],
        "id": 2,
        "name": "Kanban",
```

```
          "slug": "kanban",
          "description": "Sample description",
          "default_owner_role": "product-owner",
          "is_backlog_activated": false,
          "is_kanban_activated": true,
          "is_wiki_activated": false,
          "is_issues_activated": false,
          "videoconferences": null,
          "videoconferences_extra_data": ""
      }' \
  https://api.taiga.io/api/v1/project-templates
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -D '{
          "name": "Kanban",
          "description": "Sample description",
          "default_owner_role": "product-owner"
      }' \
  https://api.taiga.io/api/v1/project-templates
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON project template detail object

9.3. Get

To get a project template send a GET request specifying the project template id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/project-templates/1
```

The HTTP response is a 200 OK and the response body is a JSON project template detail object

9.4. Edit

To edit project templates send a PUT or a PATCH specifying the project template id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "description": "New description"
      }' \
  https://api.taiga.io/api/v1/project-templates/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON project template detail object

9.5. Delete

To delete project template send a DELETE specifying the project template id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/project-templates/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

10. Projects

10.1. List

To list projects send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects
```

The HTTP response is a 200 OK and the response body is a JSON list of project list entry objects

The results can be filtered using the following parameters:

- **member**: member id
- **members**: member ids

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects?member=1
```

The results can be ordered using the order_by parameter with the values:

- **memberships__user_order**: the project order specified by the user

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects?member=1&order_by=memberships__user_order
```

10.2. Create

To create projects send a POST request with the following data:

- **name** (required)
- **description** (required)
- **creation_template**: base template for the project
- **is_backlog_activated**
- **is_issues_activated**
- **is_kanban_activated**
- **is_private**
- **is_wiki_activated**
- **videoconferences**: appear-in or talky, the third party used for meetups if enabled
- **videoconferences_extra_data**: string used for the videoconference chat url generation
- **total_milestones**
- **total_story_points**

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Beta project",
          "description": "Beta description"
      }' \
  https://api.taiga.io/api/v1/projects
```

```
curl -v -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Beta project",
          "description": "Taiga",
          "creation_template": 1,
          "is_backlog_activated": false,
          "is_issues_activated": true,
          "is_kanban_activated": true,
          "is_private": false,
          "is_wiki_activated": true,
          "videoconferences": "appear-in",
          "videoconferences_extra_data": null,
          "total_milestones": 3,
          "total_story_points": 20.0
      }' \
  https://api.taiga.io/api/v1/projects
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON project detail object

10.3. Get

To get a project send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1
```

The HTTP response is a 200 OK and the response body is a JSON project detail object

10.4. Get by slug

To get a project send a GET request specifying the project slug as parameter:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/by_slug?slug=test
```

The HTTP response is a 200 OK and the response body is a JSON project detail object

10.5. Edit

To edit projects send a PUT or a PATCH specifying the project id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PUT \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Beta project put",
          "description": "Beta description"
      }' \
  https://api.taiga.io/api/v1/projects/1
```

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Beta project patch"
      }' \
  https://api.taiga.io/api/v1/projects/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON project detail object

10.6. Delete

To delete projects send a DELETE specifying the project id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

10.7. Bulk update order

To update the projects order for the logged in user send a POST request with a json list where each element is a json object with two attributes, the project id and the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '[
          {
              "project_id": 123,
              "order": 2
          },
          {
              "project_id": 456,
              "order": 2
          }
      ]' \
  https://api.taiga.io/api/v1/projects/bulk_update_order
```

When the update is successful, the HTTP response is a 200 OK and the response body is empty

10.8. Get modules configuration

To get a project modules configuration send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/modules
```

The HTTP response is a 200 OK and the response body is a JSON project modules configuration object

10.9. Edit modules configuration

To edit a project modules configuration send a PATCH specifying the project id in the url.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "github": {
              "secret": "new_secret"
          }
      }' \
  https://api.taiga.io/api/v1/projects/1/modules
```

When edition succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

10.10. Stats

To get a project stats send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/stats
```

The HTTP response is a 200 OK and the response body is a JSON project stats object

10.11. Issue stats

To get a project issue stats send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/issues_stats
```

The HTTP response is a 200 OK and the response body is a JSON project issue stats object

10.12. Issue filters data

To get a project issue filters data send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/issue_filters_data
```

The HTTP response is a 200 OK and the response body is a JSON project issue filters data object

10.13. Tag colors

To get a project tag colors stats send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/tags_colors
```

The HTTP response is a 200 OK and the response body is a JSON project tag colors object

10.14. Like a project

To like a project send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/like
```

The HTTP response is a 200 OK with an empty body response

10.15. Unlike a project

To unlike a project send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/unlike
```

The HTTP response is a 200 OK with an empty body response

10.16. List project fans

To get the list of fans from a project send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/fans
```

The HTTP response is a 200 OK and the response body is a JSON list of project voter object

10.17. Watch a project

To watch a project send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "notify_level": 3
      }' \
  https://api.taiga.io/api/v1/projects/1/watch
```

The HTTP response is a 200 OK with an empty body response

10.18. Stop watching project

To stop watching a project send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

10.19. List project watchers

To get the list of watchers from a project send a GET request specifying the project id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of project watcher object

10.20. Create template

To create a template from a selected project send a POST request specifying the project id in the url with the following parameters:

- **name** (required)
- **description** (required)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "template_name": "Beta template",
        "template_description": "Beta template description"
    }' \
  https://api.taiga.io/api/v1/projects/1/create_template
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON project template detail object

10.21. Leave

To leave a project send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/projects/1/leave
```

The HTTP response is a 200 OK with an empty body response

11. Memberships/Invitations

11.1. List

To list memberships send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/memberships
```

The HTTP response is a 200 OK and the response body is a JSON list of membership detail objects

The results can be filtered using the following parameters:

- **project**: project id
- **role**: role id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/memberships?project=1
```

11.2. Create

To create memberships/invitations send a POST request with the following data:

- **project** (required)
- **role** (required): Role to the membership
- **email** (required): user email

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "role": 12,
        "email": "test@test.com"
    }' \
  https://api.taiga.io/api/v1/memberships
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON membership detail object

11.3. Bulk creation

To create multiple memberships at the same time send a POST request with the following data:

- **project_id** (required)
- **bulk_memberships** (required): a list of dicts with
  - **role_id**
  - **email**
- **invitation_extra_text**: string

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_memberships": [
              {"role_id": 10, "email": "test@test.com"},
              {"role_id": 12, "email": "john@doe.com"}
          ]
      }' \
  https://api.taiga.io/api/v1/memberships/bulk_create
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON list of membership detail object

11.4. Get

To get a membership send a GET request specifying the membership id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/memberships/1
```

The HTTP response is a 200 OK and the response body is a JSON membership detail object

11.5. Edit

To edit memberships send a PUT or a PATCH specifying the membership id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "role": 10
      }' \
  https://api.taiga.io/api/v1/memberships/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON membership detail object

11.6. Delete

To delete memberships/invitations send a DELETE specifying the membership id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/memberships/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

11.7. Resend invitation

To resend an invitation send a POST request specifying the membership id in the url

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/memberships/1/resend_invitation
```

The HTTP response is a 200 OK and the response body is a JSON membership detail object

11.8. Get Invitation (by token)

To get an invitation send a GET request specifying the invitation id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/invitations/8e3af6bcd4
```

The HTTP response is a 200 OK and the response body is a JSON membership detail object

12. Milestones

12.1. List

To list milestones send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones
```

The HTTP response is a 200 OK and the response body is a JSON list of milestone detail objects

The results can be filtered using the following parameters:

- **project**: project ID
- **closed**: `true` to get only closed milestones or `false` to get only opened ones.

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones?project=1
```

NOTE

When you filter milestones by project ID (`/api/v1/milestones?project=<projectID>`) the response has two new headers:

- `Taiga-Info-Total-Opened-Milestones`: the numer of opened milestones for this project.
- `Taiga-Info-Total-Closed-Milestones`: the numer of closed milestones for this project.

12.2. Create

To create milestone send a POST request with the following data:

- **project** (required): project id
- **name** (required): string
- **estimated_start** (required): iso date (YYYY-MM-DD)
- **estimated_finish** (required): iso date (YYYY-MM-DD)

- **disponibility**: float
- **slug**: slug
- **order**: integer
- **watchers**: array of watcher id's

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "name": "Sprint 1",
        "estimated_start": "2014-10-20",
        "estimated_finish": "2014-11-04",
        "disponibility": 30,
        "slug": "sprint-1",
        "order": 1,
        "watchers": []
    }' \
  https://api.taiga.io/api/v1/milestones
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "name": "Sprint 3",
        "estimated_start": "2014-10-20",
        "estimated_finish": "2014-11-04"
    }' \
  https://api.taiga.io/api/v1/milestones
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON milestone detail object

12.3. Get

To get a milestone send a GET request specifying the milestone id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1
```

The HTTP response is a 200 OK and the response body is a JSON milestone detail object

12.4. Edit

To edit milestones send a PUT or a PATCH specifying the milestone id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Sprint 2"
      }' \
  https://api.taiga.io/api/v1/milestones/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON milestone detail object

12.5. Delete

To delete milestones send a DELETE specifying the milestone id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

12.6. Stats

To get the milestone stats send a GET request specifying the milestone id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1/stats
```

The HTTP response is a 200 OK and the response body is a JSON milestone stats detail object

12.7. Watch a milestone

To watch a milestone send a POST request specifying the milestone id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1/watch
```

The HTTP response is a 200 OK with an empty body response

12.8. Stop watching a milestone

To stop watching an milestone send a POST request specifying the milestone id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

12.9. List milestone watchers

To get the list of watchers from a milestone send a GET request specifying the milestone id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/milestones/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of milestone watcher object

13. User stories

13.1. List

To list user stories send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories
```

The HTTP response is a 200 OK and the response body is a JSON list of user story list objects

The results can be filtered using the following parameters:

- **project**: project id
- **milestone**: milestone id
- **milestone__isnull**: (true|false) if you are looking for user stories associated with a milestone or not
- **status**: status id
- **status__is_archived**: (true|false)
- **watchers**: watching user id
- **assigned_to**: assigned to user id
- **status__is_closed**: (true|false)

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories?project=1
```

13.2. Create

To create user stories send a POST request with the following data:

- **assigned_to**: user id
- **backlog_order**: order in the backlog
- **blocked_note**: reason why the user story is blocked
- **client_requirement**: boolean
- **description**: string
- **is_blocked**: boolean
- **is_closed**: boolean
- **kanban_order**: order in the kanban
- **milestone**: milestone id
- **points**: dictionary of points

- **project** (required): project id
- **sprint_order**: order in the milestone
- **status**: status id
- **subject** (required)
- **tags**: array of strings
- **team_requirement**: boolean
- **watchers**: array of watcher id's

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "assigned_to": null,
          "backlog_order": 2,
          "blocked_note": "blocking reason",
          "client_requirement": false,
          "description": "Implement API CALL",
          "is_blocked": false,
          "is_closed": true,
          "kanban_order": 37,
          "milestone": null,
          "points": {
              "129": 361,
              "130": 361,
              "131": 361,
              "132": 364
          },
          "project": 3,
          "sprint_order": 2,
          "status": 13,
          "subject": "Customer personal data",
          "tags": [
              "service catalog",
              "customer"
          ],
          "team_requirement": false,
          "watchers": []
      }' \
  https://api.taiga.io/api/v1/userstories
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "subject": "Customer personal data"
      }' \
  https://api.taiga.io/api/v1/userstories
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON user story detail object

13.3. Get

To get a user story send a GET request specifying the user story id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1
```

The HTTP response is a 200 OK and the response body is a JSON user story detail (GET) object

13.4. Get by ref

To get a user story send a GET request specifying the user story reference and the project id as parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/by_ref?ref=1&project=1
```

The HTTP response is a 200 OK and the response body is a JSON user story detail (GET) object

13.5. Edit

To edit user stories send a PUT or a PATCH specifying the user story id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "subject": "Patching subject"
      }' \
  https://api.taiga.io/api/v1/userstories/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON user story detail object

13.6. Delete

To delete user stories send a DELETE specifying the user story id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

13.7. Bulk creation

To create multiple user stories at the same time send a POST request with the following data:

- **project_id** (required)
- **status_id**
- **bulk_stories**: user story subjects, one per line

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_stories": "US 1 \n US 2 \n US 3"
      }' \
  https://api.taiga.io/api/v1/userstories/bulk_create
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON list of user story detail object

13.8. Bulk update backlog order

To update the backlog order of multiple user stories at the same time send a POST request with the following data:

- **project_id** (required)
- **bulk_stories**: list where each element is a json object with two attributes, the user story id and the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_stories": [
              {
                  "us_id": 123,
                  "order": 2
              },
              {
                  "us_id": 456,
                  "order": 2
              }
          ]
      }' \
  https://api.taiga.io/api/v1/userstories/bulk_update_backlog_order
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON list of user story detail object

13.9. Bulk update kanban order

To update the kanban order of multiple user stories at the same time send a POST request with the following data:

- **project_id** (required)
- **bulk_stories**: list where each element is a json object with two attributes, the user story id and the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_stories": [
              {
                  "us_id": 123,
                  "order": 2
              },
              {
                  "us_id": 456,
                  "order": 2
              }
          ]
      }' \
  https://api.taiga.io/api/v1/userstories/bulk_update_kanban_order
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON list of user story detail object

13.10. Bulk update sprint order

To update the sprint order of multiple user stories at the same time send a POST request with the following data:

- **project_id** (required)
- **bulk_stories**: list where each element is a json object with two attributes, the user story id and the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_stories": [
              {
                  "us_id": 123,
                  "order": 2
              },
              {
                  "us_id": 456,
                  "order": 2
              }
          ]
      }' \
  https://api.taiga.io/api/v1/userstories/bulk_update_sprint_order
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON list of user story detail object

13.11. Vote a user story

To add a vote to a user story send a POST request specifying the user story id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/upvote
```

The HTTP response is a 200 OK with an empty body response

13.12. Remove vote from a user story

To remove a vote from a user story send a POST request specifying the user story id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/downvote
```

When remove of the vote succeeded, the HTTP response is a 200 OK with an empty body response

13.13. Get user story voters list

To get the list of voters from a user story send a GET request specifying the user story id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/voters
```

The HTTP response is a 200 OK and the response body is a JSON list of user story voter object

13.14. Watch a user story

To watch a user story send a POST request specifying the project id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/watch
```

The HTTP response is a 200 OK with an empty body response

13.15. Stop watching a user story

To stop watching a user story send a POST request specifying the user story id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

13.16. List user story watchers

To get the list of watchers from a user story send a GET request specifying the user story id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of user story watcher object

13.17. List attachments

To list user story attachments send a GET request with the following parameters:

- **project**: project id
- **object_id**: user story id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/attachments?object_id=81\&project=3
```

The HTTP response is a 200 OK and the response body is a JSON list of attachment detail objects

13.18. Create attachment

To create user story attachments send a POST request with the following data:

- **object_id** (required): user story id
- **project** (required): project id
- **attached_file** (required): attaching file
- **description**
- **is_deprecated**

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "object_id=81" \
  -F "project=3" \
  -F "attached_file=@/tmp/test.png" \
  https://api.taiga.io/api/v1/userstories/attachments
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON attachment detail object

13.19. Get attachment

To get a user story attachment send a GET request specifying the user story attachment id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/attachments/415
```

The HTTP response is a 200 OK and the response body is a JSON attachment detail object

13.20. Edit attachment

To edit user story attachments send a PUT or a PATCH specifying the user story attachment id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "description=patching description" \
  https://api.taiga.io/api/v1/userstories/attachments/417
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON attachment detail object

13.21. Delete attachment

To delete user story attachments send a DELETE specifying the user story attachment id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/attachments/415
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

14. User story status

14.1. List

To list user story status send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-statuses
```

The HTTP response is a 200 OK and the response body is a JSON list of user story status detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-statuses?project=1
```

14.2. Create

To create user story statuses send a POST request with the following data:

- **color**: in hexadecimal
- **is_closed**: (true|false)
- **name** (required)
- **order**: integer
- **project**: (required): project id
- **wip_limit**: integer representing the max number of user stories in this status

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "is_closed": true,
        "name": "New status",
        "order": 8,
        "project": 3,
        "wip_limit": 6
    }' \
  https://api.taiga.io/api/v1/userstory-statuses
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "name": "New status name"
      }' \
  https://api.taiga.io/api/v1/userstory-statuses
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON user story status detail object

14.3. Get

To get a user story status send a GET request specifying the user story status id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-statuses/1
```

The HTTP response is a 200 OK and the response body is a JSON user story status detail object

14.4. Edit

To edit user story statuses send a PUT or a PATCH specifying the user story status id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Patch status name"
      }' \
  https://api.taiga.io/api/v1/userstory-statuses/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON user story status detail object

14.5. Delete

To delete user story satuses send a DELETE specifying the user story status id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-statuses/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

14.6. Bulk update order

To update the order of multiple user story statues at the same time send a POST request with the following data:

- **project** (required)
- **bulk_userstory_statuses**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project_id": 3,
        "bulk_userstory_statuses": [[1,10], [2,5]]
    }' \
  https://api.taiga.io/api/v1/userstory-statuses/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

15. Points

15.1. List

To list points send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/points
```

The HTTP response is a 200 OK and the response body is a JSON list of point detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/points?project=1
```

15.2. Create

To create points send a POST request with the following data:

- **color**: in hexadecimal
- **name** (required)
- **order**: integer
- **value** (required): integer
- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "name": "Huge",
        "order": 8,
        "value": 40,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/points
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "Very huge"
      }' \
  https://api.taiga.io/api/v1/points
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON point detail object

15.3. Get

To get a point send a GET request specifying the point id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/points/1
```

The HTTP response is a 200 OK and the response body is a JSON point detail object

15.4. Edit

To edit points send a PUT or a PATCH specifying the point id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Patch name"
      }' \
  https://api.taiga.io/api/v1/points/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON point detail object

15.5. Delete

To delete points send a DELETE specifying the point id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/points/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

15.6. Bulk update order

To update the order of multiple points at the same time send a POST request with the following data:

- **project** (required)

- **bulk_points**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project_id": 3,
        "bulk_points": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/points/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

16. User story custom attribute

16.1. List

To list user story custom attributes send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-custom-attributes
```

The HTTP response is a 200 OK and the response body is a JSON list of user story custom attribute detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-custom-attributes?project=1
```

16.2. Create

To create user story custom attributes send a POST request with the following data:

- **name**: (required) text

- **description**: text
- **order**: integer
- **project**: (required) integer, project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Duration",
        "description": "Duration in minutes",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/userstory-custom-attributes
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Duration",
          "project": 3
      }' \
  https://api.taiga.io/api/v1/userstory-custom-attributes
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON user story custom attribute detail object

16.3. Get

To get a user story custom attribute send a GET request specifying the user story custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-custom-attributes/1
```

The HTTP response is a 200 OK and the response body is a JSON user story custom attribute detail object

