MOREGATE

MGProject

The complete guide

Installation, administration and daily use

2.0

Contents

1 What MGProject is

2 Requirements

3 Installation

4 Licence and users

5 Accounts, groups and signing in

6 Projects and boards

7 Working with issues

8 Boards, backlog and sprints

9 Planning Overview

10 Time recording

11 Reports

12 Workflows and statuses

13 Issue types, fields and screens

14 Permissions and notifications

15 Searching with MQL

16 Routines and automation

17 Webhooks and links

18 Service Management

19 Bitbucket Cloud

20 Importing from Jira

21 Your own look, and announcements

22 Attachments and the viewer

23 Mail queue

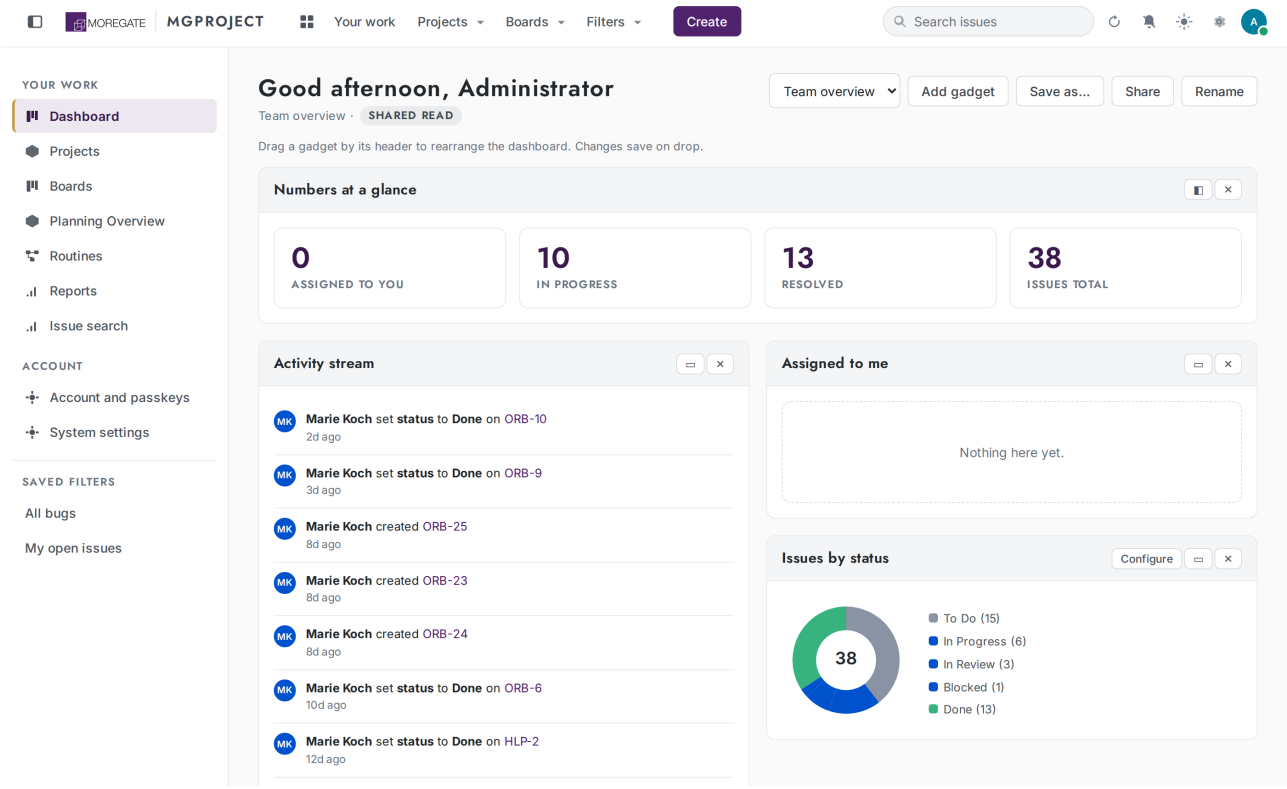
24 Audit log

25 Keeping it running

01

What MGProject is

MGProject is an issue tracker you run yourself. Projects, boards, sprints, a backlog, planning overviews, time recording, routines, automation and reports - on your own server, with your own database, and without anything leaving the house.



The dashboard after signing in: gadgets you choose yourself.

This guide describes every function in the program. Read the first three chapters if you are installing it; start at “Working with issues” if somebody has done that for you.

02

Requirements

Required	
Operating system	Ubuntu 24.04 LTS or newer, 64-bit (x86-64)
Memory	2 GB, 4 GB with a busy PostgreSQL
Disk	1 GB, plus room for attachments
Database	PostgreSQL 14 or newer - SQLite is gone from 2.0 on
Office preview	LibreOffice; deploy.sh asks whether to install it
Python	not needed - the package brings its own
Network	none at install time
In front	nginx or HAProxy for TLS, if reachable from outside

Ubuntu 24.04 LTS is the floor, and newer releases are fine. The package carries its own Python runtime, so you may upgrade, replace or remove the system Python without touching MGProject. What it ties to is the C library of the distribution.

Browsers: any current Firefox, Chrome, Edge or Safari. Passkeys need a device that offers them and a host name - they do not work on a bare IP address.

03

Installation

1 Unpack the package on the server.
2 Run the deployment script as root.
3 Note the administrator password it prints - it is shown once.

```
unzip mgproject-2_0-2026-10-20-1203-linux-x86_64.zip
cd mgproject
sudo ./deploy.sh
```

The script creates the service account, installs the application under /opt/mgproject, puts the data at /var/lib/mgproject and starts a systemd unit called mgproject.service. It listens on 127.0.0.1:6633. Nothing is fetched from the internet.

Settings in the environment

```
MGPROJECT_SECRET=a-long-random-string
MGPROJECT_URL=https://project.example.com
MGPROJECT_MAX_UPLOAD_MB=100
```

Key	What for
MGPROJECT_SECRET	signs the sessions; change it and everybody signs in again
MGPROJECT_URL	what notification e-mails link to
MGPROJECT_MAX_UPLOAD_MB	the ceiling for attachments

The database

On its first run deploy.sh asks for the PostgreSQL connection - server, database, user, password - and keeps it in database.json in the data directory. The database itself has to exist; MGProject creates its schema when it starts.

```
sudo -u postgres createuser --pwprompt mgproject
sudo -u postgres createdb -O mgproject mgproject
```

An installation still running on SQLite is moved across once during the update to 2.0. Its SQLite file is left untouched - it is the way back.

A second, independent installation

deploy.sh asks whether the default installation is meant or a further one: its own paths, service, account, port and database, checked against everything already on the server first. In the interface the two are told apart by the installation's name (System settings › Instance).

Behind a reverse proxy

```
location / {
    proxy_pass http://127.0.0.1:6633;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
    client_max_body_size 100M;
    proxy_read_timeout 120s;
}
```

client_max_body_size is the usual reason a larger attachment is refused: nginx allows one megabyte by default, whatever MGProject permits.

Updating

An update is the same command with a newer package. Data, logos and settings are left alone; only the application directory is replaced. deploy.sh backs the database up first with pg_dump, and schema changes run as numbered steps whose state is shown under System settings › Instance.