16.4. Edit

To edit user story custom attributes send a PUT or a PATCH specifying the user story custom attribute id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
         "name": "Duration"
      }' \
  https://api.taiga.io/api/v1/userstory-custom-attributes/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON user story custom attribute detail object

16.5. Delete

To delete user story custom attributes send a DELETE specifying the user story custom attribute id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstory-custom-attributes/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

16.6. Bulk update order

To update the order of multiple user story custom attributes at the same time send a POST request with the following data:

- **project** (required)
- **bulk_userstory_custom_attributes**: list where each element is a list, the first element is the custom attribute id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project_id": 3,
        "bulk_userstory_custom_attributes": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/userstory-custom-attributes/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

17. User story custom attributes values

17.1. Get

To get a user story custom attribute send a GET request specifying the user story custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/userstories/custom-attributes-values/1
```

The HTTP response is a 200 OK and the response body is a JSON user story custom attribute detail object

17.2. Edit

To edit user story custom attributes values send a PUT or a PATCH specifying the user story id in the url. "attribute_values" must be a JSON object with pairs user story custom atribute id - value. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "attributes_values": {"1": "240 min"},
        "version": 2
      }' \
  https://api.taiga.io/api/v1/userstories/custom-attributes-values/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON user story custom attribute detail object

18. Tasks

18.1. List

To list tasks send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks
```

The HTTP response is a 200 OK and the response body is a JSON list of task list objects

The results can be filtered using the following parameters:

- **project**: project id
- **user_story**: user story id
- **milestone**: milestone id
- **watchers**: watching user id
- **assigned_to**: assigned to user id
- **status__is_closed**: (true|false)

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks?project=1
```

18.2. Create

To create tasks send a POST request with the following data:

- **assigned_to**: user id
- **blocked_note**: reason why the task is blocked
- **description**: string
- **is_blocked**: boolean
- **is_closed**: boolean

- **milestone**: milestone id
- **project** (required): project id
- **user_story**: user story id
- **status**: status id
- **subject** (required)
- **tags**: array of strings
- **us_order**: order in the user story,
- **taskboard_order**: order in the taskboard,
- **is_iocaine**: boolean,
- **external_reference**: tuple of ("service", serviceId),
- **watchers**: array of watcher id's

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "assigned_to": null,
          "blocked_note": "blocking reason",
          "description": "Implement API CALL",
          "is_blocked": false,
          "is_closed": true,
          "milestone": null,
          "project": 3,
          "user_story": 17,
          "status": 13,
          "subject": "Customer personal data",
          "tags": [
              "service catalog",
              "customer"
          ],
          "us_order": 1,
          "taskboard_order": 1,
          "is_iocaine": false,
          "external_reference": null,
          "watchers": []
      }' \
  https://api.taiga.io/api/v1/tasks
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 3,
        "subject": "Customer personal data"
      }' \
  https://api.taiga.io/api/v1/tasks
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON task detail object

18.3. Get

To get a task send a GET request specifying the task id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1
```

The HTTP response is a 200 OK and the response body is a JSON task detail (GET) object

18.4. Get by ref

To get a task send a GET request specifying the task reference and the project id as parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/by_ref?ref=1&project=1
```

The HTTP response is a 200 OK and the response body is a JSON task detail (GET) object

18.5. Edit

To edit tasks send a PUT or a PATCH specifying the task id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "subject": "Patching subject"
      }' \
  https://api.taiga.io/api/v1/tasks/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON task detail object

18.6. Delete

To delete tasks send a DELETE specifying the task id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

18.7. Bulk creation

To create multiple tasks at the same time send a POST request with the following data:

- **project_id** (required)
- **status_id**
- **sprint_id**: milestone id (optional)
- **us_id**: user story id (optional)
- **bulk_tasks**: task subjects, one per line

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project_id": 3,
        "bulk_tasks": "Task 1 \n Task 2 \n Task 3"
      }' \
  https://api.taiga.io/api/v1/tasks/bulk_create
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON list of

task detail object

18.8. Vote a task

To vote tasks send a POST request specifying the task id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/upvote
```

The HTTP response is a 200 OK with an empty body response

18.9. Remove vote from a task

To remove a vote from a task send a POST request specifying the task id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/downvote
```

When remove of the vote succeeded, the HTTP response is a 200 OK with an empty body response

18.10. Get task voters list

To get the list of voters from a task send a GET request specifying the task id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/voters
```

The HTTP response is a 200 OK and the response body is a JSON list of task voter object

18.11. Watch a task

To watch a task send a POST request specifying the task id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/watch
```

The HTTP response is a 200 OK with an empty body response

18.12. Stop watching a task

To stop watching a task send a POST request specifying the task id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

18.13. List task watchers

To get the list of watchers from a task send a GET request specifying the task id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of task watcher object

18.14. List attachments

To list task attachments send a GET request with the following parameters:

- **project**: project id
- **object_id**: task id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/attachments?object_id=81\&project=3
```

The HTTP response is a 200 OK and the response body is a JSON list of attachment detail objects

18.15. Create attachment

To create task attachments send a POST request with the following data:

- **object_id** (required): task id
- **project** (required): project id
- **attached_file** (required): attaching file
- **description**
- **is_deprecated**

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "object_id=81" \
  -F "project=3" \
  -F "attached_file=@/tmp/test.png" \
  https://api.taiga.io/api/v1/tasks/attachments
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON attachment detail object

18.16. Get attachment

To get a task attachment send a GET request specifying the task attachment id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/attachments/415
```

The HTTP response is a 200 OK and the response body is a JSON attachment detail object

18.17. Edit attachment

To edit task attachments send a PUT or a PATCH specifying the task attachment id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "description=patching description" \
  https://api.taiga.io/api/v1/tasks/attachments/417
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON attachment detail object

18.18. Delete attachment

To delete task attachments send a DELETE specifying the task attachment id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/attachments/415
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

19. Task status

19.1. List

To list task status send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-statuses
```

The HTTP response is a 200 OK and the response body is a JSON list of task status detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-statuses?project=1
```

19.2. Create

To create task statuses send a POST request with the following data:

- **color**: in hexadecimal
- **is_closed**: (true | false)
- **name** (required)
- **order**: integer

- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "is_closed": true,
        "name": "New status",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/task-statuses
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "New status name"
      }' \
  https://api.taiga.io/api/v1/task-statuses
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON task status detail object

19.3. Get

To get a task status send a GET request specifying the task status id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-statuses/1
```

The HTTP response is a 200 OK and the response body is a JSON task status detail object

19.4. Edit

To edit task statuses send a PUT or a PATCH specifying the task status id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Patch status name"
      }' \
  https://api.taiga.io/api/v1/task-statuses/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON task status detail object

19.5. Delete

To delete task satuses send a DELETE specifying the task status id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-statuses/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

19.6. Bulk update order

To update the order of multiple task statuses at the same time send a POST request with the following data:

- **project** (required)
- **bulk_task_statuses**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_task_statuses": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/task-statuses/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

20. Task custom attribute

20.1. List

To list task custom attributes send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-custom-attributes
```

The HTTP response is a 200 OK and the response body is a JSON list of task custom attribute detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-custom-attributes?project=1
```

20.2. Create

To create task custom attributes send a POST request with the following data:

- **name**: (required) text
- **description**: text
- **order**: integer
- **project**: (required) integer, project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Duration",
        "description": "Duration in minutes",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/task-custom-attributes
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Duration",
          "project": 3
      }' \
  https://api.taiga.io/api/v1/task-custom-attributes
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON task custom attribute detail object

20.3. Get

To get a task custom attribute send a GET request specifying the task custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-custom-attributes/1
```

The HTTP response is a 200 OK and the response body is a JSON task custom attribute detail object

20.4. Edit

To edit task custom attributes send a PUT or a PATCH specifying the task custom attribute id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Duration"
      }' \
  https://api.taiga.io/api/v1/task-custom-attributes/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON task custom attribute detail object

20.5. Delete

To delete task custom attributes send a DELETE specifying the task custom attribute id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/task-custom-attributes/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

20.6. Bulk update order

To update the order of multiple task custom attributes at the same time send a POST request with the following data:

- **project** (required)
- **bulk_task_custom_attributes**: list where each element is a list, the first element is the custom attribute id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_task_custom_attributes": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/task-custom-attributes/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

21. Task custom attributes values

21.1. Get

To get a task custom attribute send a GET request specifying the task custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/tasks/custom-attributes-values/1
```

The HTTP response is a 200 OK and the response body is a JSON task custom attribute detail object

21.2. Edit

To edit task custom attributes values send a PUT or a PATCH specifying the task id in the url. "attribute_values" must be a JSON object with pairs task custom atribute id - value. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "attributes_values": {"1": "240 min"},
          "version": 2
      }' \
  https://api.taiga.io/api/v1/tasks/custom-attributes-values/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON task custom attribute detail object

22. Issues

22.1. List

To list issues send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues
```

The HTTP response is a 200 OK and the response body is a JSON list of issue detail list objects

The results can be filtered using the following parameters:

- **project**: project id
- **status**: status id
- **severity**: severity id
- **priority**: priority id
- **owner**: owner user id
- **assigned_to**: assigned to user id
- **tags**: separated by ","
- **type**: issue type id
- **watchers**: watching user id
- **status__is_closed**: (true|false)

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues?project=1
```

The results can be ordered using the order_by parameter with the values:

- **type**
- **severity**
- **status**
- **priority**
- **created_date**
- **modified_date**
- **owner**
- **assigned_to**
- **subject**

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues?project=1\&order_by=priority
```

22.2. Create

To create issues send a POST request with the following data:

- **assigned_to**: user id
- **blocked_note**: reason why the issue is blocked
- **description**: string
- **is_blocked**: boolean
- **is_closed**: boolean
- **milestone**: milestone id
- **project** (required): project id
- **status**: status id
- **severity**: severity id
- **priority**: priority id
- **type**: type id
- **subject** (required)
- **tags**: array of strings
- **watchers**: array of watcher id's

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "assigned_to": null,
          "blocked_note": "blocking reason",
          "description": "Implement API CALL",
          "is_blocked": false,
          "is_closed": true,
          "milestone": null,
          "project": 3,
          "status": 13,
          "severity": 2,
          "priority": 3,
          "type": 1,
          "subject": "Customer personal data",
          "tags": [
              "service catalog",
              "customer"
          ],
          "watchers": []
      }' \
  https://api.taiga.io/api/v1/issues
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "subject": "Customer personal data"
      }' \
  https://api.taiga.io/api/v1/issues
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON issue detail object

22.3. Get

To get an issue send a GET request specifying the issue id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1
```

The HTTP response is a 200 OK and the response body is a JSON issue detail (GET) object

22.4. Get by ref

To get an issue send a GET request specifying the issue reference end project id as parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/by_ref?ref=1&project=1
```

The HTTP response is a 200 OK and the response body is a JSON issue detail (GET) object

22.5. Edit

To edit issues send a PUT or a PATCH specifying the issue id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "subject": "Patching subject"
     }' \
  https://api.taiga.io/api/v1/issues/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON issue detail object

22.6. Delete

To delete issues send a DELETE specifying the issue id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

22.7. Vote an issue

To vote issues send a POST specifying the issue id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/upvote
```

When vote succeeded, the HTTP response is a 200 OK with an empty body response

22.8. Remove vote from an issue

To remove a vote from an issue send a POST specifying the issue id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/downvote
```

When remove of the vote succeeded, the HTTP response is a 200 OK with an empty body response

22.9. Get issue voters list

To get the list of voters from an issue send a GET request specifying the issue id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/voters
```

The HTTP response is a 200 OK and the response body is a JSON issue voters detail object

22.10. Watch an issue

To watch an issue send a POST request specifying the issue id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/watch
```

The HTTP response is a 200 OK with an empty body response

22.11. Stop watching an issue

To stop watching an issue send a POST request specifying the issue id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

22.12. List issue watchers

To get the list of watchers from an issue send a GET request specifying the issue id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of issue watcher object

22.13. List attachments

To list issue attachments send a GET request with the following parameters:

- **project**: project id
- **object_id**: issue id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/attachments?object_id=81\&project=3
```

The HTTP response is a 200 OK and the response body is a JSON list of attachment detail objects

22.14. Create attachment

To create issue attachments send a POST request with the following data:

- **object_id** (required): issue id
- **project** (required): project id
- **attached_file** (required): attaching file
- **description**
- **is_deprecated**

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "object_id=81" \
  -F "project=3" \
  -F "attached_file=@/tmp/test.png" \
  https://api.taiga.io/api/v1/issues/attachments
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON attachment detail object

22.15. Get attachment

To get an issue attachment send a GET request specifying the issue attachment id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/attachments/415
```

The HTTP response is a 200 OK and the response body is a JSON attachment detail object

22.16. Edit attachment

To edit issue attachments send a PUT or a PATCH specifying the issue attachment id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "description=patching description" \
  https://api.taiga.io/api/v1/issues/attachments/417
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON attachment detail object

22.17. Delete attachment

To delete issue attachments send a DELETE specifying the issue attachment id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/attachments/415
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

23. Issue status

23.1. List

To list issue status send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-statuses
```

The HTTP response is a 200 OK and the response body is a JSON list of issue status detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-statuses?project=1
```

23.2. Create

To create issue statuses send a POST request with the following data:

- **color**: in hexadecimal
- **is_closed**: (true|false)

- **name** (required)
- **order**: integer
- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "is_closed": true,
        "name": "New status",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/issue-statuses
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "New status name"
      }' \
  https://api.taiga.io/api/v1/issue-statuses
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON issue status detail object

23.3. Get

To get a issue status send a GET request specifying the issue status id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-statuses/1
```

The HTTP response is a 200 OK and the response body is a JSON issue status detail object

23.4. Edit

To edit issue statuses send a PUT or a PATCH specifying the issue status id in the url. In a PATCH

request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Patch status name"
      }' \
  https://api.taiga.io/api/v1/issue-statuses/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON issue status detail object

23.5. Delete

To delete issue satuses send a DELETE specifying the issue status id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-statuses/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

23.6. Bulk update order

To update the order of multiple issue statuses at the same time send a POST request with the following data:

- **project** (required)
- **bulk_issue_statuses**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project_id": 3,
        "bulk_issue_statuses": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/issue-statuses/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

24. Issue types

24.1. List

To list issue types send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-types
```

The HTTP response is a 200 OK and the response body is a JSON list of issue type detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-types?project=1
```

24.2. Create

To create issue types send a POST request with the following data:

- **color**: in hexadecimal
- **name** (required)
- **order**: integer
- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "name": "New type",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/issue-types
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "New type name"
      }' \
  https://api.taiga.io/api/v1/issue-types
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON issue type detail object

24.3. Get

To get an issue type send a GET request specifying the issue type id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-types/1
```

The HTTP response is a 200 OK and the response body is a JSON issue type detail object

24.4. Edit

To edit issue types send a PUT or a PATCH specifying the issue type id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Patch type name"
      }' \
  https://api.taiga.io/api/v1/issue-types/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON issue type detail object

24.5. Delete

To delete issue statuses send a DELETE specifying the issue type id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-types/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

24.6. Bulk update order

To update the order of multiple issue types at the same time send a POST request with the following data:

- **project** (required)
- **bulk_issue_types**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_issue_types": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/issue-types/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

25. Priorities

25.1. List

To list priorities send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/priorities
```

The HTTP response is a 200 OK and the response body is a JSON list of priority detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/priorities?project=1
```

25.2. Create

To create priorities send a POST request with the following data:

- **color**: in hexadecimal
- **name** (required)
- **order**: integer
- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "name": "New priority",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/priorities
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "New priority name"
      }' \
  https://api.taiga.io/api/v1/priorities
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON priority detail object

25.3. Get

To get a priority send a GET request specifying the priority id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/priorities/1
```

The HTTP response is a 200 OK and the response body is a JSON priority detail object

25.4. Edit

To edit priorities send a PUT or a PATCH specifying the priority id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Patch name"
      }' \
  https://api.taiga.io/api/v1/priorities/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON priority detail object

25.5. Delete

To delete priorities send a DELETE specifying the priority id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/priorities/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

25.6. Bulk update order

To update the order of multiple priorities at the same time send a POST request with the following data:

- **project** (required)
- **bulk_priorities**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_priorities": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/priorities/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

26. Severities

26.1. List

To list severities send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/severities
```

The HTTP response is a 200 OK and the response body is a JSON list of severity detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/severities?project=1
```

26.2. Create

To create severities send a POST request with the following data:

- **color**: in hexadecimal
- **name** (required)
- **order**: integer
- **project**: (required): project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "color": "#AAAAAA",
        "name": "New severity",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/severities
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project": 3,
          "name": "New severity name"
      }' \
  https://api.taiga.io/api/v1/severities
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON severity detail object

26.3. Get

To get a severity send a GET request specifying the severity id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/severities/1
```

The HTTP response is a 200 OK and the response body is a JSON severity detail object

26.4. Edit

To edit severities send a PUT or a PATCH specifying the severity id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Patch name"
      }' \
  https://api.taiga.io/api/v1/severities/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON severity detail object

26.5. Delete

To delete severities send a DELETE specifying the severity id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/severities/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

26.6. Bulk update order

To update the order of multiple severities at the same time send a POST request with the following data:

- **project** (required)
- **bulk_severities**: list where each element is a list, the first element is the status id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_severities": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/severities/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

27. Issue custom attribute

27.1. List

To list issue custom attributes send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-custom-attributes
```

The HTTP response is a 200 OK and the response body is a JSON list of issue custom attribute detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-custom-attributes?project=1
```

27.2. Create

To create issue custom attributes send a POST request with the following data:

- **name**: (required) text
- **description**: text
- **order**: integer
- **project**: (required) integer, project id

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "Duration",
        "description": "Duration in minutes",
        "order": 8,
        "project": 3
    }' \
  https://api.taiga.io/api/v1/issue-custom-attributes
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Duration",
          "project": 3
      }' \
  https://api.taiga.io/api/v1/issue-custom-attributes
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON issue custom attribute detail object

27.3. Get

To get a issue custom attribute send a GET request specifying the issue custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-custom-attributes/1
```

The HTTP response is a 200 OK and the response body is a JSON issue custom attribute detail object

27.4. Edit

To edit issue custom attributes send a PUT or a PATCH specifying the issue custom attribute id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "name": "Duration"
      }' \
  https://api.taiga.io/api/v1/issue-custom-attributes/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON issue custom attribute detail object

27.5. Delete

To delete issue custom attributes send a DELETE specifying the issue custom attribute id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issue-custom-attributes/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

27.6. Bulk update order

To update the order of multiple issue custom attributes at the same time send a POST request with the following data:

- **project** (required)
- **bulk_issue_custom_attributes**: list where each element is a list, the first element is the custom attribute id and the second the new order

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "project_id": 3,
          "bulk_issue_custom_attributes": [[1,10], [2,5]]
      }' \
  https://api.taiga.io/api/v1/issue-custom-attributes/bulk_update_order
```

When the update is successful, the HTTP response is a 204 NO CONTENT with an empty body response

28. Issue custom attributes values

28.1. Get

To get a issue custom attribute send a GET request specifying the issue custom attribute id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/issues/custom-attributes-values/1
```

The HTTP response is a 200 OK and the response body is a JSON issue custom attribute detail object

28.2. Edit

To edit issue custom attributes values send a PUT or a PATCH specifying the issue id in the url. "attribute_values" must be a JSON object with pairs issue custom atribute id - value. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "attributes_values": {"1": "240 min"},
          "version": 2
      }' \
  https://api.taiga.io/api/v1/issues/custom-attributes-values/1
```

When the update is successful, the HTTP response is a 200 OK and the response body is a JSON issue custom attribute detail object

29. Wiki pages

29.1. List

To list wiki pages send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki
```

The HTTP response is a 200 OK and the response body is a JSON list of wiki page detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki?project=1
```

29.2. Create

To create wiki pages send a POST request with the following data:

- **project** (required): project id
- **slug** (required): slug
- **content** (required): string
- **watchers**: array of watcher id's

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 1,
        "slug": "home",
        "content": "Lorem ipsum dolor.",
        "watchers": []
    }' \
  https://api.taiga.io/api/v1/wiki
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 1,
        "slug": "home",
        "content": "Lorem ipsum dolor."
      }' \
  https://api.taiga.io/api/v1/wiki
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON wiki page detail object

29.3. Get

To get a wiki page send a GET request specifying the wiki page id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/1
```

The HTTP response is a 200 OK and the response body is a JSON wiki page detail object

29.4. Get by slug

To get a wiki page send a GET request specifying the wiki page slug and the project id as parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/by_slug?slug=home\&project=1
```

The HTTP response is a 200 OK and the response body is a JSON wiki page detail object

29.5. Edit

To edit wiki pages send a PUT or a PATCH specifying the wiki page id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
          "subject": "Patching subject"
      }' \
  https://api.taiga.io/api/v1/wiki/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON wiki page detail object

29.6. Delete

To delete wiki page send a DELETE specifying the wiki page id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

29.7. Watch a wiki page

To watch a wiki page send a POST request specifying the wiki page id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/1/watch
```

The HTTP response is a 200 OK with an empty body response

29.8. Stop watching a wiki page

To stop watching a wiki page send a POST request specifying the wiki page id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/1/unwatch
```

The HTTP response is a 200 OK with an empty body response

29.9. List wiki page watchers

To get the list of watchers from a wiki page send a GET request specifying the wiki page id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/1/watchers
```

The HTTP response is a 200 OK and the response body is a JSON list of wiki page watcher object

29.10. List attachments

To list wiki page attachments send a GET request with the following parameters:

- **project**: project id
- **object_id**: wiki page id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/attachments?object_id=81\&project=3
```

The HTTP response is a 200 OK and the response body is a JSON list of attachment detail objects

29.11. Create attachment

To create wiki page attachments send a POST request with the following data:

- **object_id** (required): wiki page id
- **project** (required): project id
- **attached_file** (required): attaching file
- **description**
- **is_deprecated**

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "object_id=81" \
  -F "project=3" \
  -F "attached_file=@/tmp/test.png" \
  https://api.taiga.io/api/v1/wiki/attachments
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON attachment detail object

29.12. Get attachment

To get an wiki page attachment send a GET request specifying the wiki page attachment id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/attachments/415
```

The HTTP response is a 200 OK and the response body is a JSON attachment detail object

29.13. Edit attachment

To edit wiki page attachments send a PUT or a PATCH specifying the wiki page attachment id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "description=patching description" \
  https://api.taiga.io/api/v1/wiki/attachments/417
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON attachment detail object

29.14. Delete attachment

To delete wiki page attachments send a DELETE specifying the wiki page attachment id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki/attachments/415
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

30. Wiki links

30.1. List

To list wiki links send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki-links
```

The HTTP response is a 200 OK and the response body is a JSON list of wiki link detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki-links?project=1
```

30.2. Create

To create wiki links send a POST request with the following data:

- **project** (required): project id
- **title** (required): string
- **href** (required): wiki page slug
- **order**: integer

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 1,
        "title": "Home page",
        "href": "home",
        "order": 1
    }' \
  https://api.taiga.io/api/v1/wiki-links
```

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 1,
        "title": "Home page",
        "href": "home"
    }' \
  https://api.taiga.io/api/v1/wiki-links
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON wiki link detail object

30.3. Get

To get a wiki link send a GET request specifying the wiki link id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki-links/1
```

The HTTP response is a 200 OK and the response body is a JSON wiki link detail object

30.4. Edit

To edit wiki links send a PUT or a PATCH specifying the wiki link id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "subject": "Patching subject"
      }' \
  https://api.taiga.io/api/v1/wiki-links/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON wiki link detail object

30.5. Delete

To delete wiki link send a DELETE specifying the wiki link id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/wiki-links/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

31. History

31.1. Get user story, task, issue or wiki page history

To get the history of a user story, task, issue or wiki page send a GET request specifying the id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/history/userstory/1
```

The HTTP response is a 200 OK and the response body is a JSON of a list of history entry detail objects

31.2. Delete comment

To delete history comment send a POST specifying the history entry id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/history/userstory/1/delete_comment?id=10
```

When deleted succesfully, the HTTP response is a 204 NO CONTENT with an empty body response

31.3. Undelete comment

To undelete history comment send a POST specifying the history entry id in the url

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/history/userstory/1/undelete_comment?id=10
```

When deleted succesfully, the HTTP response is a 204 NO CONTENT with an empty body response

32. Users

32.1. List

To list users send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users
```

The HTTP response is a 200 OK and the response body is a JSON list of user detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users?project=1
```

32.2. Get

To get a user send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1
```

The HTTP response is a 200 OK and the response body is a JSON user detail object

32.3. Me

To get your own user send a GET request to the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/me
```

The HTTP response is a 200 OK and the response body is a JSON user detail object

32.4. Get user stats

To get the stats from a user send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/stats
```

The HTTP response is a 200 OK and the response body is a JSON user stats object

32.5. Get watched content

To get the user watched content send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/watched
```

The HTTP response is a 200 OK and the response body is a list of JSON watched detail object

The results can be filtered using the following parameters:

- **type**: of the content. Possible values: project, userstory, task and issue
- **q**: text to search in the subject of the element

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/watched?type=project&q=test
```

32.6. Get liked content

To get the user liked content send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/liked
```

The HTTP response is a 200 OK and the response body is a list of JSON liked detail object

The results can be filtered using the following parameters:

- **q**: text to search in the subject of the element

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/liked?q=test
```

32.7. Get voted content

To get the user voted content send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/voted
```

The HTTP response is a 200 OK and the response body is a list of JSON voted detail object

The results can be filtered using the following parameters:

- **type**: of the content. Possible values: userstory, task and issue
- **q**: text to search in the subject of the element

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/liked?type=userstory&q=test
```

32.8. Edit

To edit users send a PUT or a PATCH specifying the user id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
         "username": "patchedusername"
      }' \
  https://api.taiga.io/api/v1/users/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON user detail object

32.9. Delete

To delete users send a DELETE specifying the user id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

32.10. Get contacts

To get a user contacts send a GET request specifying the user id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/1/contacts
```

The HTTP response is a 200 OK and the response body is a list of JSON contact detail object

32.11. Cancel

To cancel a user account send a POST with the following data

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
         "cancel_token": "'${CANCEL_TOKEN}'"
      }' \
  https://api.taiga.io/api/v1/users/cancel
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

32.12. Change avatar

To change your user avatar send a POST with the following data

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F "avatar=@/tmp/test.png" \
  https://api.taiga.io/api/v1/users/change_avatar
```

When the change is successful, the HTTP response is a 200 OK and the response body is a JSON user detail object

32.13. Remove avatar

To remove your user avatar send a POST with the following data

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/users/remove_avatar
```

When the change is successful, the HTTP response is a 200 OK and the response body is a JSON user detail object

32.14. Change email

To change your user email send a POST with the following data

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "email_token": "'${EMAIL_TOKEN}'"
      }' \
  https://api.taiga.io/api/v1/users/change_email
```

When the change is successful, the HTTP response is a 200 OK and the response body is a JSON user detail object

32.15. Change password

To change your user password send a POST with the following data

```
curl -X POST \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "current_password": "'${CURRENT_PASSWORD}'",
        "password": "'${NEW_PASSWORD}'"
      }' \
  https://api.taiga.io/api/v1/users/change_password
```

When the change succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

32.16. Password recovery

To request a user password recovery send a POST with the following data:

- **username** (required): this field also supports the user email

```
curl -X POST \
  -d '{
          "username": "'${USERNAME_OR_EMAIL}'"
      }' \
  https://api.taiga.io/api/v1/users/password_recovery
```

When the password recovery request succeeded, the HTTP response is a 200 OK and the response body is a JSON object:

```
{
    "detail": "Mail sended successfully!",
    "email": "${USER_EMAIL}"
}
```

32.17. Change password from recovery

To change a user password from a request recovery send a POST with the following data

```
curl -X POST \
  -d '{
          "token": "'${CHANGE_PASSWORD_TOKEN}'",
          "password"; "'${NEW_TOKEN}'"
      }' \
  https://api.taiga.io/api/v1/users/change_password_from_recovery
```

When the password change succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

33. Notify policies

33.1. List

To list the notify policies of the current user send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/notify-policies
```

The HTTP response is a 200 OK and the response body is a JSON list of notify policy detail objects

33.2. Get

To get a notify policy send a GET request specifying the notify policy id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/notify-policies/1
```

The HTTP response is a 200 OK and the response body is a JSON notify policy detail object

33.3. Edit

To edit notify policies send a PUT or a PATCH specifying the notify policy id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "notify_level": 2
    }' \
  https://api.taiga.io/api/v1/notify-policies/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON notify policy detail object

34. Feedback

34.1. Create

To create feedback send a POST request with the following data:

- **comment** (required)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "comment": "Testing feedback"
      }' \
  https://api.taiga.io/api/v1/feedback
```

When created successfully, the HTTP response is a 201 Created and the response body is a JSON feedback object

35. Export/Import

35.1. Export

To get a project dump send a GET request with the project id:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/exporter/1
```

Depending on server configuration it can return two results:

- A 202 Accepted and as response body a JSON of export request accepted.
- A 200 OK and as response body a JSON of export detail for synch mode.

35.2. Import

To load a project dump send a POST request with the following file:

- **dump** (required)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -F 'dump=@my-dump-file.json' \
  https://api.taiga.io/api/v1/importer/load_dump
```

Depending on server configuration it can return two results:

- A 202 Accepted and as response body a JSON of import request accepted.
- A 201 Created and the response body is a JSON of project detail object

36. Webhooks

36.1. List

To list webhooks send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooks
```

The HTTP response is a 200 OK and the response body is a JSON list of webhook detail objects

The results can be filtered using the following parameters:

- **project**: project id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooks?project=1
```

36.2. Create

To create webhook send a POST request with the following data:

- **project** (required): project id
- **name** (required): string
- **url** (required): payload url
- **key** (required): secret key

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "project": 1,
        "name": "My service webhook",
        "url": "http://myservice.com/webhooks",
        "key": "my-very-secret-key"
      }' \
  https://api.taiga.io/api/v1/webhooks
```

When the creation is successful, the HTTP response is a 201 Created and the response body is a JSON webhook detail object

36.3. Get

To get a webhook send a GET request specifying the webhook id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooks/1
```

The HTTP response is a 200 OK and the response body is a JSON webhook detail object

36.4. Edit

To edit a webhook send a PUT or a PATCH specifying the webhook id in the url. In a PATCH request you just need to send the modified data, in a PUT one the whole object must be sent.

```
curl -X PATCH \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  -d '{
        "name": "My service name"
      }' \
  https://api.taiga.io/api/v1/webhooks/1
```

When the creation is successful, the HTTP response is a 200 OK and the response body is a JSON webhook detail object

36.5. Delete

To delete a webhook send a DELETE specifying the webhook id in the url

```
curl -X DELETE \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooks/1
```

When delete succeeded, the HTTP response is a 204 NO CONTENT with an empty body response

36.6. Test

To test a webhook send a POST request specifying the webhook id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooks/1/test
```

The HTTP response is a 200 OK and the response body is a JSON webhook log detail object with the resault of the test.

36.7. Logs list

To list webhook logs send a GET request to the url:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooklogs
```

The HTTP response is a 200 OK and the response body is a JSON list of webhook log detail objects

The results can be filtered using the following parameters:

- **webhook**: webhook id

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooklogs?webhook=1
```

36.8. Log get

To get a webhook log send a GET request specifying the webhook log id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooklogs/1
```

The HTTP response is a 200 OK and the response body is a JSON webhook log detail object

36.9. Resend request

To resend a request from a webhook log send a POST request specifying the webhook log id in the url

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/webhooklogs/1/resend
```

The HTTP response is a 200 OK and the response body is a JSON webhook log detail object with the resault of the resend.

37. Timelines

37.1. List user timeline

To list a user timeline send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/timeline/user/1
```

This API call returns only actions directly executed by the specified user.

The HTTP response is a 200 OK and the response body is a JSON list of timeline entry detail

37.2. List profile timeline

To list a profile timeline send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/timeline/profile/1
```

This API call returns actions executed by the specified user and related to them, watching objects, actions by related team members, belonging to projects...

The HTTP response is a 200 OK and the response body is a JSON list of timeline entry detail

37.3. List project timeline

To list a project timeline send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/timeline/project/1
```

This API call returns actions executed by different users related to the specified project.

The HTTP response is a 200 OK and the response body is a JSON list of timeline entry detail

38. Locales

38.1. List

To list the available locales send a GET request with the following parameters:

```
curl -X GET \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${AUTH_TOKEN}" \
  https://api.taiga.io/api/v1/locales
```

The HTTP response is a 200 OK and the response body is a JSON list of locale objects

39. Objects Summary

39.1. Attachment


```
{
    "id": 415,
    "project": 3,
    "owner": 2,
    "name": "img_20141102_214512.jpg",
    "attached_file":
"attachments/8/1/6/e/68a8e4edb4ba422d201d65406b161be03e052d9b63e265aaa826179497c8/img_201
41102_214512.jpg",
    "size": 173780,
    "url":
"https://media.taiga.io/attachments/8/1/6/e/68a8e4edb4ba422d201d65406b161be03e052d9b63e26
5aaa826179497c8/img_20141102_214512.jpg",
    "description": "asdasd",
    "is_deprecated": false,
    "created_date": "2014-11-11T14:22:29+0000",
    "modified_date": "2014-11-11T14:22:43+0000",
    "object_id": 81,
    "order": 0,
    "sha1": "da2631d805f12a1b533738a0912e9b9c2261dbef"
}
```


39.2. Application object


```
{
    "id": "a60c3208-5234-11e5-96df-68f72800aadd",
    "name": "test",
    "web": "qweqweqwe",
    "description": "asdasdasdasd",
    "icon_url": "http://www.iconarchive.com/icons/yellowicon/game-stars/256/Mario-
icon.png"
}
```


39.3. Application token object

```
{
    "user": 4,
    "id": 5,
    "application": {
        "id": "a60c3208-5234-11e5-96df-68f72800aadd",
        "name": "Testing application",
        "web": "http://taiga.io",
        "description": "Testing external app",
        "icon_url": "https://tree.taiga.io/images/beta.png"
    },
    "auth_code": "5a6e869a-52e6-11e5-ad57-68f72800aadd",
    "next_url": "http://tree.taiga.io/redirect?auth_code=5a6e869a-52e6-11e5-ad57-
68f72800aadd"
}
```


If id and auth_code are null it means there is no application token generated and you need to authorize one.

39.4. Authorization code object


```
{
    "auth_code": "5a6e869a-52e6-11e5-ad57-68f72800aadd",
    "state": "random-state",
    "next_url": "http://tree.taiga.io/redirect?auth_code=5a6e869a-52e6-11e5-ad57-
68f72800aadd"
}
```


39.5. Cyphered token object


```
{
    "cyphered_token": "eyJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiQTEyOEtXIn0.E-Ee1cRgG0JEd90yJu-
Dgl_vwKHTHdPy2YHRbCsMvfiJx0OvR12E8g.kGwJPnWQJecFPEae.ebQtpRNPbKh6FBS-
LSUhw1xNARl0Q5loCO4fAk00LHFqcDpAwba7LHeR3MPx9T9LfA.KM-Id_041g8OdWaseGyV8g"
}
```


39.6. User detail

```
{
    "big_photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
    "bio": "",
    "color": "black",
    "lang": "",
    "timezone": "",
    "email": "beta.testing@taiga.io",
    "full_name": "Beta testing",
    "full_name_display": "Beta testing",
    "github_id": null,
    "id": 1,
    "is_active": true,
    "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
    "username": "beta.tester"
}
```


39.7. User contact detail


```
{
    "id": 32090,
    "username": "beta.tester"
    "full_name": "Beta testing",
    "full_name_display": "Beta testing",
    "color": "#2c3e58",
    "bio": "",
    "lang": "",
    "timezone": "",
    "is_active": true,
    "photo": "http//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
    "big_photo":
"http//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
    "roles": [
        "Product Owner"
    ],
    "projects_with_me": []
}
```


39.8. User authentication-detail

```
{
    "auth_token":
"eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjo3fq:1XmPud:LKXVD9Z0rmHJjiyy0m4YaaHlQS1",
    "bio": "",
    "is_active": true,
    "email": "beta.testing@taiga.io",
    "github_id": null,
    "color": "#FC8EAC",
    "lang": "",
    "full_name_display": "Beta testing",
    "timezone": "",
    "id": 7,
    "full_name": "Beta testing",
    "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
    "username": "beta.tester",
    "big_photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80"
}
```


39.9. User stats detail


```
{
    "roles": [
        "Back"
    ],
    "total_num_closed_userstories": 0,
    "total_num_contacts": 8,
    "total_num_projects": 1
}
```


39.10. Search results detail


```
{
    "wikipages": [{"id": 1, "slug": "wiki-page", "content": "wiki content"}, ...],
    "userstories":  [{"id": 1, "ref": 10, "subject": "User story subject", "status": 23,
"total_points": 12.5}, ...],
    "issues":  [{"id": 1, "ref": 20, "subject": "Issue subject", "status": 10,
"assigned_to": 12}, ...],
    "tasks":  [{"id": 1, "ref": 30, "subject": "Task  subject", "status": 15,
"assigned_to": 54}, ...],
    "count": 8
}
```

39.11. User storage data


```
{
    "key": "favorite-forest",
    "value": "Taiga",
    "created_date": "2014-11-13T16:58:35+0000",
    "modified_date": "2014-11-13T16:58:35+0000"
}
```


40. Project templates detail


```
{
    "default_options": {
        "us_status": "New",
        "points": "?",
        "priority": "Normal",
        "severity": "Normal",
        "task_status": "New",
        "issue_type": "Bug",
        "issue_status": "New"
    },
    "us_statuses": [
        {
            "wip_limit": null,
            "color": "#999999",
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false
        },
        {
            "wip_limit": null,
            "color": "#f57900",
            "name": "Ready",
            "slug": "ready",
            "order": 2,
            "is_closed": false
        },
        {
            "wip_limit": null,
            "color": "#729fcf",
            "name": "In progress",
            "slug": "in-progress",
            "order": 3,
```