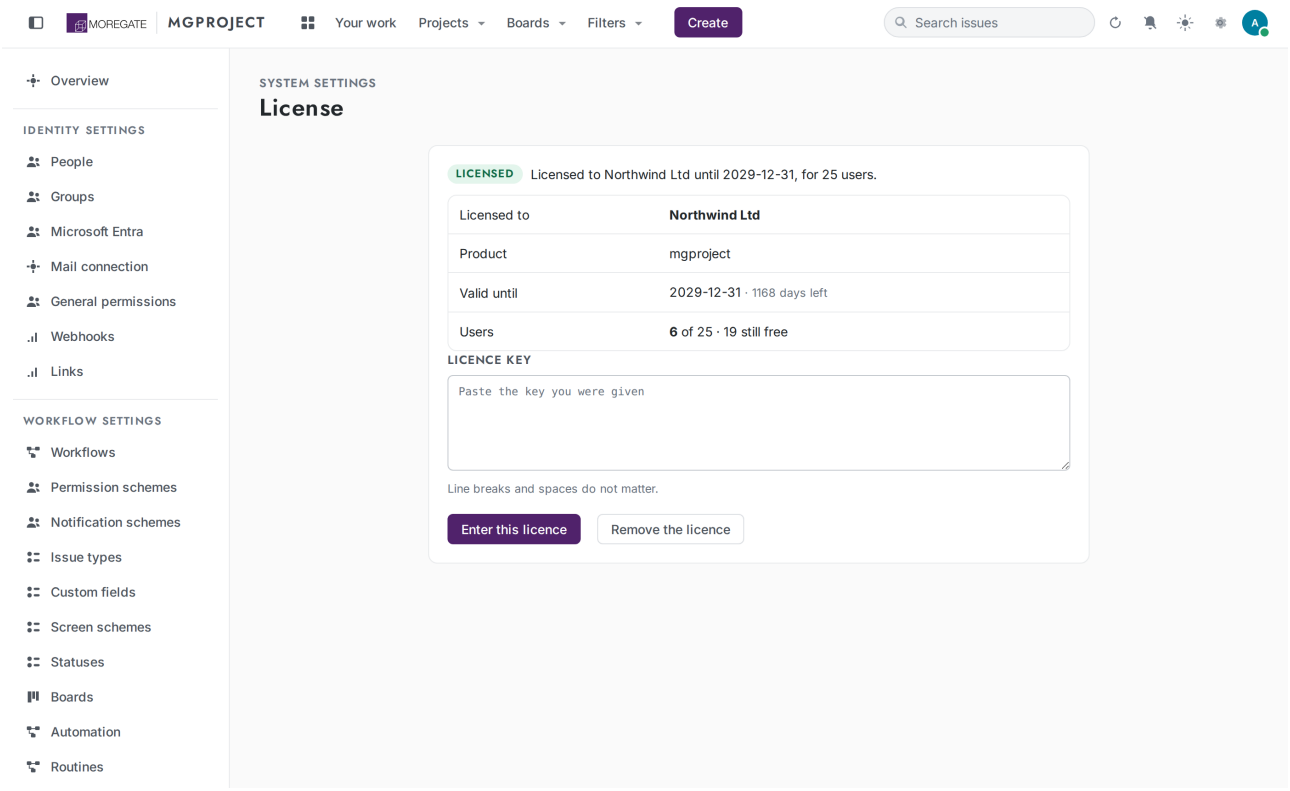
If anything goes wrong, one command brings program and database back to before the last update:

```
sudo ./deploy.sh --rollback
```

An update skips versions: from any 1.x straight to 2.0, through every step in between.

04

Licence and users



System settings License: customer, product, expiry and how many of the licensed seats are in use.

The first installation needs no licence. Sign in, change the password you were given, and work - nothing is locked. What a licence pays for is updates, and with them the fixes and the support.