```
            "is_closed": false
        },
        {
            "wip_limit": null,
            "color": "#4e9a06",
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 4,
            "is_closed": false
        },
        {
            "wip_limit": null,
            "color": "#cc0000",
            "name": "Done",
            "slug": "done",
            "order": 5,
            "is_closed": true
        }
    ],
    "points": [
        {
            "value": null,
            "name": "?",
            "order": 1
        },
        {
            "value": 0.0,
            "name": "0",
            "order": 2
        },
        {
            "value": 0.5,
            "name": "1/2",
            "order": 3
        },
        {
            "value": 1.0,
            "name": "1",
            "order": 4
        },
        {
            "value": 2.0,
            "name": "2",
            "order": 5
        },
        {
            "value": 3.0,
            "name": "3",
```

```
            "order": 6
        },
        {
            "value": 5.0,
            "name": "5",
            "order": 7
        },
        {
            "value": 8.0,
            "name": "8",
            "order": 8
        },
        {
            "value": 10.0,
            "name": "10",
            "order": 9
        },
        {
            "value": 15.0,
            "name": "15",
            "order": 10
        },
        {
            "value": 20.0,
            "name": "20",
            "order": 11
        },
        {
            "value": 40.0,
            "name": "40",
            "order": 12
        }
    ],
    "task_statuses": [
        {
            "color": "#999999",
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false
        },
        {
            "color": "#729fcf",
            "name": "In progress",
            "slug": "in-progress",
            "order": 2,
            "is_closed": false
        },
```

```
        {
            "color": "#f57900",
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 3,
            "is_closed": true
        },
        {
            "color": "#4e9a06",
            "name": "Closed",
            "slug": "closed",
            "order": 4,
            "is_closed": true
        },
        {
            "color": "#cc0000",
            "name": "Needs Info",
            "slug": "needs-info",
            "order": 5,
            "is_closed": false
        }
    ],
    "issue_statuses": [
        {
            "color": "#999999",
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false
        },
        {
            "color": "#729fcf",
            "name": "In progress",
            "slug": "in-progress",
            "order": 2,
            "is_closed": false
        },
        {
            "color": "#f57900",
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 3,
            "is_closed": true
        },
        {
            "color": "#4e9a06",
            "name": "Closed",
            "slug": "closed",
```

```
            "order": 4,
            "is_closed": true
        },
        {
            "color": "#cc0000",
            "name": "Needs Info",
            "slug": "needs-info",
            "order": 5,
            "is_closed": false
        },
        {
            "color": "#d3d7cf",
            "name": "Rejected",
            "slug": "rejected",
            "order": 6,
            "is_closed": true
        },
        {
            "color": "#75507b",
            "name": "Postponed",
            "slug": "postponed",
            "order": 7,
            "is_closed": false
        }
    ],
    "issue_types": [
        {
            "color": "#cc0000",
            "name": "Bug",
            "order": 1
        },
        {
            "color": "#729fcf",
            "name": "Question",
            "order": 2
        },
        {
            "color": "#4e9a06",
            "name": "Enhancement",
            "order": 3
        }
    ],
    "priorities": [
        {
            "color": "#999999",
            "name": "Low",
            "order": 1
        },
```

```
        {
            "color": "#4e9a06",
            "name": "Normal",
            "order": 3
        },
        {
            "color": "#CC0000",
            "name": "High",
            "order": 5
        }
    ],
    "severities": [
        {
            "color": "#999999",
            "name": "Wishlist",
            "order": 1
        },
        {
            "color": "#729fcf",
            "name": "Minor",
            "order": 2
        },
        {
            "color": "#4e9a06",
            "name": "Normal",
            "order": 3
        },
        {
            "color": "#f57900",
            "name": "Important",
            "order": 4
        },
        {
            "color": "#CC0000",
            "name": "Critical",
            "order": 5
        }
    ],
    "roles": [
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "add_milestone", "modify_milestone", "delete_milestone",
                "view_milestones", "view_project", "add_task", "modify_task",
                "delete_task", "view_tasks", "add_us", "modify_us",
                "delete_us", "view_us", "add_wiki_page", "modify_wiki_page",
                "delete_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
```

```
            ],
            "order": 10,
            "computable": true,
            "slug": "ux",
            "name": "UX"
        },
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "add_milestone", "modify_milestone", "delete_milestone",
                "view_milestones", "view_project", "add_task", "modify_task",
                "delete_task", "view_tasks", "add_us", "modify_us",
                "delete_us", "view_us", "add_wiki_page", "modify_wiki_page",
                "delete_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
            ],
            "order": 20,
            "computable": true,
            "slug": "design",
            "name": "Design"
        },
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "add_milestone", "modify_milestone", "delete_milestone",
                "view_milestones", "view_project", "add_task", "modify_task",
                "delete_task", "view_tasks", "add_us", "modify_us",
                "delete_us", "view_us", "add_wiki_page", "modify_wiki_page",
                "delete_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
            ],
            "order": 30,
            "computable": true,
            "slug": "front",
            "name": "Front"
        },
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "add_milestone", "modify_milestone", "delete_milestone",
                "view_milestones", "view_project", "add_task", "modify_task",
                "delete_task", "view_tasks", "add_us", "modify_us",
                "delete_us", "view_us", "add_wiki_page", "modify_wiki_page",
                "delete_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
            ],
            "order": 40,
            "computable": true,
```

```
            "slug": "back",
            "name": "Back"
        },
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "add_milestone", "modify_milestone", "delete_milestone",
                "view_milestones", "view_project", "add_task", "modify_task",
                "delete_task", "view_tasks", "add_us", "modify_us",
                "delete_us", "view_us", "add_wiki_page", "modify_wiki_page",
                "delete_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
            ],
            "order": 50,
            "computable": false,
            "slug": "product-owner",
            "name": "Product Owner"
        },
        {
            "permissions": [
                "add_issue", "modify_issue", "delete_issue", "view_issues",
                "view_milestones", "view_project", "view_tasks", "view_us",
                "modify_wiki_page", "view_wiki_pages", "add_wiki_link",
                "delete_wiki_link", "view_wiki_links"
            ],
            "order": 60,
            "computable": false,
            "slug": "stakeholder",
            "name": "Stakeholder"
        }
    ],
    "id": 2,
    "name": "Kanban",
    "slug": "kanban",
    "description": "Sample description",
    "created_date": "2014-04-22T14:50:19+0000",
    "modified_date": "2014-07-25T13:11:42+0000",
    "default_owner_role": "product-owner",
    "is_backlog_activated": false,
    "is_kanban_activated": true,
    "is_wiki_activated": false,
    "is_issues_activated": false,
    "videoconferences": null,
    "videoconferences_extra_data": ""
}
```

40.1. Project list entry


```
{
    "anon_permissions": [],
    "created_date": "2014-09-16T15:39:49+0000",
    "creation_template": 1,
    "default_issue_status": 574,
    "default_issue_type": 127,
    "default_points": 977,
    "default_priority": 245,
    "default_severity": 408,
    "default_task_status": 411,
    "default_us_status": 352,
    "description": "Taiga",
    "i_am_owner": true,
    "id": 87,
    "is_backlog_activated": false,
    "is_issues_activated": true,
    "is_kanban_activated": true,
    "is_private": false,
    "is_liked": true,
    "is_watched": true,
    "is_wiki_activated": true,
    "members": [
        2,
        120,
        5,
        8766,
        16,
        121,
        8971
    ],
    "modified_date": "2014-10-29T07:35:38+0000",
    "my_permissions": [
        "admin_project_values",
        "view_tasks",
        "view_milestones",
        "view_project",
        "delete_us",
        "modify_project",
        "remove_member",
        "vote_issues",
        "add_wiki_link",
        "add_issue",
        "add_task",
        "delete_wiki_page",
```

```
            "delete_project",
            "add_us",
            "view_wiki_pages",
            "delete_task",
            "delete_wiki_link",
            "view_wiki_links",
            "modify_issue",
            "view_issues",
            "modify_wiki_link",
            "add_wiki_page",
            "delete_milestone",
            "modify_us",
            "modify_wiki_page",
            "admin_roles",
            "delete_issue",
            "add_milestone",
            "modify_task",
            "add_member",
            "modify_milestone",
            "view_us"
        ],
        "name": "AIL",
        "owner": 2,
        "public_permissions": [],
        "slug": "ail",
        "likes": 3,
        "tags": null,
        "tags_colors": {
            "api": "#ce5c00",
            "cuidador": "#204a87",
            "gestor": "#73d216",
            "health": "#a40000",
            "notes": "#888a85",
            "tags": "#edd400"
        },
        "total_milestones": 3,
        "total_story_points": 20.0,
        "videoconferences": "appear-in",
        "videoconferences_salt": null,
        "voters": 1,
        "watchers": [
          7
        ]
    }
```

40.2. Project detail


```
{
    "is_fan": true,
    "total_fans": 2,
    "is_watcher": true,
    "total_watchers": 12,
    "tags": [],
    "anon_permissions": [
        "view_project",
        "view_milestones",
        "view_us",
        "view_tasks",
        "view_issues",
        "view_wiki_pages",
        "view_wiki_links"
    ],
    "public_permissions": [
        "view_project",
        "view_milestones",
        "view_us",
        "view_tasks",
        "view_issues",
        "view_wiki_pages",
        "view_wiki_links"
    ],
    "my_permissions": [
        "add_comments_to_issue",
        "view_wiki_pages",
        "admin_roles",
        "add_wiki_page",
        "delete_project",
        "add_us",
        "delete_task",
        "add_milestone",
        "delete_issue",
        "admin_project_values",
        "modify_wiki_page",
        "add_comments_to_task",
        "modify_us",
        "view_tasks",
        "modify_task",
        "add_task",
        "modify_milestone",
        "add_member",
        "remove_member",
```

```
            "delete_wiki_link",
            "add_wiki_link",
            "modify_project",
            "delete_wiki_page",
            "view_issues",
            "view_us",
            "modify_issue",
            "view_project",
            "add_comments_to_us",
            "add_issue",
            "request_membership",
            "view_milestones",
            "delete_us",
            "view_wiki_links",
            "add_us_to_project",
            "modify_wiki_link",
            "delete_milestone"
        ],
        "i_am_owner": true,
        "tags_colors": {
            "laborum": "#67eac4",
            "cumque": "#ad75ec",
            "labore": "#6fdf52",
            "error": "#11f957",
            "fugiat": "#1c563a",
            "animi": "#d93411",
            "dolorem": "#604860",
            "ea": "#2c80b2",
            "et": "#a5bc1d",
            "expedita": "#c4a000",
            "corrupti": "#432493",
            "veniam": "#a40000",
            "sunt": "#98f4c9",
            "velit": "#790ea4",
            "unde": "#da2470",
            "doloremque": "#61405d",
            "voluptates": "#6639aa",
            "pariatur": "#7b0e4e",
            "ex": "#e06613",
            "quo": "#857670",
            "repudiandae": "#3a2b71",
            "consequuntur": "#ce24ec",
            "dolores": "#7fea8e",
            "eum": "#ee6c40",
            "earum": "#24bec9",
            "sequi": "#9f6274",
            "magnam": "#d1fac1",
            "nostrum": "#0cf81b",
```

```
        "esse": "#d77661",
        "est": "#75507b",
        "voluptatibus": "#681ad4",
        "quaerat": "#0b4425",
        "similique": "#710c97",
        "suscipit": "#38abf3",
        "quasi": "#5dae16",
        "voluptas": "#729359",
        "corporis": "#ed9c91",
        "quis": "#ef2929",
        "impedit": "#cde1f0",
        "possimus": "#fccc1b",
        "assumenda": "#52b91a",
        "ipsum": "#da3ba4",
        "provident": "#7fdcf2",
        "odio": "#edb520",
        "sit": "#abdcde",
        "quae": "#d91a8b",
        "numquam": "#fa8ea8",
        "perferendis": "#73d216",
        "repellat": "#807389",
        "dolorum": "#db7ec2",
        "beatae": "#b844bd",
        "eveniet": "#5d26b5",
        "ut": "#e74669",
        "quam": "#0149d1",
        "eligendi": "#5d8273",
        "libero": "#ad7fa8",
        "temporibus": "#8ae234",
        "facere": "#5c3566",
        "laudantium": "#9e3f1f",
        "non": "#37031f",
        "ullam": "#98ad13",
        "commodi": "#3b70df",
        "illum": "#898c66",
        "soluta": "#1398ab",
        "veritatis": "#d3d7cf",
        "ab": "#da2361",
        "saepe": "#b87b67",
        "sint": "#3b2404",
        "eius": "#860b86",
        "ipsam": "#fa74af",
        "amet": "#db04fb",
        "nesciunt": "#4c8404",
        "rerum": "#b1c629",
        "quibusdam": "#c49ac2",
        "enim": "#150d4a",
        "nemo": "#f57900",
```

```
        "iusto": "#3a10e8",
        "illo": "#3531fd",
        "eaque": "#3e7c66",
        "autem": "#5e8c91",
        "blanditiis": "#fce94f",
        "deserunt": "#e7b695",
        "maiores": "#cbb2b3",
        "eos": "#8a6433",
        "placeat": "#cc0000",
        "odit": "#e2b537",
        "nulla": "#894727",
        "aperiam": "#a2b100",
        "quisquam": "#ebca0b",
        "excepturi": "#5c3c96",
        "fuga": "#e86797",
        "dicta": "#939b44",
        "nam": "#729fcf",
        "consectetur": "#97176f",
        "reiciendis": "#2e3436",
        "inventore": "#3465a4",
        "facilis": "#0f6b6b",
        "qui": "#61f611",
        "sapiente": "#850c56",
        "itaque": "#edd400",
        "hic": "#f75f0b",
        "natus": "#e610c1",
        "id": "#87ea5d",
        "asperiores": "#a69134",
        "officiis": "#204a87",
        "cupiditate": "#144bba",
        "voluptatem": "#00d60c",
        "rem": "#688119",
        "accusantium": "#b36f86",
        "perspiciatis": "#afb825",
        "aliquam": "#631249",
        "recusandae": "#47e087",
        "a": "#86f7e4",
        "harum": "#b42d3c",
        "obcaecati": "#9ccd46",
        "aut": "#9ae4e4",
        "accusamus": "#801cf7",
        "voluptatum": "#02d22f",
        "necessitatibus": "#84e3b6",
        "nihil": "#98a352",
        "fugit": "#9345df",
        "dolor": "#641bd9",
        "explicabo": "#2892cb",
        "ducimus": "#ea6bb9",
```

```
        "consequatur": "#3ad7db",
        "neque": "#888a85",
        "modi": "#494e30",
        "adipisci": "#257dec",
        "magni": "#429e6f",
        "cum": "#ab14d9",
        "quidem": "#ae6519",
        "maxime": "#1acc29",
        "porro": "#ce5c00",
        "exercitationem": "#ac7c74",
        "culpa": "#f5e53b",
        "reprehenderit": "#6c82c6",
        "laboriosam": "#fcaf3e",
        "at": "#27e90d",
        "dolore": "#61b076",
        "praesentium": "#0cd131",
        "architecto": "#9d1e93",
        "vel": "#4e9a06",
        "sed": "#c15b7b",
        "quas": "#6e3390",
        "iure": "#019320"
    },
    "total_closed_milestones": 0,
    "notify_level": 1,
    "us_statuses": [
        {
            "id": 1,
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false,
            "is_archived": false,
            "color": "#999999",
            "wip_limit": null,
            "project": 1
        },
        {
            "id": 2,
            "name": "Ready",
            "slug": "ready",
            "order": 2,
            "is_closed": false,
            "is_archived": false,
            "color": "#ff8a84",
            "wip_limit": null,
            "project": 1
        },
        {
```

```
            "id": 3,
            "name": "In progress",
            "slug": "in-progress",
            "order": 3,
            "is_closed": false,
            "is_archived": false,
            "color": "#ff9900",
            "wip_limit": null,
            "project": 1
        },
        {
            "id": 4,
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 4,
            "is_closed": false,
            "is_archived": false,
            "color": "#fcc000",
            "wip_limit": null,
            "project": 1
        },
        {
            "id": 5,
            "name": "Done",
            "slug": "done",
            "order": 5,
            "is_closed": true,
            "is_archived": false,
            "color": "#669900",
            "wip_limit": null,
            "project": 1
        },
        {
            "id": 6,
            "name": "Archived",
            "slug": "archived",
            "order": 6,
            "is_closed": true,
            "is_archived": true,
            "color": "#5c3566",
            "wip_limit": null,
            "project": 1
        }
    ],
    "points": [
        {
            "id": 1,
            "name": "?",
```

```
            "order": 1,
            "value": null,
            "project": 1
        },
        {
            "id": 2,
            "name": "0",
            "order": 2,
            "value": 0.0,
            "project": 1
        },
        {
            "id": 3,
            "name": "1/2",
            "order": 3,
            "value": 0.5,
            "project": 1
        },
        {
            "id": 4,
            "name": "1",
            "order": 4,
            "value": 1.0,
            "project": 1
        },
        {
            "id": 5,
            "name": "2",
            "order": 5,
            "value": 2.0,
            "project": 1
        },
        {
            "id": 6,
            "name": "3",
            "order": 6,
            "value": 3.0,
            "project": 1
        },
        {
            "id": 7,
            "name": "5",
            "order": 7,
            "value": 5.0,
            "project": 1
        },
        {
            "id": 8,
```

```
            "name": "8",
            "order": 8,
            "value": 8.0,
            "project": 1
        },
        {
            "id": 9,
            "name": "10",
            "order": 9,
            "value": 10.0,
            "project": 1
        },
        {
            "id": 10,
            "name": "13",
            "order": 10,
            "value": 13.0,
            "project": 1
        },
        {
            "id": 11,
            "name": "20",
            "order": 11,
            "value": 20.0,
            "project": 1
        },
        {
            "id": 12,
            "name": "40",
            "order": 12,
            "value": 40.0,
            "project": 1
        }
    ],
    "task_statuses": [
        {
            "id": 1,
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false,
            "color": "#999999",
            "project": 1
        },
        {
            "id": 2,
            "name": "In progress",
            "slug": "in-progress",
```

```
            "order": 2,
            "is_closed": false,
            "color": "#ff9900",
            "project": 1
        },
        {
            "id": 3,
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 3,
            "is_closed": true,
            "color": "#ffcc00",
            "project": 1
        },
        {
            "id": 4,
            "name": "Closed",
            "slug": "closed",
            "order": 4,
            "is_closed": true,
            "color": "#669900",
            "project": 1
        },
        {
            "id": 5,
            "name": "Needs Info",
            "slug": "needs-info",
            "order": 5,
            "is_closed": false,
            "color": "#999999",
            "project": 1
        }
    ],
    "issue_statuses": [
        {
            "id": 1,
            "name": "New",
            "slug": "new",
            "order": 1,
            "is_closed": false,
            "color": "#8C2318",
            "project": 1
        },
        {
            "id": 2,
            "name": "In progress",
            "slug": "in-progress",
            "order": 2,
```

```
            "is_closed": false,
            "color": "#5E8C6A",
            "project": 1
        },
        {
            "id": 3,
            "name": "Ready for test",
            "slug": "ready-for-test",
            "order": 3,
            "is_closed": true,
            "color": "#88A65E",
            "project": 1
        },
        {
            "id": 4,
            "name": "Closed",
            "slug": "closed",
            "order": 4,
            "is_closed": true,
            "color": "#BFB35A",
            "project": 1
        },
        {
            "id": 5,
            "name": "Needs Info",
            "slug": "needs-info",
            "order": 5,
            "is_closed": false,
            "color": "#89BAB4",
            "project": 1
        },
        {
            "id": 6,
            "name": "Rejected",
            "slug": "rejected",
            "order": 6,
            "is_closed": true,
            "color": "#CC0000",
            "project": 1
        },
        {
            "id": 7,
            "name": "Postponed",
            "slug": "postponed",
            "order": 7,
            "is_closed": false,
            "color": "#666666",
            "project": 1
```

```
        }
    ],
    "issue_types": [
        {
            "id": 1,
            "name": "Bug",
            "order": 1,
            "color": "#89BAB4",
            "project": 1
        },
        {
            "id": 2,
            "name": "Question",
            "order": 2,
            "color": "#ba89a8",
            "project": 1
        },
        {
            "id": 3,
            "name": "Enhancement",
            "order": 3,
            "color": "#89a8ba",
            "project": 1
        }
    ],
    "priorities": [
        {
            "id": 1,
            "name": "Low",
            "order": 1,
            "color": "#666666",
            "project": 1
        },
        {
            "id": 2,
            "name": "Normal",
            "order": 3,
            "color": "#669933",
            "project": 1
        },
        {
            "id": 3,
            "name": "High",
            "order": 5,
            "color": "#CC0000",
            "project": 1
        }
    ],
```

```
    "severities": [
        {
            "id": 1,
            "name": "Wishlist",
            "order": 1,
            "color": "#666666",
            "project": 1
        },
        {
            "id": 2,
            "name": "Minor",
            "order": 2,
            "color": "#669933",
            "project": 1
        },
        {
            "id": 3,
            "name": "Normal",
            "order": 3,
            "color": "#0000FF",
            "project": 1
        },
        {
            "id": 4,
            "name": "Important",
            "order": 4,
            "color": "#FFA500",
            "project": 1
        },
        {
            "id": 5,
            "name": "Critical",
            "order": 5,
            "color": "#CC0000",
            "project": 1
        }
    ],
    "userstory_custom_attributes": [
        {
            "id": 1,
            "name": "assumenda laboriosam maxime",
            "description": "nihil pariatur in nisi asperiores soluta unde similique
deserunt consequatur ipsa dolorum",
            "field_type": "TEXT",
            "order": 1,
            "project": 1
        },
        {
```

```
            "id": 2,
            "name": "officia dolorum possimus",
            "description": "eum nesciunt consectetur similique nihil velit rerum error
magnam nostrum quisquam dolore",
            "field_type": "TEXT",
            "order": 2,
            "project": 1
        },
        {
            "id": 3,
            "name": "vero facere soluta",
            "description": "quod vel dolorum obcaecati cupiditate",
            "field_type": "TEXT",
            "order": 3,
            "project": 1
        }
    ],
    "task_custom_attributes": [
        {
            "id": 1,
            "name": "vel",
            "description": "quo nisi ratione rem sit eum fugiat veniam porro deserunt
optio voluptas",
            "field_type": "TEXT",
            "order": 1,
            "project": 1
        },
        {
            "id": 2,
            "name": "libero quibusdam",
            "description": "alias rerum dolorum quas quod maiores iure error
perferendis",
            "field_type": "TEXT",
            "order": 2,
            "project": 1
        },
        {
            "id": 3,
            "name": "ipsa illo",
            "description": "magnam pariatur cumque",
            "field_type": "TEXT",
            "order": 3,
            "project": 1
        }
    ],
    "issue_custom_attributes": [
        {
            "id": 1,
```

```
            "name": "aut accusantium cumque",
            "description": "blanditiis dolores commodi",
            "field_type": "TEXT",
            "order": 1,
            "project": 1
        },
        {
            "id": 2,
            "name": "molestias ducimus cumque",
            "description": "deleniti iste nostrum",
            "field_type": "TEXT",
            "order": 2,
            "project": 1
        },
        {
            "id": 3,
            "name": "quasi eius",
            "description": "quaerat nulla minima eius officia saepe",
            "field_type": "TEXT",
            "order": 3,
            "project": 1
        }
    ],
    "roles": [
        {
            "id": 1,
            "name": "UX",
            "slug": "ux",
            "order": 10,
            "computable": true
        },
        {
            "id": 2,
            "name": "Design",
            "slug": "design",
            "order": 20,
            "computable": true
        },
        {
            "id": 3,
            "name": "Front",
            "slug": "front",
            "order": 30,
            "computable": true
        },
        {
            "id": 4,
            "name": "Back",
```

```
            "slug": "back",
            "order": 40,
            "computable": true
        },
        {
            "id": 5,
            "name": "Product Owner",
            "slug": "product-owner",
            "order": 50,
            "computable": false
        },
        {
            "id": 6,
            "name": "Stakeholder",
            "slug": "stakeholder",
            "order": 60,
            "computable": false
        }
    ],
    "members": [
        {
            "id": 5,
            "username": "admin",
            "full_name": "Administrator",
            "full_name_display": "Administrator",
            "color": "",
            "photo":
"//www.gravatar.com/avatar/64e1b8d34f425d19e1ee2ea7236d3028?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Design",
            "user": 5,
            "role": 2,
            "is_owner": false
        },
        {
            "id": 11,
            "username": "user5",
            "full_name": "Alicia Flores",
            "full_name_display": "Alicia Flores",
            "color": "#D70A53",
            "photo":
"//www.gravatar.com/avatar/c9ba9d485f9a9153ebf53758feb0980c?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Back",
            "user": 11,
            "role": 4,
```

```
            "is_owner": true
        },
        {
            "id": 9,
            "username": "user3",
            "full_name": "Concepcion Garrido",
            "full_name_display": "Concepcion Garrido",
            "color": "#D70A53",
            "photo":
"//www.gravatar.com/avatar/9971a763f5dfc5cbd1ce1d2865b4fcfa?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "UX",
            "user": 9,
            "role": 1,
            "is_owner": true
        },
        {
            "id": 10,
            "username": "user4",
            "full_name": "Esther Ferrer",
            "full_name_display": "Esther Ferrer",
            "color": "#40826D",
            "photo":
"//www.gravatar.com/avatar/f31e0063c7cd6da19b6467bc48d2b14b?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Front",
            "user": 10,
            "role": 3,
            "is_owner": true
        },
        {
            "id": 15,
            "username": "user9",
            "full_name": "Francisca Gallardo",
            "full_name_display": "Francisca Gallardo",
            "color": "#67CF00",
            "photo":
"//www.gravatar.com/avatar/69b60d39a450e863609ae3546b12b360?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "UX",
            "user": 15,
            "role": 1,
            "is_owner": true
        },
        {
```

```
            "id": 13,
            "username": "user7",
            "full_name": "German Benitez",
            "full_name_display": "German Benitez",
            "color": "#4B0082",
            "photo":
"//www.gravatar.com/avatar/6d7e702bd6c6fc568fca7577f9ca8c55?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Product Owner",
            "user": 13,
            "role": 5,
            "is_owner": true
        },
        {
            "id": 12,
            "username": "user6",
            "full_name": "Julian Hidalgo",
            "full_name_display": "Julian Hidalgo",
            "color": "#67CF00",
            "photo":
"//www.gravatar.com/avatar/74cb769a5e64d445b8550789e1553502?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Product Owner",
            "user": 12,
            "role": 5,
            "is_owner": true
        },
        {
            "id": 14,
            "username": "user8",
            "full_name": "Lourdes Aguilar",
            "full_name_display": "Lourdes Aguilar",
            "color": "#CC0000",
            "photo":
"//www.gravatar.com/avatar/dce0e8ed702cd85d5132e523121e619b?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Design",
            "user": 14,
            "role": 2,
            "is_owner": false
        },
        {
            "id": 7,
            "username": "user1",
            "full_name": "Purificacion Montero",
```

```
            "full_name_display": "Purificacion Montero",
            "color": "#67CF00",
            "photo":
"//www.gravatar.com/avatar/aed1e43be0f69f07ce6f34a907bc6328?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "UX",
            "user": 7,
            "role": 1,
            "is_owner": true
        },
        {
            "id": 6,
            "username": "user6532909695705815086",
            "full_name": "Silvia Soto",
            "full_name_display": "Silvia Soto",
            "color": "#FFFF00",
            "photo":
"//www.gravatar.com/avatar/ece2f7a2dec5f21b2858fecabdcacacc?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Design",
            "user": 6,
            "role": 2,
            "is_owner": false
        },
        {
            "id": 8,
            "username": "user2",
            "full_name": "Sofia Ramirez",
            "full_name_display": "Sofia Ramirez",
            "color": "#4B0082",
            "photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "is_active": true,
            "role_name": "Product Owner",
            "user": 8,
            "role": 5,
            "is_owner": true
        }
    ],
    "id": 1,
    "default_points": 1,
    "default_us_status": 1,
    "default_task_status": 1,
    "default_priority": 2,
    "default_severity": 3,
```

```
    "default_issue_status": 1,
    "default_issue_type": 1,
    "name": "Project Example 0",
    "slug": "project-0",
    "description": "Project example 0 description",
    "created_date": "2015-10-01T09:52:54+0000",
    "modified_date": "2015-10-01T09:53:13+0000",
    "owner": 13,
    "total_milestones": 8,
    "total_story_points": 622.0,
    "is_backlog_activated": true,
    "is_kanban_activated": true,
    "is_wiki_activated": true,
    "is_issues_activated": true,
    "videoconferences": null,
    "videoconferences_extra_data": null,
    "creation_template": 1,
    "is_private": false,
    "userstories_csv_uuid": null,
    "tasks_csv_uuid": null,
    "issues_csv_uuid": null
}
```


40.3. Project modules configuration


```
{
    "github": {
        "secret": "b7a95570051a49ad8fb9482b5ef40262",
        "webhooks_url": "https://api.taiga.io/api/v1/github-hook?project=3"
    }
}
```


40.4. Project stats detail


```
{
    "assigned_points": 331.5,
    "assigned_points_per_role": {
        "43": 3.0,
        "44": 10.5,
        "45": 140.0,
        "46": 178.0
    },
    "closed_points": 310.5,
    "closed_points_per_role": {
```

```
        "43": 3.0,
        "44": 7.5,
        "45": 133.0,
        "46": 167.0
    },
    "defined_points": 342.5,
    "defined_points_per_role": {
        "43": 3.5,
        "44": 12.0,
        "45": 146.0,
        "46": 181.0
    },
    "milestones": [
        {
            "client-increment": 0,
            "evolution": 400.0,
            "name": "Sprint 0",
            "optimal": 400.0,
            "team-increment": 0
        },
        {
            "client-increment": 0,
            "evolution": 400.0,
            "name": "Sprint 1",
            "optimal": 380.0,
            "team-increment": 0
        },
        {
            "client-increment": 0,
            "evolution": 388.5,
            "name": "Sprint 2",
            "optimal": 360.0,
            "team-increment": 0
        },
        {
            "client-increment": 0,
            "evolution": 373.5,
            "name": "Sprint 3",
            "optimal": 340.0,
            "team-increment": 0.5
        },
        {
            "client-increment": 0,
            "evolution": 361.5,
            "name": "Sprint 4",
            "optimal": 320.0,
            "team-increment": 0.5
        },
```

```
        {
            "client-increment": 0,
            "evolution": 357.0,
            "name": "Sprint 5",
            "optimal": 300.0,
            "team-increment": 3.5
        },
        {
            "client-increment": 0,
            "evolution": 351.5,
            "name": "Sprint 6",
            "optimal": 280.0,
            "team-increment": 5.0
        },
        {
            "client-increment": 0,
            "evolution": 347.0,
            "name": "Sprint 7",
            "optimal": 260.0,
            "team-increment": 5.0
        },
        {
            "client-increment": 0,
            "evolution": 346.5,
            "name": "Sprint 8",
            "optimal": 240.0,
            "team-increment": 5.0
        },
        {
            "client-increment": 0,
            "evolution": 325.5,
            "name": "Sprint 9",
            "optimal": 220.0,
            "team-increment": 6.0
        },
        {
            "client-increment": 0,
            "evolution": 308.0,
            "name": "Sprint 10 (new-taiga)",
            "optimal": 200.0,
            "team-increment": 7.0
        },
        {
            "client-increment": 0,
            "evolution": 199.5,
            "name": "Sprint 11 (new-taiga)",
            "optimal": 180.0,
            "team-increment": 7.0
```

```
        },
        {
            "client-increment": 0,
            "evolution": 192.5,
            "name": "Sprint 13 (new-taiga)",
            "optimal": 160.0,
            "team-increment": 7.0
        },
        {
            "client-increment": 0,
            "evolution": 157.0,
            "name": "Sprint 14",
            "optimal": 140.0,
            "team-increment": 8.0
        },
        {
            "client-increment": 0,
            "evolution": 141.0,
            "name": "Sprint 15",
            "optimal": 120.0,
            "team-increment": 8.0
        },
        {
            "client-increment": 0,
            "evolution": 108.5,
            "name": "Sprint 16",
            "optimal": 100.0,
            "team-increment": 8.0
        },
        {
            "client-increment": [
                0
            ],
            "evolution": 89.5,
            "name": "Future sprint",
            "optimal": 80.0,
            "team-increment": [
                8.0
            ]
        },
        {
            "client-increment": [
                0
            ],
            "evolution": null,
            "name": "Future sprint",
            "optimal": 60.0,
            "team-increment": [
```

```
                8.0
            ]
        },
        {
            "client-increment": [
                0
            ],
            "evolution": null,
            "name": "Future sprint",
            "optimal": 40.0,
            "team-increment": [
                8.0
            ]
        },
        {
            "client-increment": [
                0
            ],
            "evolution": null,
            "name": "Future sprint",
            "optimal": 20.0,
            "team-increment": [
                8.0
            ]
        },
        {
            "client-increment": [
                0
            ],
            "evolution": null,
            "name": "Project End",
            "optimal": 0.0,
            "team-increment": [
                8.0
            ]
        }
    ],
    "name": "Taiga",
    "speed": 20.7,
    "total_milestones": 20,
    "total_points": 400.0
}
```


40.5. Project issue stats detail


```
{
```

```
    "closed_issues": 560,
    "issues_per_assigned_to": {
        "0": {
            "color": "black",
            "count": 185,
            "id": 0,
            "name": "Unassigned",
            "username": "Unassigned"
        },
        "111": {
            "color": "#6427c0",
            "count": 2,
            "id": 111,
            "name": "Juan de la Cruz Garcia",
            "username": "juan.delacruz"
        },
        "13": {
            "color": "#CC0000",
            "count": 100,
            "id": 13,
            "name": "Xavier Juli\u00e1n",
            "username": "xavier.julian"
        },
        "2": {
            "color": "#007000",
            "count": 93,
            "id": 2,
            "name": "Jes\u00fas Espino",
            "username": "jesus.espino"
        },
        "24": {
            "color": "#d9ff2f",
            "count": 3,
            "id": 24,
            "name": "Enrique Posner",
            "username": "eposner"
        },
        "38": {
            "color": "#FF9900",
            "count": 8,
            "id": 38,
            "name": "Mario Garcia",
            "username": "mario.garcia"
        },
        "5": {
            "color": "#761CEC",
            "count": 3,
            "id": 5,
```

```
            "name": "Pablo Ruiz M\u00fazquiz",
            "username": "pablo.ruiz"
        },
        "6": {
            "color": "#40826D",
            "count": 85,
            "id": 6,
            "name": "Juanfran",
            "username": "juanfran.alcantara"
        },
        "7": {
            "color": "#FC8EAC",
            "count": 76,
            "id": 7,
            "name": "Alejandro Alonso",
            "username": "alejandro.alonso"
        },
        "8": {
            "color": "#A5694F",
            "count": 16,
            "id": 8,
            "name": "Andrey Antukh",
            "username": "andrei.antoukh"
        },
        "9": {
            "color": "#FFF8E7",
            "count": 97,
            "id": 9,
            "name": "David Barrag\u00e1n Merino",
            "username": "david.barragan"
        }
    },
    "issues_per_owner": {
        "11": {
            "color": "#8f0030",
            "count": 1,
            "id": 11,
            "name": "Anler Hern\u00e1ndez Peral",
            "username": "anler.hernandez"
        },
        "111": {
            "color": "#6427c0",
            "count": 1,
            "id": 111,
            "name": "Juan de la Cruz Garcia",
            "username": "juan.delacruz"
        },
        "13": {
```

```
            "color": "#CC0000",
            "count": 64,
            "id": 13,
            "name": "Xavier Juli\u00e1n",
            "username": "xavier.julian"
        },
        "17": {
            "color": "#FFFF00",
            "count": 6,
            "id": 17,
            "name": "Alonso Torres",
            "username": "alonso.torres"
        },
        "18": {
            "color": "#C0FF33",
            "count": 14,
            "id": 18,
            "name": "Miguel de la Cruz",
            "username": "miguel.delacruz"
        },
        "19": {
            "color": "#67CF00",
            "count": 7,
            "id": 19,
            "name": "Antonio de la Torre",
            "username": "antonio.delatorre"
        },
        "2": {
            "color": "#007000",
            "count": 50,
            "id": 2,
            "name": "Jes\u00fas Espino",
            "username": "jesus.espino"
        },
        "24": {
            "color": "#d9ff2f",
            "count": 135,
            "id": 24,
            "name": "Enrique Posner",
            "username": "eposner"
        },
        "3": {
            "color": "#4B0082",
            "count": 80,
            "id": 3,
            "name": "Iv\u00e1n L\u00f3pez",
            "username": "ivan.lopez"
        },
```

```
        "38": {
            "color": "#FF9900",
            "count": 6,
            "id": 38,
            "name": "Mario Garcia",
            "username": "mario.garcia"
        },
        "4": {
            "color": "#708090",
            "count": 13,
            "id": 4,
            "name": "Pablo Alba",
            "username": "pablo.alba"
        },
        "5": {
            "color": "#761CEC",
            "count": 16,
            "id": 5,
            "name": "Pablo Ruiz M\u00fazquiz",
            "username": "pablo.ruiz"
        },
        "6": {
            "color": "#40826D",
            "count": 20,
            "id": 6,
            "name": "Juanfran",
            "username": "juanfran.alcantara"
        },
        "7": {
            "color": "#FC8EAC",
            "count": 106,
            "id": 7,
            "name": "Alejandro Alonso",
            "username": "alejandro.alonso"
        },
        "8": {
            "color": "#A5694F",
            "count": 11,
            "id": 8,
            "name": "Andrey Antukh",
            "username": "andrei.antoukh"
        },
        "9": {
            "color": "#FFF8E7",
            "count": 138,
            "id": 9,
            "name": "David Barrag\u00e1n Merino",
            "username": "david.barragan"
```