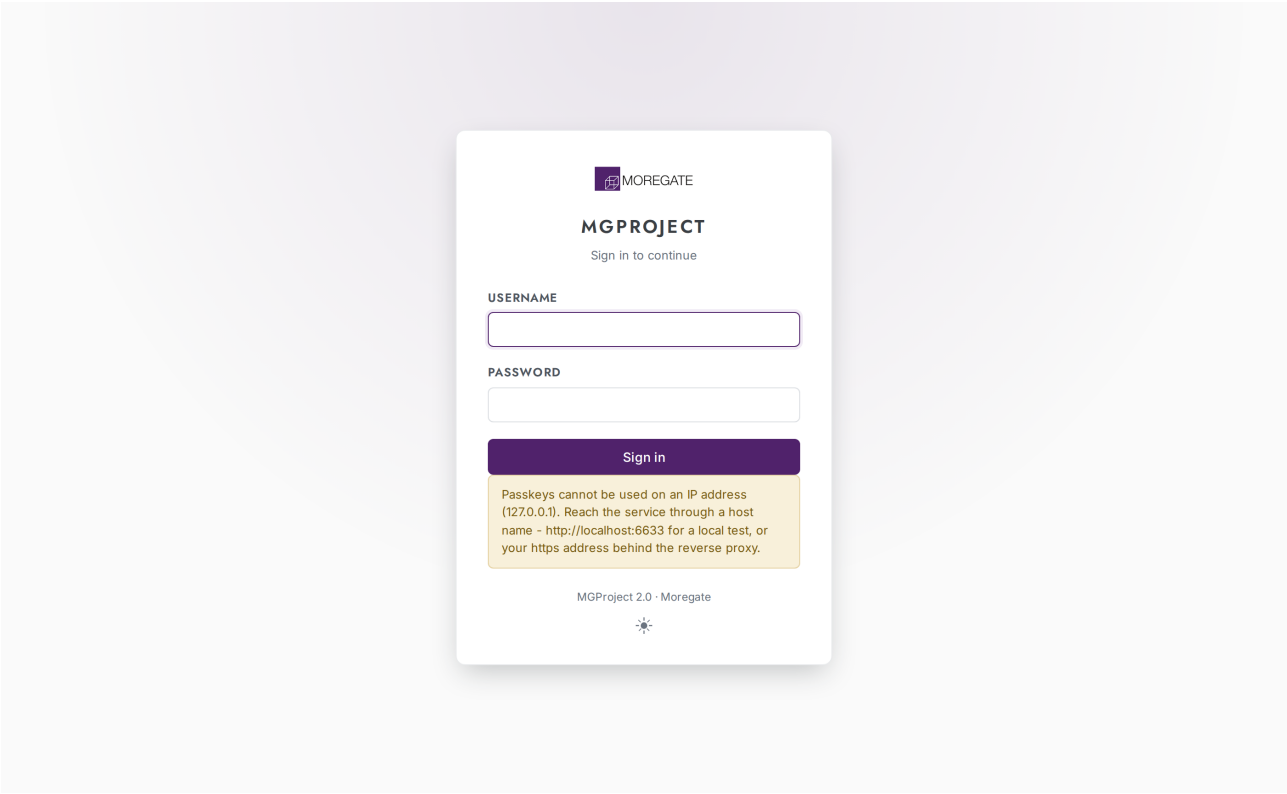
- Without a licence: the instance can be read but not changed
- An update refuses to install without a valid licence
- An expired licence still installs the update it was valid for

Accounts (licence for 25)	What happens
up to 25	normal
26 - 27	two in hand, a note on the licence page
28 and above	read only until accounts are deactivated

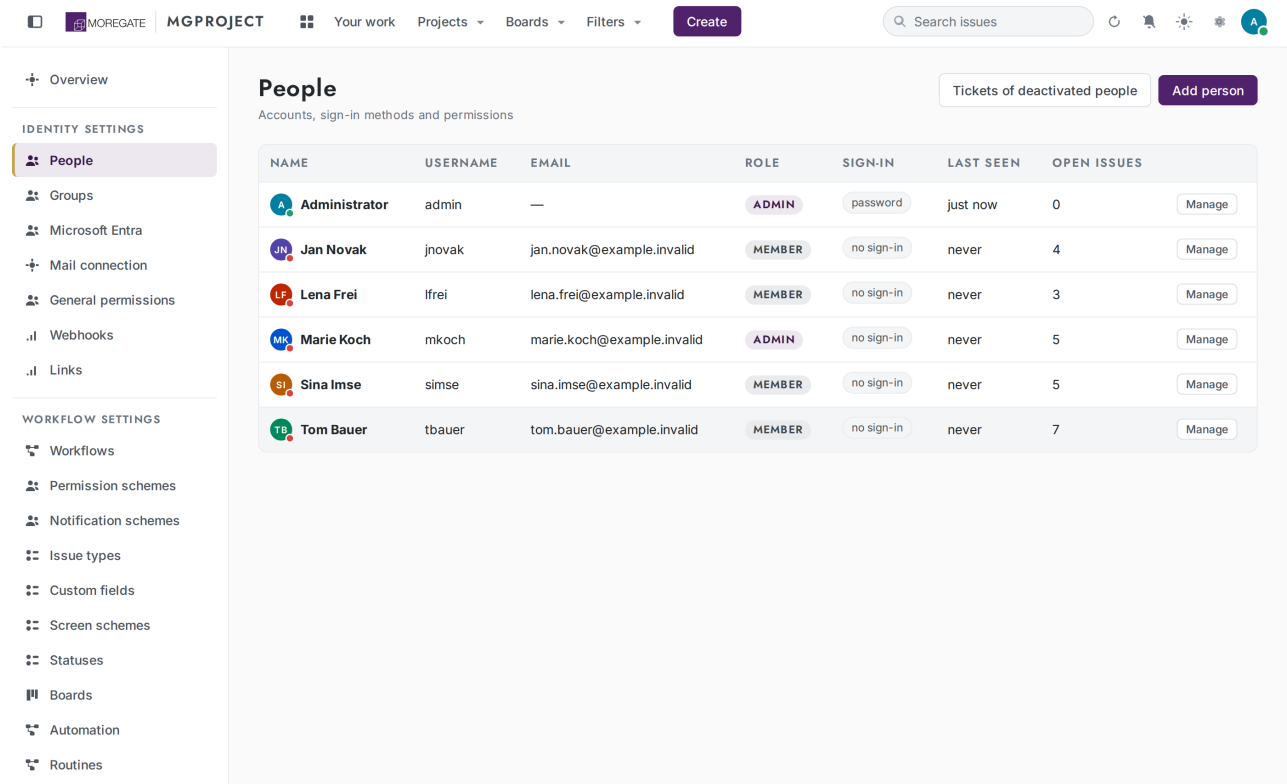
> Counted are all active accounts, whether they sign in with a password or through Entra ID. Deactivated accounts do not count.