```
        }
    },
    "issues_per_priority": {
        "1": {
            "color": "#888a85",
            "count": 50,
            "id": 1,
            "name": "Low"
        },
        "2": {
            "color": "#4e9a06",
            "count": 397,
            "id": 2,
            "name": "Normal"
        },
        "3": {
            "color": "#a40000",
            "count": 221,
            "id": 3,
            "name": "High"
        }
    },
    "issues_per_severity": {
        "1": {
            "color": "#888a85",
            "count": 9,
            "id": 1,
            "name": "Wishlist"
        },
        "2": {
            "color": "#4e9a06",
            "count": 21,
            "id": 2,
            "name": "Minor"
        },
        "3": {
            "color": "#204a87",
            "count": 464,
            "id": 3,
            "name": "Normal"
        },
        "4": {
            "color": "#ce5c00",
            "count": 118,
            "id": 4,
            "name": "Important"
        },
        "5": {
```

```
            "color": "#a40000",
            "count": 56,
            "id": 5,
            "name": "Critical"
        }
    },
    "issues_per_status": {
        "1": {
            "color": "#8C2318",
            "count": 90,
            "id": 1,
            "name": "New"
        },
        "2": {
            "color": "#5E8C6A",
            "count": 7,
            "id": 2,
            "name": "In progress"
        },
        "3": {
            "color": "#88A65E",
            "count": 370,
            "id": 3,
            "name": "Ready for test"
        },
        "4": {
            "color": "#BFB35A",
            "count": 89,
            "id": 4,
            "name": "Closed"
        },
        "5": {
            "color": "#89BAB4",
            "count": 2,
            "id": 5,
            "name": "Needs Info"
        },
        "6": {
            "color": "#CC0000",
            "count": 101,
            "id": 6,
            "name": "Rejected"
        },
        "7": {
            "color": "#666666",
            "count": 9,
            "id": 7,
            "name": "Postponed"
```

```
        }
    },
    "issues_per_type": {
        "1": {
            "color": "#f57900",
            "count": 501,
            "id": 1,
            "name": "Bug"
        },
        "6": {
            "color": "#4e9a06",
            "count": 167,
            "id": 6,
            "name": "Enhancement"
        }
    },
    "last_four_weeks_days": {
        "by_open_closed": {
            "closed": [
                8,
                3,
                2,
                0,
                0,
                4,
                5,
                2,
                3,
                3,
                0,
                0,
                2,
                4,
                4,
                10,
                6,
                0,
                0,
                12,
                0,
                4,
                5,
                5,
                0,
                0,
                0,
                0
            ],
```

```
            "open": [
                7,
                3,
                2,
                0,
                0,
                4,
                9,
                8,
                1,
                1,
                3,
                0,
                5,
                4,
                5,
                17,
                4,
                6,
                0,
                9,
                6,
                10,
                9,
                10,
                0,
                0,
                0,
                0
            ]
        },
        "by_priority": {
            "1": {
                "color": "#888a85",
                "data": [
                    18,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
                    17,
```

```
                    17,
                    18,
                    18,
                    19,
                    18,
                    18,
                    18,
                    18,
                    19,
                    22,
                    22,
                    22,
                    22,
                    22,
                    22
                ],
                "id": 1,
                "name": "Low"
            },
            "2": {
                "color": "#4e9a06",
                "data": [
                    33,
                    30,
                    28,
                    27,
                    27,
                    29,
                    28,
                    28,
                    28,
                    27,
                    27,
                    27,
                    30,
                    32,
                    32,
                    38,
                    36,
                    39,
                    39,
                    43,
                    43,
                    49,
                    49,
                    58,
                    55,
                    55,
```

```
                    55,
                    55
                ],
                "id": 2,
                "name": "Normal"
            },
            "3": {
                "color": "#a40000",
                "data": [
                    23,
                    22,
                    23,
                    22,
                    22,
                    24,
                    30,
                    33,
                    32,
                    31,
                    31,
                    31,
                    33,
                    33,
                    33,
                    40,
                    35,
                    33,
                    33,
                    38,
                    32,
                    35,
                    37,
                    33,
                    31,
                    31,
                    31,
                    31
                ],
                "id": 3,
                "name": "High"
            }
        },
        "by_severity": {
            "1": {
                "color": "#888a85",
                "data": [
                    8,
                    8,
```

```
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8,
                8
            ],
            "id": 1,
            "name": "Wishlist"
        },
        "2": {
            "color": "#4e9a06",
            "data": [
                6,
                3,
                3,
                3,
                3,
                3,
                3,
                3,
                2,
                2,
                2,
                2,
                2,
                2,
                2,
                2,
                3,
```

```
                    3,
                    3,
                    3,
                    3,
                    3,
                    3,
                    4,
                    4,
                    4,
                    4,
                    4,
                    4,
                    4
                ],
                "id": 2,
                "name": "Minor"
            },
            "3": {
                "color": "#204a87",
                "data": [
                    52,
                    51,
                    48,
                    46,
                    46,
                    48,
                    48,
                    51,
                    51,
                    52,
                    52,
                    52,
                    55,
                    56,
                    54,
                    61,
                    59,
                    63,
                    63,
                    66,
                    63,
                    70,
                    74,
                    82,
                    79,
                    79,
                    79,
                    79
```

```
            ],
            "id": 3,
            "name": "Normal"
        },
        "4": {
            "color": "#ce5c00",
            "data": [
                7,
                7,
                8,
                8,
                8,
                10,
                13,
                14,
                13,
                11,
                10,
                10,
                12,
                12,
                13,
                16,
                12,
                10,
                10,
                11,
                14,
                16,
                16,
                15,
                14,
                14,
                14,
                14
            ],
            "id": 4,
            "name": "Important"
        },
        "5": {
            "color": "#a40000",
            "data": [
                1,
                0,
                1,
                1,
                1,
                1,
```

```
                    3,
                    3,
                    3,
                    2,
                    3,
                    3,
                    3,
                    4,
                    5,
                    8,
                    8,
                    6,
                    6,
                    11,
                    5,
                    5,
                    6,
                    4,
                    3,
                    3,
                    3,
                    3
                ],
                "id": 5,
                "name": "Critical"
            }
        },
        "by_status": {}
    },
    "opened_issues": 108,
    "total_issues": 668
}
```


40.6. Project issue filters data detail


```
{
    "assigned_to": [
        [
            10,
            0
        ],
        [
            6,
            85
        ],
        [
```

```
            3,
            0
        ],
        [
            15,
            0
        ],
        [
            1,
            0
        ],
        [
            7,
            76
        ],
        [
            19,
            0
        ],
        [
            17,
            0
        ],
        [
            8,
            16
        ],
        [
            24,
            3
        ],
        [
            25,
            0
        ],
        [
            9,
            97
        ],
        [
            13,
            100
        ],
        [
            5,
            3
        ],
        [
```

```
            null,
            185
        ],
        [
            16,
            0
        ],
        [
            12623,
            0
        ],
        [
            111,
            2
        ],
        [
            38,
            8
        ],
        [
            4,
            0
        ],
        [
            18,
            0
        ],
        [
            2,
            93
        ],
        [
            11,
            0
        ],
        [
            26,
            0
        ]
    ],
    "created_by": [
        [
            12623,
            0
        ],
        [
            10,
            0
```

```
        ],
        [
            7,
            106
        ],
        [
            11,
            1
        ],
        [
            13,
            64
        ],
        [
            24,
            135
        ],
        [
            26,
            0
        ],
        [
            16,
            0
        ],
        [
            4,
            13
        ],
        [
            18,
            14
        ],
        [
            6,
            20
        ],
        [
            3,
            80
        ],
        [
            19,
            7
        ],
        [
            15,
            0
```

```
        ],
        [
            17,
            6
        ],
        [
            1,
            0
        ],
        [
            2,
            50
        ],
        [
            9,
            138
        ],
        [
            25,
            0
        ],
        [
            38,
            6
        ],
        [
            8,
            11
        ],
        [
            111,
            1
        ],
        [
            5,
            16
        ]
    ],
    "owners": [
        [
            12623,
            0
        ],
        [
            10,
            0
        ],
        [
```

```
            7,
            106
        ],
        [
            11,
            1
        ],
        [
            13,
            64
        ],
        [
            24,
            135
        ],
        [
            26,
            0
        ],
        [
            16,
            0
        ],
        [
            4,
            13
        ],
        [
            18,
            14
        ],
        [
            6,
            20
        ],
        [
            3,
            80
        ],
        [
            19,
            7
        ],
        [
            15,
            0
        ],
        [
```

```
            17,
            6
        ],
        [
            1,
            0
        ],
        [
            2,
            50
        ],
        [
            9,
            138
        ],
        [
            25,
            0
        ],
        [
            38,
            6
        ],
        [
            8,
            11
        ],
        [
            111,
            1
        ],
        [
            5,
            16
        ]
    ],
    "priorities": [
        [
            1,
            50
        ],
        [
            2,
            397
        ],
        [
            3,
            221
```

```
        ]
    ],
    "severities": [
        [
            1,
            9
        ],
        [
            2,
            21
        ],
        [
            3,
            464
        ],
        [
            4,
            118
        ],
        [
            5,
            56
        ]
    ],
    "statuses": [
        [
            1,
            90
        ],
        [
            2,
            7
        ],
        [
            3,
            370
        ],
        [
            4,
            89
        ],
        [
            5,
            2
        ],
        [
            6,
            101
```

```
        ],
        [
            7,
            9
        ]
    ],
    "tags": [
        [
            "667",
            1
        ],
        [
            "admin",
            1
        ],
        [
            "analytics",
            1
        ],
        [
            "attachment",
            1
        ],
        [
            "attachments",
            2
        ],
        [
            "back",
            35
        ],
        [
            "backlog",
            6
        ],
        [
            "block",
            1
        ],
        [
            "comments",
            1
        ],
        [
            "create",
            1
        ],
        [
```

```
            "design",
            5
        ],
        [
            "dise\u00f1o",
            14
        ],
        [
            "drag & drop",
            1
        ],
        [
            "edition",
            2
        ],
        [
            "email",
            2
        ],
        [
            "feedback",
            41
        ],
        [
            "filters",
            2
        ],
        [
            "font",
            3
        ],
        [
            "form",
            1
        ],
        [
            "forms",
            1
        ],
        [
            "front",
            72
        ],
        [
            "frontend",
            1
        ],
        [
```

```
            "historico",
            1
        ],
        [
            "home",
            1
        ],
        [
            "hover",
            1
        ],
        [
            "images",
            1
        ],
        [
            "issues",
            7
        ],
        [
            "kamban",
            1
        ],
        [
            "kan",
            1
        ],
        [
            "kanban",
            5
        ],
        [
            "landing",
            14
        ],
        [
            "layout",
            1
        ],
        [
            "lightbox",
            3
        ],
        [
            "loading",
            1
        ],
        [
```

```
            "login",
            1
        ],
        [
            "logo",
            2
        ],
        [
            "manage members",
            1
        ],
        [
            "markdown",
            2
        ],
        [
            "member",
            1
        ],
        [
            "migraci\u00f3n redmine",
            9
        ],
        [
            "notifications",
            2
        ],
        [
            "ordering",
            1
        ],
        [
            "paginacion",
            1
        ],
        [
            "performance",
            1
        ],
        [
            "points",
            1
        ],
        [
            "register",
            1
        ],
        [
```

```
            "responsive",
            1
        ],
        [
            "spinner",
            1
        ],
        [
            "sprint",
            1
        ],
        [
            "style",
            1
        ],
        [
            "svg",
            1
        ],
        [
            "tags",
            1
        ],
        [
            "taiga-web",
            1
        ],
        [
            "task",
            1
        ],
        [
            "taskboard",
            4
        ],
        [
            "tasks",
            1
        ],
        [
            "title",
            1
        ],
        [
            "us",
            3
        ],
        [
```

```
            "user-settings",
            2
        ],
        [
            "uss",
            1
        ],
        [
            "ux",
            23
        ],
        [
            "web",
            1
        ],
        [
            "wiki",
            5
        ]
    ],
    "types": [
        [
            1,
            501
        ],
        [
            6,
            167
        ]
    ]
}
```


40.7. Project tag colors data detail


```
{
    "667": "#74da61",
    "Back": "#b52b36",
    "Backlog": "#1fe3af",
    "Documentaci\u00f3n": "#60f92f",
    "Taskboard": "#066725",
    "activity": "#ce5c00",
    "admin": "#d033e2",
    "analisis": "#ac1d8e",
    "analytics": "#66d9b5",
    "api": "#a033a5",
    "appearin": "#d3d7cf",
```

```
    "attachment": "#ad7fa8",
    "attachments": "#05cb34",
    "back": "#73d216",
    "backend": "#754a08",
    "backlog": "#fce94f",
    "block": "#0214b4",
    "blog": "#169254",
    "checksley": "#9d4cd6",
    "coffeescript": "#4e9a06",
    "comments": "#5b17a6",
    "controllers": "#3465a4",
    "create": "#2e3436",
    "design": "#056f11",
    "directive": "#3d981b",
    "dise\u00f1o": "#65b784",
    "doc": "#f7f029",
    "documentaci\u00f3n": "#c4a000",
    "drag & drop": "#4557ef",
    "edition": "#194562",
    "email": "#a88b7d",
    "favicon": "#2ba3a0",
    "feedback": "#ef15f4",
    "filters": "#9ce5ff",
    "font": "#464257",
    "form": "#d00b39",
    "forms": "#453ca9",
    "front": "#a40000",
    "front-refactor": "#edd400",
    "frontend": "#9ecc84",
    "general": "#75507b",
    "graph": "#29a184",
    "graphs": "#ef2929",
    "historico": "#59f4b6",
    "history": "#66f79d",
    "home": "#e83249",
    "hover": "#fbc588",
    "i18n": "#61d28b",
    "icono": "#4b1726",
    "images": "#19f49d",
    "issue": "#f57900",
    "issues": "#fcaf3e",
    "kamban": "#9296c7",
    "kan": "#6769db",
    "kanban": "#4073e8",
    "kk": "#2ed451",
    "landing": "#b95618",
    "layout": "#207b87",
    "license": "#234571",
```

```
    "lightbox": "#97a2af",
    "loading": "#14b85f",
    "login": "#2736fa",
    "logo": "#5807dd",
    "manage members": "#4ed477",
    "markdown": "#90320b",
    "member": "#6467ba",
    "migraci\u00f3n redmine": "#eacd33",
    "model": "#1d06a0",
    "newsletter": "#755df5",
    "notifications": "#1b60d2",
    "ordering": "#428ac4",
    "orm": "#e14417",
    "paginacion": "#c17776",
    "performance": "#476ee0",
    "permisos": "#729fcf",
    "points": "#819f41",
    "questions": "#cc0000",
    "refactor": "#90b816",
    "register": "#13a282",
    "relationship": "#6a73a9",
    "responsive": "#786745",
    "retrospectiva": "#6d4244",
    "roles": "#037c88",
    "spinner": "#6be317",
    "sprint": "#b9f866",
    "stats": "#e350d5",
    "style": "#26ec8d",
    "subida de ficheros": "#4c0568",
    "summary": "#05535e",
    "svg": "#8feba6",
    "tags": "#9b6ef5",
    "taiga reporting": "#1dae9d",
    "taiga-web": "#12b429",
    "task": "#7fbb72",
    "taskboard": "#0f4b15",
    "tasks": "#55d872",
    "test": "#5c3566",
    "text": "#372ea0",
    "title": "#3c6de1",
    "us": "#888a85",
    "user-settings": "#c44e5a",
    "uss": "#219112",
    "ux": "#796384",
    "ux/dise\u00f1o pendiente": "#94dfcb",
    "videoconferencia": "#8ae234",
    "web": "#ca84d1",
    "wiki": "#204a87"
```

```
}
```


40.8. Project voter detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user6532909695705815086"
}
```


40.9. Project watcher detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user6532909695705815086"
}
```


40.10. Project template detail


```
{
    "created_date": "2014-11-11T09:58:35+0000",
    "default_options": {
        "issue_status": "New",
        "issue_type": "Bug",
        "points": "?",
        "priority": "Normal",
        "severity": "Normal",
        "task_status": "New",
        "us_status": "New"
    },
    "default_owner_role": "ux",
    "description": "Beta template description",
    "id": 3,
    "is_backlog_activated": true,
    "is_issues_activated": true,
    "is_kanban_activated": false,
    "is_wiki_activated": true,
    "issue_statuses": [
        {
            "color": "#8C2318",
            "is_closed": false,
```

```
            "name": "New",
            "order": 1
        },
        {
            "color": "#5E8C6A",
            "is_closed": false,
            "name": "In progress",
            "order": 2
        },
        {
            "color": "#88A65E",
            "is_closed": true,
            "name": "Ready for test",
            "order": 3
        },
        {
            "color": "#BFB35A",
            "is_closed": true,
            "name": "Closed",
            "order": 4
        },
        {
            "color": "#89BAB4",
            "is_closed": false,
            "name": "Needs Info",
            "order": 5
        },
        {
            "color": "#CC0000",
            "is_closed": true,
            "name": "Rejected",
            "order": 6
        },
        {
            "color": "#666666",
            "is_closed": false,
            "name": "Postponed",
            "order": 7
        }
    ],
    "issue_types": [
        {
            "color": "#89BAB4",
            "name": "Bug",
            "order": 1
        },
        {
            "color": "#ba89a8",
```

```
            "name": "Question",
            "order": 2
        },
        {
            "color": "#89a8ba",
            "name": "Enhancement",
            "order": 3
        }
    ],
    "modified_date": "2014-11-11T09:58:35+0000",
    "name": "Beta template",
    "points": [
        {
            "name": "?",
            "order": 1,
            "value": null
        },
        {
            "name": "0",
            "order": 2,
            "value": 0.0
        },
        {
            "name": "1/2",
            "order": 3,
            "value": 0.5
        },
        {
            "name": "1",
            "order": 4,
            "value": 1.0
        },
        {
            "name": "2",
            "order": 5,
            "value": 2.0
        },
        {
            "name": "3",
            "order": 6,
            "value": 3.0
        },
        {
            "name": "5",
            "order": 7,
            "value": 5.0
        },
        {
```

```
            "name": "8",
            "order": 8,
            "value": 8.0
        },
        {
            "name": "10",
            "order": 9,
            "value": 10.0
        },
        {
            "name": "15",
            "order": 10,
            "value": 15.0
        },
        {
            "name": "20",
            "order": 11,
            "value": 20.0
        },
        {
            "name": "40",
            "order": 12,
            "value": 40.0
        }
    ],
    "priorities": [
        {
            "color": "#666666",
            "name": "Low",
            "order": 1
        },
        {
            "color": "#669933",
            "name": "Normal",
            "order": 3
        },
        {
            "color": "#CC0000",
            "name": "High",
            "order": 5
        }
    ],
    "roles": [
        {
            "computable": true,
            "name": "UX",
            "order": 10,
            "permissions": [
```