05

Accounts, groups and signing in



The sign-in page. Passkeys and Entra ID can be set up beside the password.



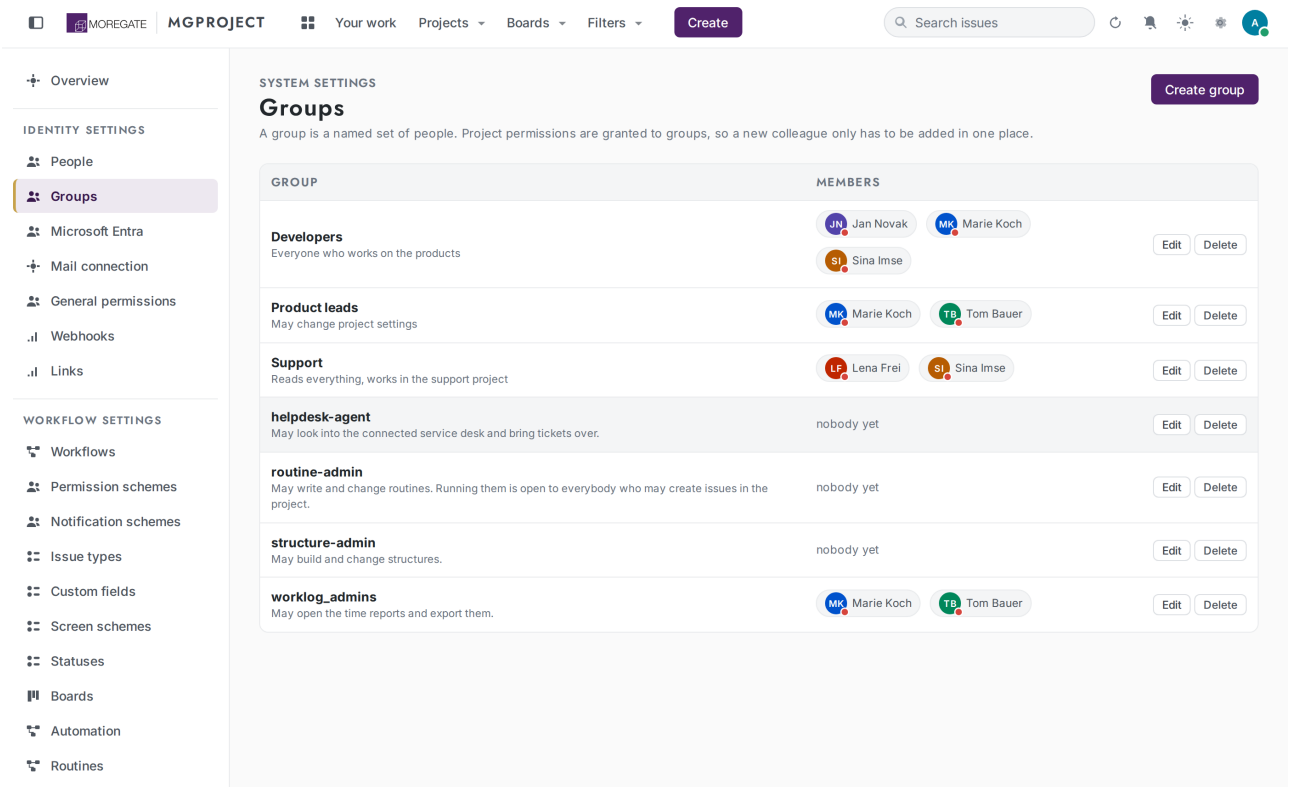
System settings People: create accounts, reset passwords, grant roles, deactivate accounts.