```
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "add_milestone",
                "modify_milestone",
                "delete_milestone",
                "view_milestones",
                "view_project",
                "add_task",
                "modify_task",
                "delete_task",
                "view_tasks",
                "add_us",
                "modify_us",
                "delete_us",
                "view_us",
                "add_wiki_page",
                "modify_wiki_page",
                "delete_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "ux"
        },
        {
            "computable": true,
            "name": "Design",
            "order": 20,
            "permissions": [
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "add_milestone",
                "modify_milestone",
                "delete_milestone",
                "view_milestones",
                "view_project",
                "add_task",
                "modify_task",
                "delete_task",
                "view_tasks",
                "add_us",
                "modify_us",
                "delete_us",
```

```
                "view_us",
                "add_wiki_page",
                "modify_wiki_page",
                "delete_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "design"
        },
        {
            "computable": true,
            "name": "Front",
            "order": 30,
            "permissions": [
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "add_milestone",
                "modify_milestone",
                "delete_milestone",
                "view_milestones",
                "view_project",
                "add_task",
                "modify_task",
                "delete_task",
                "view_tasks",
                "add_us",
                "modify_us",
                "delete_us",
                "view_us",
                "add_wiki_page",
                "modify_wiki_page",
                "delete_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "front"
        },
        {
            "computable": true,
            "name": "Back",
            "order": 40,
            "permissions": [
```

```
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "add_milestone",
                "modify_milestone",
                "delete_milestone",
                "view_milestones",
                "view_project",
                "add_task",
                "modify_task",
                "delete_task",
                "view_tasks",
                "add_us",
                "modify_us",
                "delete_us",
                "view_us",
                "add_wiki_page",
                "modify_wiki_page",
                "delete_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "back"
        },
        {
            "computable": false,
            "name": "Product Owner",
            "order": 50,
            "permissions": [
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "add_milestone",
                "modify_milestone",
                "delete_milestone",
                "view_milestones",
                "view_project",
                "add_task",
                "modify_task",
                "delete_task",
                "view_tasks",
                "add_us",
                "modify_us",
                "delete_us",
```

```
                "view_us",
                "add_wiki_page",
                "modify_wiki_page",
                "delete_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "product-owner"
        },
        {
            "computable": false,
            "name": "Stakeholder",
            "order": 60,
            "permissions": [
                "add_issue",
                "modify_issue",
                "delete_issue",
                "view_issues",
                "view_milestones",
                "view_project",
                "view_tasks",
                "view_us",
                "modify_wiki_page",
                "view_wiki_pages",
                "add_wiki_link",
                "delete_wiki_link",
                "view_wiki_links"
            ],
            "slug": "stakeholder"
        }
    ],
    "severities": [
        {
            "color": "#666666",
            "name": "Wishlist",
            "order": 1
        },
        {
            "color": "#669933",
            "name": "Minor",
            "order": 2
        },
        {
            "color": "#0000FF",
            "name": "Normal",
            "order": 3
```

```
        },
        {
            "color": "#FFA500",
            "name": "Important",
            "order": 4
        },
        {
            "color": "#CC0000",
            "name": "Critical",
            "order": 5
        }
    ],
    "slug": "beta-template",
    "task_statuses": [
        {
            "color": "#999999",
            "is_closed": false,
            "name": "New",
            "order": 1
        },
        {
            "color": "#ff9900",
            "is_closed": false,
            "name": "In progress",
            "order": 2
        },
        {
            "color": "#ffcc00",
            "is_closed": true,
            "name": "Ready for test",
            "order": 3
        },
        {
            "color": "#669900",
            "is_closed": true,
            "name": "Closed",
            "order": 4
        },
        {
            "color": "#999999",
            "is_closed": false,
            "name": "Needs Info",
            "order": 5
        }
    ],
    "us_statuses": [
        {
            "color": "#999999",
```

```
            "is_closed": false,
            "name": "New",
            "order": 1,
            "wip_limit": null
        },
        {
            "color": "#ff8a84",
            "is_closed": false,
            "name": "Ready",
            "order": 2,
            "wip_limit": null
        },
        {
            "color": "#ff9900",
            "is_closed": false,
            "name": "In progress",
            "order": 3,
            "wip_limit": null
        },
        {
            "color": "#fcc000",
            "is_closed": false,
            "name": "Ready for test",
            "order": 4,
            "wip_limit": null
        },
        {
            "color": "#669900",
            "is_closed": true,
            "name": "Done",
            "order": 5,
            "wip_limit": null
        }
    ],
    "videoconferences": null,
    "videoconferences_extra_data": null
}
```


40.11. Membership detail

```
{
    "role_name": "Front",
    "full_name": "Alicia Diaz",
    "user_email": "accusamus@doloribus.net",
    "email": "accusamus@doloribus.net",
    "color": "#C0FF33",
    "photo": "//www.gravatar.com/avatar/e2cc11c9138524224f87aa72c1a343db?size=80",
    "project_name": "Project Example 0",
    "project_slug": "user6532909695705815086-project-example-0",
    "invited_by": null,
    "id": 7,
    "user": 8,
    "project": 1,
    "role": 3,
    "is_owner": true,
    "created_at": "2014-11-17T16:19:04+0000",
    "invitation_extra_text": null
}
```


40.12. Milestone detail


```
{
    "is_watcher": false,
    "total_watchers": 0,
    "user_stories": [...],
    "total_points": 172.5,
    "closed_points": 57.0,
    "client_increment_points": 0,
    "team_increment_points": 0,
    "id": 1,
    "name": "Sprint 2015-8-7",
    "slug": "sprint-2015-8-7",
    "owner": 7,
    "project": 1,
    "estimated_start": "2015-08-07",
    "estimated_finish": "2015-08-22",
    "created_date": "2015-08-07T09:52:54+0000",
    "modified_date": "2015-10-01T09:52:54+0000",
    "closed": false,
    "disponibility": 0.0,
    "order": 10
}
```

40.13. Milestone watcher detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user6532909695705815086"
}
```


40.14. Milestone stats detail


```
{
    "name": "Sprint 1",
    "estimated_start": "2014-10-20",
    "estimated_finish": "2014-10-21",
    "total_points": {
        "11": 10,
        "12": 40,
        "13": 30,
        "14": 20
    },
    "completed_points": 40,
    "total_userstories": 10,
    "completed_userstories": 4,
    "total_tasks": 40,
    "completed_tasks": 12,
    "iocaine_doses": 3,
    "days": [
        {
            "day": "2014-10-20",
            "name": "21",
            "open_points": 100,
            "optimal_points": 50
        },
        {
            "day": "2014-10-21",
            "name": "21"
            "open_points": 60,
            "optimal_points": 0
        }
    ]
}
```

40.15. User story detail


```
{
    "is_voter": true,
    "total_voters": 2,
    "is_watcher": false,
    "total_watchers": 1,
    "watchers": [
        12
    ],
    "tags": [
        "officiis",
        "est",
        "expedita"
    ],
    "external_reference": null,
    "points": {
        "2": 7,
        "3": 10,
        "4": 7,
        "1": 8
    },
    "total_points": 31.0,
    "comment": "",
    "milestone_slug": "sprint-2015-8-7",
    "milestone_name": "Sprint 2015-8-7",
    "origin_issue": null,
    "blocked_note_html": "",
    "description_html": "<p>Cum explicabo esse quis impedit earum dolore, voluptatibus
atque provident vel, impedit mollitia eius debitis saepe veritatis praesentium recusandae
rem consequuntur alias, recusandae architecto blanditiis odit nihil enim dolores? Ab rem
totam et delectus blanditiis, numquam iusto accusamus blanditiis tempore asperiores odit
dolor voluptatibus officiis perferendis sunt, tempora nihil nobis quo, dolores porro
ducimus sapiente aperiam nesciunt exercitationem nam hic et deserunt saepe, et officia
modi laboriosam reiciendis placeat quisquam nostrum eum? Iste corrupti repellat, nam
voluptates architecto, in fugiat tempora, ex similique neque voluptatibus?</p>",
    "status_extra_info": {
        "name": "Ready",
        "color": "#ff8a84"
    },
    "assigned_to_extra_info": {
        "username": "user8",
        "full_name_display": "Lourdes Aguilar",
        "photo":
"//www.gravatar.com/avatar/dce0e8ed702cd85d5132e523121e619b?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
```

```
        "big_photo":
"//www.gravatar.com/avatar/dce0e8ed702cd85d5132e523121e619b?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "owner_extra_info": {
        "username": "user2",
        "full_name_display": "Sofia Ramirez",
        "photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
        "big_photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "id": 1,
    "version": 1,
    "is_blocked": false,
    "blocked_note": "",
    "ref": 1,
    "milestone": 1,
    "project": 1,
    "owner": 8,
    "status": 2,
    "is_closed": false,
    "backlog_order": 10000,
    "sprint_order": 10000,
    "kanban_order": 10000,
    "created_date": "2015-10-01T09:52:54+0000",
    "modified_date": "2015-10-01T09:52:54+0000",
    "finish_date": null,
    "subject": "Create the html template",
    "description": "Cum explicabo esse quis impedit earum dolore, voluptatibus atque
provident vel, impedit mollitia eius debitis saepe veritatis praesentium recusandae rem
consequuntur alias, recusandae architecto blanditiis odit nihil enim dolores? Ab rem
totam et delectus blanditiis, numquam iusto accusamus blanditiis tempore asperiores odit
dolor voluptatibus officiis perferendis sunt, tempora nihil nobis quo, dolores porro
ducimus sapiente aperiam nesciunt exercitationem nam hic et deserunt saepe, et officia
modi laboriosam reiciendis placeat quisquam nostrum eum? Iste corrupti repellat, nam
voluptates architecto, in fugiat tempora, ex similique neque voluptatibus?",
    "assigned_to": 14,
    "client_requirement": false,
    "team_requirement": false,
    "generated_from_issue": null,
    "neighbors": {
        "next": {
            "id": 2,
```

```
            "ref": 7,
            "subject": "get_actions() does not check for 'delete_selected' in actions"
        },
        "previous": {}
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.16. User story detail (GET)


```
{
    "assigned_to": 19,
    "backlog_order": 2,
    "blocked_note": "",
    "blocked_note_html": "",
    "client_requirement": false,
    "comment": "",
    "created_date": "2014-07-29T07:15:50+0000",
    "description": "Implement API CALL",
    "description_html": "<p>Implement API CALL</p>",
    "finish_date": "2014-09-16T17:17:16+0000",
    "generated_from_issue": null,
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "kanban_order": 37,
    "milestone": 98,
    "milestone_name": "Sprint 03",
    "milestone_slug": "sprint-03",
    "modified_date": "2014-10-10T12:07:13+0000",
    "origin_issue": null,
    "owner": 19,
    "points": {
        "129": 361,
        "130": 361,
        "131": 361,
        "132": 364
    },
    "project": 31,
    "ref": 31,
    "sprint_order": 2,
    "status": 83,
    "subject": "Customer personal data",
    "tags": [
```

```
        "service catalog",
        "customer"
    ],
    "team_requirement": false,
    "total_points": 1.0,
    "version": 8,
    "watchers": [],
    "neighbors": {
        "previous": {
            "id": 10969,
            "ref": 1286,
            "subject": "US/Task/Issue Visualization and edition refactor"
        },
        "next": {
            "id": 9085,
            "ref": 1212,
            "subject": "Limit sizes"
        }
    },
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```


NOTE neighbors, status_extra_info and assigned_to_extra_info are read only fields

40.17. User story detail (LIST)


```
{
    "assigned_to": 19,
    "backlog_order": 2,
    "blocked_note": "",
    "blocked_note_html": "",
    "client_requirement": false,
    "comment": "",
    "created_date": "2014-07-29T07:15:50+0000",
    "finish_date": "2014-09-16T17:17:16+0000",
```

```
    "generated_from_issue": null,
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "kanban_order": 37,
    "milestone": 98,
    "milestone_name": "Sprint 03",
    "milestone_slug": "sprint-03",
    "modified_date": "2014-10-10T12:07:13+0000",
    "origin_issue": null,
    "owner": 19,
    "points": {
        "129": 361,
        "130": 361,
        "131": 361,
        "132": 364
    },
    "project": 31,
    "ref": 31,
    "sprint_order": 2,
    "status": 83,
    "subject": "Customer personal data",
    "tags": [
        "service catalog",
        "customer"
    ],
    "team_requirement": false,
    "total_points": 1.0,
    "version": 8,
    "watchers": [],
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.18. User story voter detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.19. User story watcher detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.20. User story status detail


```
{
    "color": "#669933",
    "id": 143,
    "is_archived": true,
    "is_closed": false,
    "name": "Open",
    "order": 1,
    "project": 40,
    "wip_limit": null
}
```


40.21. Point detail


```
{
    "color": "#669933",
    "id": 143,
    "name": "Huge",
    "order": 8,
    "value": 40,
    "project": 3
}
```

40.22. User story custom attribute detail


```
{
    "id": 1,
    "name": "Duration",
    "description": "Duration in minutes",
    "order": 1,
    "project": 1
}
```


40.23. User story custom attributes values detail


```
{
    "user_story": 1,
    "attributes_values": {
        "1": "240 minutes"
    },
    "version": 2
}
```


40.24. Task detail


```
{
    "is_voter": false,
    "total_voters": 3,
    "is_watcher": false,
    "total_watchers": 1,
    "watchers": [
        6
    ],
    "tags": [
        "porro",
        "placeat"
    ],
    "external_reference": null,
    "comment": "",
    "milestone_slug": "sprint-2015-8-7",
    "blocked_note_html": "",
    "description_html": "<p>Nostrum consequatur quasi odio, pariatur modi id distinctio
sequi. Voluptatibus illo ratione eius necessitatibus ad quibusdam nulla reiciendis
laborum doloribus, nostrum repudiandae asperiores eveniet libero consequuntur expedita
velit accusantium a commodi blanditiis? Quaerat ipsum pariatur nulla maiores harum ipsa
```

```
eveniet, vel omnis culpa quisquam nulla, minus iste ad modi aliquid inventore ab iusto
optio tempora hic voluptas?</p>",
    "is_closed": false,
    "status_extra_info": {
        "name": "Needs Info",
        "color": "#999999"
    },
    "assigned_to_extra_info": {
        "username": "user2",
        "full_name_display": "Sofia Ramirez",
        "photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
        "big_photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "owner_extra_info": {
        "username": "user3",
        "full_name_display": "Concepcion Garrido",
        "photo":
"//www.gravatar.com/avatar/9971a763f5dfc5cbd1ce1d2865b4fcfa?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
        "big_photo":
"//www.gravatar.com/avatar/9971a763f5dfc5cbd1ce1d2865b4fcfa?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "id": 1,
    "version": 1,
    "is_blocked": false,
    "blocked_note": "",
    "user_story": 1,
    "ref": 2,
    "owner": 9,
    "status": 5,
    "project": 1,
    "milestone": 1,
    "created_date": "2015-10-01T09:52:54+0000",
    "modified_date": "2015-10-01T09:52:55+0000",
    "finished_date": "2015-08-16T15:54:13+0000",
    "subject": "Create testsuite with matrix builds",
    "us_order": 1,
    "taskboard_order": 1,
    "description": "Nostrum consequatur quasi odio, pariatur modi id distinctio sequi.
Voluptatibus illo ratione eius necessitatibus ad quibusdam nulla reiciendis laborum
doloribus, nostrum repudiandae asperiores eveniet libero consequuntur expedita velit
```

```
accusantium a commodi blanditiis? Quaerat ipsum pariatur nulla maiores harum ipsa
eveniet, vel omnis culpa quisquam nulla, minus iste ad modi aliquid inventore ab iusto
optio tempora hic voluptas?",
    "assigned_to": 8,
    "is_iocaine": false,
    "neighbors": {
        "next": {
            "id": 2,
            "ref": 3,
            "subject": "Lighttpd support"
        },
        "previous": {}
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.25. Task detail (GET)


```
{
    "assigned_to": 19,
    "blocked_note": "",
    "blocked_note_html": "",
    "comment": "",
    "milestone_slug": "sprint-2014-11-1",
    "created_date": "2014-07-29T07:15:50+0000",
    "description": "Implement API CALL",
    "description_html": "<p>Implement API CALL</p>",
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "milestone": 98,
    "modified_date": "2014-10-10T12:07:13+0000",
    "finished_date": null,
    "is_voted": true,
    "is_watched": true,
    "owner": 19,
    "project": 31,
    "user_story": 17,
    "ref": 31,
    "status": 83,
    "subject": "Customer personal data",
    "tags": [
        "service catalog",
        "customer"
```

```
    ],
    "us_order": 1,
    "taskboard_order": 1,
    "version": 8,
    "is_iocaine": false,
    "external_reference": null,
    "voters": 1,
    "watchers": [],
    "neighbors": {
        "previous": {
            "id": 57,
            "ref": 74,
            "subject": "get_actions() does not check for 'delete_selected' in actions"
        },
        "next": {
            "id": 59,
            "ref": 77,
            "subject": "Feature/improved image admin"
        }
    },
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```


NOTE neighbors, us_extra_info and assigned_to_extra_info are read only fields

40.26. Task detail (LIST)

```
{
    "assigned_to": 19,
    "blocked_note": "",
    "blocked_note_html": "",
    "comment": "",
    "milestone_slug": "sprint-2014-11-1",
    "created_date": "2014-07-29T07:15:50+0000",
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "milestone": 98,
    "modified_date": "2014-10-10T12:07:13+0000",
    "finished_date": null,
    "owner": 19,
    "project": 31,
    "user_story": 17,
    "ref": 31,
    "status": 83,
    "subject": "Customer personal data",
    "tags": [
        "service catalog",
        "customer"
    ],
    "us_order": 1,
    "taskboard_order": 1,
    "version": 8,
    "is_iocaine": false,
    "external_reference": null,
    "watchers": [],
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.27. Task voter detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.28. Task watcher detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.29. Project voter detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.30. Project watcher detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.31. Task status detail

```
{
    "color": "#669933",
    "id": 143,
    "is_closed": false,
    "name": "Open",
    "order": 1,
    "project": 40
}
```


40.32. Task custom attribute detail


```
{
    "id": 1,
    "name": "Duration",
    "description": "Duration in minutes",
    "order": 1,
    "project": 1
}
```


40.33. Task custom attributes values detail


```
{
    "task": 1,
    "attributes_values": {
        "1": "240 minutes"
    },
    "version": 2
}
```


40.34. Issue detail


```
{
    "is_voter": false,
    "total_voters": 1,
    "is_watcher": false,
    "total_watchers": 1,
    "watchers": [
        15
    ],
    "tags": [
        "dicta",
```

```
        "dolores",
        "eum",
        "amet"
    ],
    "external_reference": null,
    "is_closed": false,
    "comment": "",
    "generated_user_stories": [],
    "blocked_note_html": "",
    "description_html": "<p>Modi numquam esse qui sequi porro optio tenetur assumenda
temporibus consequatur ad, aspernatur veniam perferendis quia asperiores dolorem nihil
impedit, libero aperiam quis, perspiciatis soluta quis iste? Id quo assumenda distinctio?
Cupiditate corporis distinctio placeat dolor commodi delectus eum recusandae, non autem
maiores saepe quia cum quasi nisi, a accusantium laudantium nihil eum adipisci aut
voluptas debitis enim. Voluptatum voluptas saepe perferendis, a similique saepe sint iure
numquam sed voluptas aliquam, laboriosam voluptate sequi?</p>",
    "status_extra_info": {
        "name": "Needs Info",
        "color": "#89BAB4"
    },
    "assigned_to_extra_info": {
        "username": "user1",
        "full_name_display": "Purificacion Montero",
        "photo":
"//www.gravatar.com/avatar/aed1e43be0f69f07ce6f34a907bc6328?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
        "big_photo":
"//www.gravatar.com/avatar/aed1e43be0f69f07ce6f34a907bc6328?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "owner_extra_info": {
        "username": "user2",
        "full_name_display": "Sofia Ramirez",
        "photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
        "big_photo":
"//www.gravatar.com/avatar/5c921c7bd676b7b4992501005d243c42?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=300",
        "is_active": true
    },
    "id": 1,
    "version": 1,
    "is_blocked": false,
    "blocked_note": "",
    "ref": 43,
    "owner": 8,
```

```
    "status": 5,
    "severity": 3,
    "priority": 3,
    "type": 3,
    "milestone": null,
    "project": 1,
    "created_date": "2015-10-01T09:53:06+0000",
    "modified_date": "2015-10-01T09:53:07+0000",
    "finished_date": null,
    "subject": "Migrate to Python 3 and milk a beautiful cow",
    "description": "Modi numquam esse qui sequi porro optio tenetur assumenda temporibus
consequatur ad, aspernatur veniam perferendis quia asperiores dolorem nihil impedit,
libero aperiam quis, perspiciatis soluta quis iste? Id quo assumenda distinctio?
Cupiditate corporis distinctio placeat dolor commodi delectus eum recusandae, non autem
maiores saepe quia cum quasi nisi, a accusantium laudantium nihil eum adipisci aut
voluptas debitis enim. Voluptatum voluptas saepe perferendis, a similique saepe sint iure
numquam sed voluptas aliquam, laboriosam voluptate sequi?",
    "assigned_to": 7,
    "neighbors": {
        "next": {
            "id": 35,
            "ref": 83,
            "subject": "Lighttpd support"
        },
        "previous": {
            "id": 2,
            "ref": 44,
            "subject": "Lighttpd support"
        }
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.35. Issue detail (GET)


```
{
    "assigned_to": 19,
    "blocked_note": "",
    "blocked_note_html": "",
    "comment": "",
    "created_date": "2014-07-29T07:15:50+0000",
    "description": "Implement API CALL",
    "description_html": "<p>Implement API CALL</p>",
    "finish_date": "2014-09-16T17:17:16+0000",
```

```
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "milestone": 98,
    "modified_date": "2014-10-10T12:07:13+0000",
    "finished_date": null,
    "is_voted": true,
    "is_watched": true,
    "owner": 19,
    "project": 31,
    "ref": 31,
    "status": 83,
    "severity": 2,
    "priority": 3,
    "type": 1,
    "subject": "Customer personal data",
    "tags": [
        "service catalog",
        "customer"
    ],
    "version": 8,
    "voters": 1,
    "watchers": [
        7
    ],
    "generated_user_stories": [
    ],
    "votes": null,
    "neighbors": {
        "next": {
            "id": 16,
            "ref": 126,
            "subject": "Support for bulk actions"
        },
        "previous": {}
    },
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```

NOTE	neighbors, status_extra_info and assigned_to_extra_info are read only fields

40.36. Issue detail (LIST)

```
{
    "assigned_to": 19,
    "blocked_note": "",
    "blocked_note_html": "",
    "comment": "",
    "created_date": "2014-07-29T07:15:50+0000",
    "finish_date": "2014-09-16T17:17:16+0000",
    "id": 1149,
    "is_blocked": false,
    "is_closed": true,
    "milestone": 98,
    "modified_date": "2014-10-10T12:07:13+0000",
    "finished_date": null,
    "owner": 19,
    "project": 31,
    "ref": 31,
    "status": 83,
    "severity": 2,
    "priority": 3,
    "type": 1,
    "subject": "Customer personal data",
    "tags": [
        "service catalog",
        "customer"
    ],
    "version": 8,
    "watchers": [],
    "generated_user_stories": [
    ],
    "votes": null,
    "status_extra_info": {
        "name": "In progress",
        "color": "#5E8C6A"
    },
    "assigned_to_extra_info": {
        "username": "beta.tester",
        "full_name_display": "Beta testing",
        "photo": "//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80",
        "big_photo":
"//www.gravatar.com/avatar/4648b6d514c3ecece1b87136ceeda1d1?size=80""
    }
}
```


NOTE status_extra_info and assigned_to_extra_info are read only fields

40.37. Issue voters detail


```
[
    {
        "id": 1,
        "username": "user-1",
        "full_name": "John Doe"
    },
    {
        "id": 2,
        "username": "user-2",
        "full_name": "User 2"
    }
]
```


40.38. Issue watchers detail


```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.39. Issue status detail


```
{
    "color": "#669933",
    "id": 143,
    "is_closed": false,
    "name": "Open",
    "order": 1,
    "project": 40
}
```


40.40. Issue type detail

```
{
    "color": "#669933",
    "id": 143,
    "name": "Enhancement",
    "order": 1,
    "project": 40
}
```


40.41. Priority detail


```
{
    "color": "#669933",
    "id": 143,
    "name": "High",
    "order": 1,
    "project": 40
}
```


40.42. Severity detail


```
{
    "color": "#669933",
    "id": 143,
    "name": "Important",
    "order": 1,
    "project": 40
}
```


40.43. Issue custom attribute detail


```
{
    "id": 1,
    "name": "Duration",
    "description": "Duration in minutes",
    "order": 1,
    "project": 1
}
```

40.44. Issue custom attributes values detail


```
{
    "issue": 1,
    "attributes_values": {
        "1": "240 minutes"
    },
    "version": 2
}
```


40.45. Wiki page


```
{
    "is_watcher": true,
    "total_watchers": 1,
    "html": "<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.</p>\n<p>Facilis molestias nulla est harum saepe maiores quisquam,
adipisci voluptas debitis, expedita aspernatur architecto obcaecati blanditiis, quae et
delectus exercitationem pariatur minus modi debitis placeat? Suscipit corrupti architecto
eos provident quibusdam autem?</p>\n<p>Accusamus modi nesciunt nobis sapiente voluptate
non et, non necessitatibus nobis est qui nam hic a quidem rerum? Nisi quia sit distinctio
provident voluptatem dolores accusantium numquam, eos voluptatum distinctio cupiditate
quisquam aspernatur itaque, consequatur suscipit aut tenetur eius amet error est corporis
fuga, porro neque omnis ad tenetur fugit perspiciatis consequatur sit placeat, delectus
non repudiandae inventore. Reprehenderit enim perferendis adipisci molestiae nobis error
totam facilis excepturi, obcaecati pariatur itaque eius adipisci facilis saepe tempora
unde, debitis neque eaque laborum voluptates voluptas culpa, enim debitis nemo at
pariatur iure inventore saepe? Ipsam asperiores itaque blanditiis voluptas ipsa ipsum
nesciunt vero quod eius exercitationem, asperiores repudiandae placeat cum hic accusamus
perferendis blanditiis expedita quia eum nostrum, obcaecati sint eius totam eum veritatis
dolorem fugiat?</p>\n<p>Fuga rem adipisci, provident labore quae nobis earum, at nemo
aliquid aperiam autem libero laborum sit dolore dignissimos vel sunt, vitae aliquam
exercitationem vero recusandae mollitia quod, ratione ducimus quas nesciunt
recusandae?</p>\n<p>Voluptate temporibus vero, facilis doloribus nobis, atque ipsum
accusantium beatae quas velit ut quisquam suscipit amet. Totam dolorum iusto obcaecati
earum porro sunt nisi amet ipsam deserunt, nemo officia fugit sit neque ipsum assumenda
eaque maxime quasi harum, repudiandae dicta aut autem accusamus doloribus ab, beatae
necessitatibus at quasi quod tempora. Esse doloribus amet rem iusto, id quaerat vel
voluptates quae corrupti enim? Dolorum quod excepturi itaque ratione aut iste placeat
animi repellat quae?</p>",
```

```
    "editions": 2,
    "id": 1,
    "version": 1,
    "project": 1,
    "slug": "home",
    "content": "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.\n\nFacilis molestias nulla est harum saepe maiores quisquam,
adipisci voluptas debitis, expedita aspernatur architecto obcaecati blanditiis, quae et
delectus exercitationem pariatur minus modi debitis placeat? Suscipit corrupti architecto
eos provident quibusdam autem?\n\nAccusamus modi nesciunt nobis sapiente voluptate non
et, non necessitatibus nobis est qui nam hic a quidem rerum? Nisi quia sit distinctio
provident voluptatem dolores accusantium numquam, eos voluptatum distinctio cupiditate
quisquam aspernatur itaque, consequatur suscipit aut tenetur eius amet error est corporis
fuga, porro neque omnis ad tenetur fugit perspiciatis consequatur sit placeat, delectus
non repudiandae inventore. Reprehenderit enim perferendis adipisci molestiae nobis error
totam facilis excepturi, obcaecati pariatur itaque eius adipisci facilis saepe tempora
unde, debitis neque eaque laborum voluptates voluptas culpa, enim debitis nemo at
pariatur iure inventore saepe? Ipsam asperiores itaque blanditiis voluptas ipsa ipsum
nesciunt vero quod eius exercitationem, asperiores repudiandae placeat cum hic accusamus
perferendis blanditiis expedita quia eum nostrum, obcaecati sint eius totam eum veritatis
dolorem fugiat?\n\nFuga rem adipisci, provident labore quae nobis earum, at nemo aliquid
aperiam autem libero laborum sit dolore dignissimos vel sunt, vitae aliquam
exercitationem vero recusandae mollitia quod, ratione ducimus quas nesciunt
recusandae?\n\nVoluptate temporibus vero, facilis doloribus nobis, atque ipsum
accusantium beatae quas velit ut quisquam suscipit amet. Totam dolorum iusto obcaecati
earum porro sunt nisi amet ipsam deserunt, nemo officia fugit sit neque ipsum assumenda
eaque maxime quasi harum, repudiandae dicta aut autem accusamus doloribus ab, beatae
necessitatibus at quasi quod tempora. Esse doloribus amet rem iusto, id quaerat vel
voluptates quae corrupti enim? Dolorum quod excepturi itaque ratione aut iste placeat
animi repellat quae?",
    "owner": 13,
    "last_modifier": null,
    "created_date": "2015-10-01T09:53:12+0000",
    "modified_date": "2015-10-01T09:53:12+0000"
}
```


40.46. Wiki page watcher detail

```
{
    "full_name": "Silvia Soto",
    "id": 2,
    "username": "user65329096957058150 86"
}
```


40.47. Wiki link


```
{
    "id": 1,
    "project": 2,
    "title": "Test title",
    "href": "test",
    "order": 1
}
```


40.48. History entry


```
{
    "diff": {
        "team_requirement": [
            false,
            true
        ],
        "finish_date": [
            "None",
            "2014-11-17 16:19:04.411139+00:00"
        ],
        "is_closed": [
            false,
            true
        ]
    },
    "snapshot": null,
    "values": {},
    "values_diff": {
        "team_requirement": [
            false,
            true
        ],
        "finish_date": [
            "None",
            "2014-11-17 16:19:04.411139+00:00"
```

```
        ],
        "is_closed": [
            false,
            true
        ]
    },
    "user": {
        "pk": 2,
        "username":, "admin",
        "name": "admin",
        "photo: "//www.gravatar.com/avatar/64e1b8d34f425d19e1ee2ea7236d3028?size=80",
        "is_active": true
    },
    "delete_comment_user": null,
    "id": "9660411e-6fea-11e4-a5b3-b499ba565108",
    "created_at": "2014-11-19T12:50:06+0000",
    "type": 1,
    "key": "userstories.userstory:1",
    "comment": "",
    "comment_html": "",
    "delete_comment_date": null,
    "is_hidden": false,
    "is_snapshot": false
}
```


40.49. Notify policy


```
{
    "id": 97,
    "notify_level": 1,
    "project": 3,
    "project_name": "Testing project"
}
```


40.50. Feedback


```
{
    "comment": "testing feedback",
    "created_date": "2014-11-17T15:10:54+0000",
    "email": "testing@taiga.io",
    "id": 71,
    "full_name": "betatester"
}
```

40.51. Export detail for synch mode


```
{
    "url": "https://media.taiga.io/exports/21313/50f38116f1654076b130d1026f579fae.json"
}
```


40.52. Export accepted response


```
{
    "export_id": "e338555a-3918-4203-8fc6-81b3f7933c79"
}
```


40.53. Import accepted response


```
{
    "import_id": "e338555a-3918-4203-8fc6-81b3f7933c79"
}
```


40.54. Webhook


```
{
    "logs_counter": 6,
    "id": 1,
    "project": 1,
    "name": "webhook",
    "url": "http://localhost:8080/test",
    "key": "123"
}
```


40.55. Webhook log

```
{
    "request_data": {... payload data ...},
    "request_headers": {
        "Content-Length": "1573",
        "X-TAIGA-WEBHOOK-SIGNATURE": "32b30381523eda3df76854e84865467574fcc085"
    },
    "response_headers": {
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Length": "2",
        "Server": "WSGIServer/0.2 CPython/3.4.2",
        "Date": "Thu, 22 Jan 2015 10:43:56 GMT"
    },
    "id": 159,
    "webhook": 1,
    "url": "http://localhost:8080/test",
    "status": 200,
    "response_data": "OK",
    "duration": 0.002326,
    "created": "2015-01-22T10:43:56+0000"
}
```


40.56. Timeline entry detail


```
{
    "data": {
        "comment_html": "<p>Hic cum voluptatibus et quaerat suscipit incidunt, quasi enim
porro corrupti, accusantium accusamus quam odit deserunt ducimus, atque libero inventore
fuga numquam aspernatur dolorem pariatur fugiat cumque quas ad? Incidunt rem ut dolore
quidem sequi non neque esse quos sunt ea, quisquam non officia quidem minima a est eaque
dolorem esse impedit, minus deserunt repellendus. Quae dolores obcaecati iste voluptatum
facilis delectus quod aperiam totam possimus eum, totam voluptas quis temporibus sint
debitis aliquid in quae nulla, aspernatur voluptate velit amet, incidunt recusandae
repudiandae, culpa iure commodi corrupti modi excepturi? Omnis voluptas et assumenda, ex
nostrum qui iste commodi adipisci laborum, quaerat quasi enim, corporis laborum aliquid
eum.</p>",
        "values_diff": {
            "status": [
                "Rejected",
                "New"
            ]
        },
        "comment": "Hic cum voluptatibus et quaerat suscipit incidunt, quasi enim porro
corrupti, accusantium accusamus quam odit deserunt ducimus, atque libero inventore fuga
numquam aspernatur dolorem pariatur fugiat cumque quas ad? Incidunt rem ut dolore quidem
sequi non neque esse quos sunt ea, quisquam non officia quidem minima a est eaque dolorem
```

```
esse impedit, minus deserunt repellendus. Quae dolores obcaecati iste voluptatum facilis
delectus quod aperiam totam possimus eum, totam voluptas quis temporibus sint debitis
aliquid in quae nulla, aspernatur voluptate velit amet, incidunt recusandae repudiandae,
culpa iure commodi corrupti modi excepturi? Omnis voluptas et assumenda, ex nostrum qui
iste commodi adipisci laborum, quaerat quasi enim, corporis laborum aliquid eum.",
        "user": {
            "id": 7,
            "photo":
"//www.gravatar.com/avatar/aed1e43be0f69f07ce6f34a907bc6328?size=80&default=https%3A%2F%2
Fstatic.taiga.io%2Fimg%2Fuser-noimage.png",
            "big_photo":
"//www.gravatar.com/avatar/aed1e43be0f69f07ce6f34a907bc6328?default=https%3A%2F%2Fstatic.
taiga.io%2Fimg%2Fuser-noimage.png&size=80",
            "name": "Purificacion Montero",
            "username": "user1"
        },
        "project": {
            "description": "Project example 3 description",
            "id": 4,
            "slug": "user9-project-example-3",
            "name": "Project Example 3"
        },
        "issue": {
            "id": 83,
            "ref": 106,
            "subject": "Added file copying and processing of images (resizing)"
        }
    },
    "id": 4336,
    "content_type": 3,
    "object_id": 7,
    "namespace": "user:7",
    "event_type": "issues.issue.change",
    "project": 4,
    "data_content_type": 39,
    "created": "2015-04-16T06:57:21+0000"
}
```


40.57. Locale


```
{
    "name": "Español",
    "code": "es"
}
```

40.58. Watched


```
{
    "type": "project",
    "id": 6,
    "ref": null,
    "slug": "project-5",
    "name": "Project Example 5",
    "subject": null,
    "description": "Project example 5 description",
    "assigned_to": null,
    "status": null,
    "status_color": null,
    "tags_colors": [],
    "created_date": "2015-10-01T09:54:55+0000",
    "is_private": true,
    "is_watcher": false,
    "total_watchers": 11,
    "project": null,
    "project_name": null,
    "project_slug": null,
    "project_is_private": null,
    "assigned_to_username": null,
    "assigned_to_full_name": null,
    "assigned_to_photo": null,
    "is_fan": false,
    "total_fans": 3
}
```


NOTE is_fan and total_fans only appear for results with type "project", for issues, tasks and user stories those attributes are replaced by is_voter and total_voters

40.59. Liked

```
{
    "type": "project",
    "id": 5,
    "ref": null,
    "slug": "project-4",
    "name": "Project Example 4",
    "subject": null,
    "description": "Project example 4 description",
    "assigned_to": null,
    "status": null,
    "status_color": null,
    "tags_colors": [],
    "created_date": "2015-10-01T09:54:55+0000",
    "is_private": true,
    "project": null,
    "project_name": null,
    "project_slug": null,
    "project_is_private": null,
    "assigned_to_username": null,
    "assigned_to_full_name": null,
    "assigned_to_photo": null,
    "is_watcher": true,
    "total_watchers": 14,
    "is_fan": false,
    "total_fans": 3
}
```


40.60. Voted

```
{
    "type": "userstory",
    "id": 5,
    "ref": null,
    "slug": "project-4",
    "name": "Project Example 4",
    "subject": null,
    "description": "Project example 4 description",
    "assigned_to": null,
    "status": null,
    "status_color": null,
    "tags_colors": [],
    "created_date": "2015-10-01T09:54:55+0000",
    "is_private": true,
    "project": null,
    "project_name": null,
    "project_slug": null,
    "project_is_private": null,
    "assigned_to_username": null,
    "assigned_to_full_name": null,
    "assigned_to_photo": null,
    "is_watcher": true,
    "total_watchers": 14,
    "is_voter": false,
    "total_voters": 3
}
```


41. Contrib plugins

Taiga allows adding features through contrib plugins, each plugin can add new API endpoints, and has its own documentation.

Current supported contrib plugins that adding endpoints:

- taiga-contrib-slack: Slack integration (documentation)
- taiga-contrib-hall: Hall.com integration (documentation)