The first password comes from the deployment and has to be changed at the first sign-in. After that everybody changes their own password and sets up a passkey under “Account and passkeys” - fingerprint, face or a security key.

Deactivated people

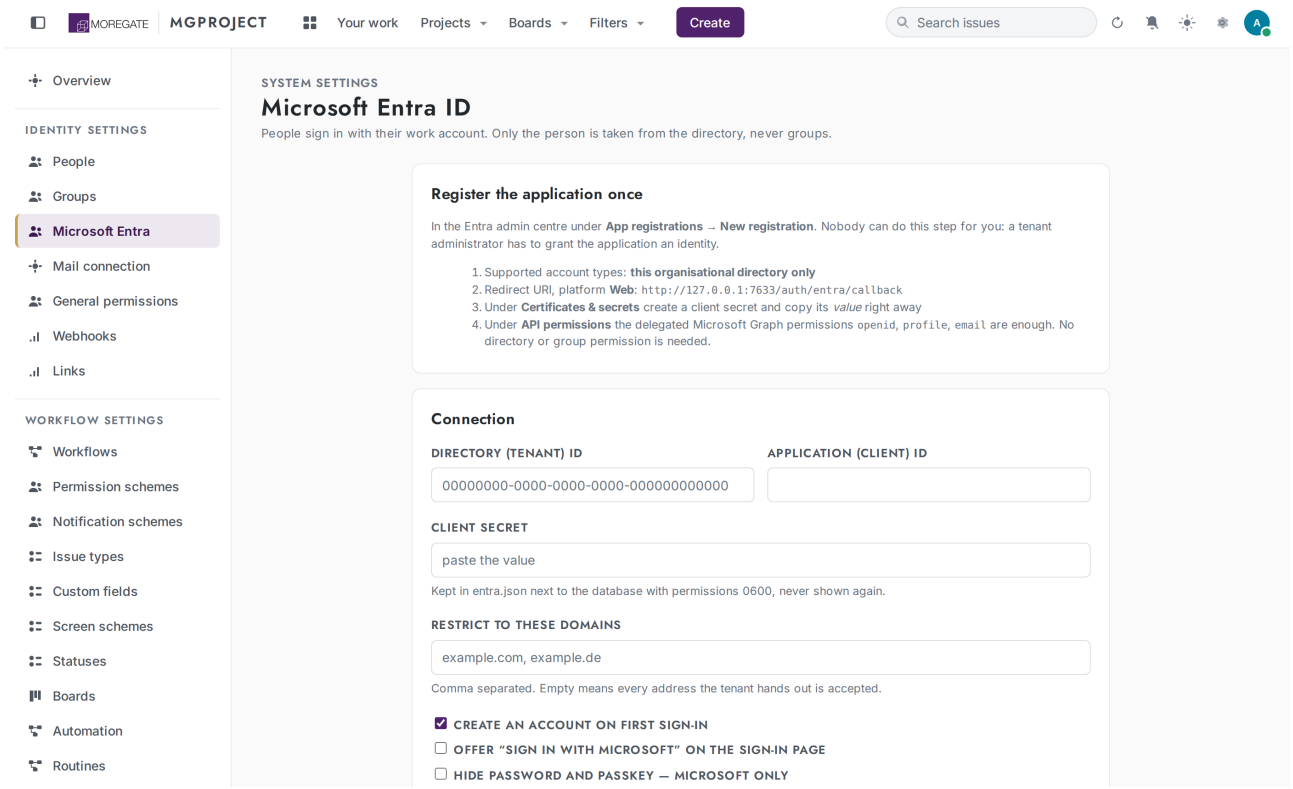
A deactivated account - by hand or by the directory sync - is marked wherever its name appears and no longer offered in lists. A field that still names such a person keeps showing them instead of “Unassigned”.

Under People a button leads to a page listing every issue of a deactivated person; they can be handed to somebody else one at a time or in a batch.



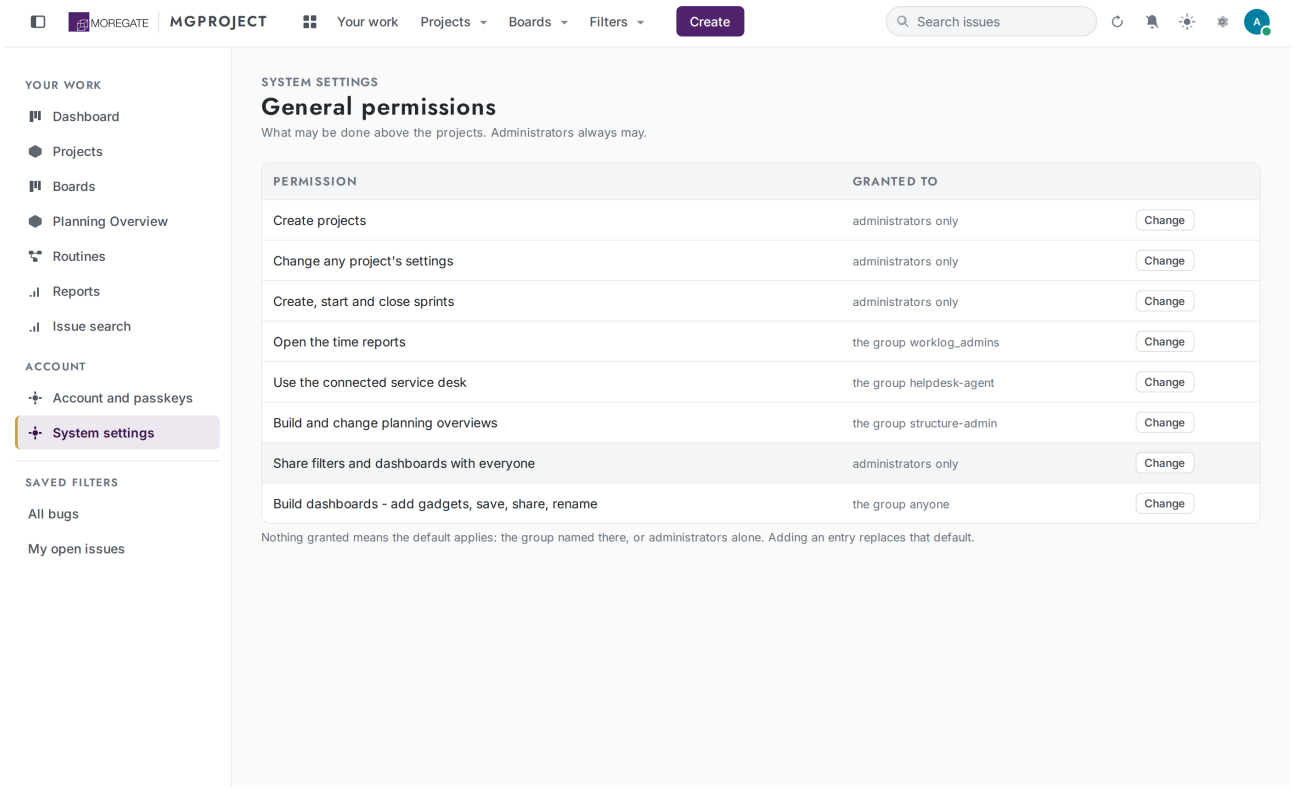
System settings Groups: permissions for several people at once.

Group	What it opens
worklog_admins	The time reports, and other people's bookings
helpdesk-agent	The connected service desk
structure-admin	Building and changing planning overviews
routine-admin	Writing and editing routines



System settings Microsoft Entra: signing in with the work account.

You need the directory ID, the application ID, a secret and the redirect URI printed on this page - copy it, do not type it. Accounts appear on first sign-in; no groups are taken from the directory.

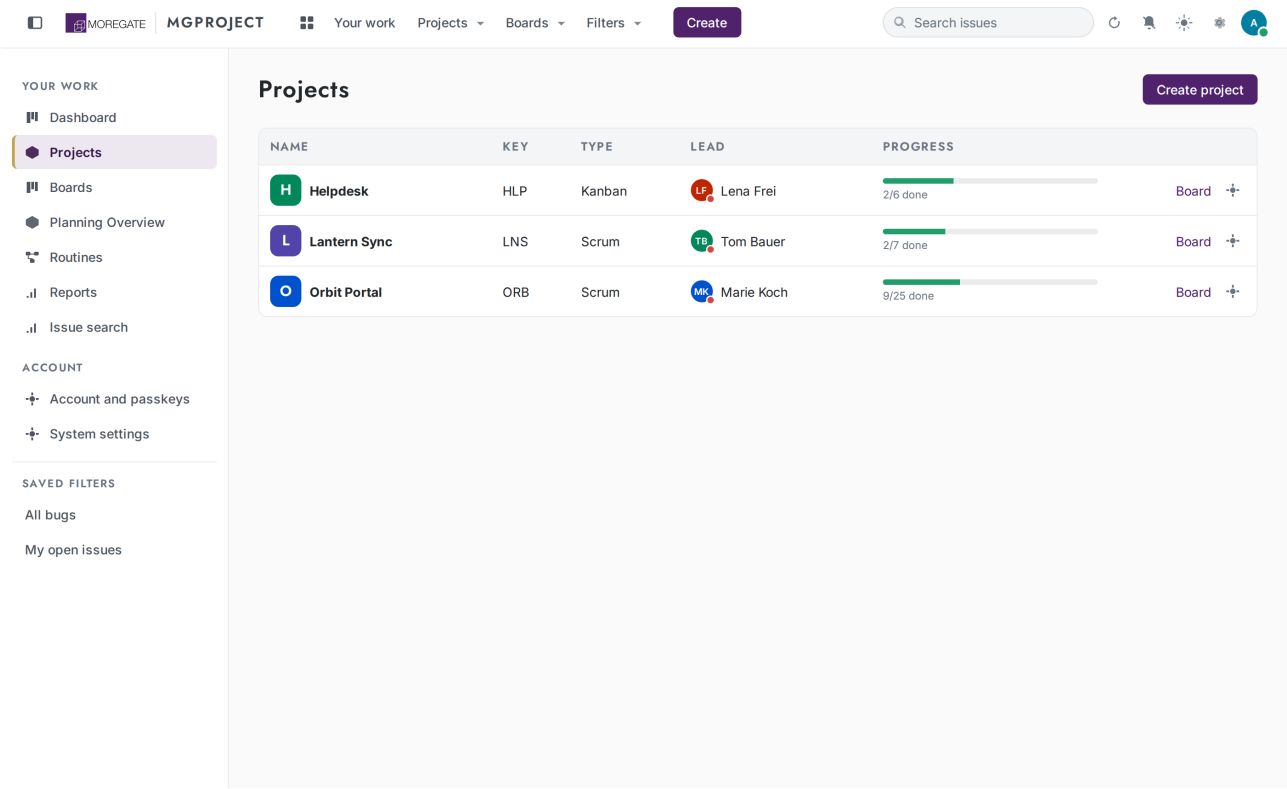


System settings General permissions: what is allowed above the projects.

This says who may create projects, who may start sprints and who sees the time reports. Everything else is granted per project.

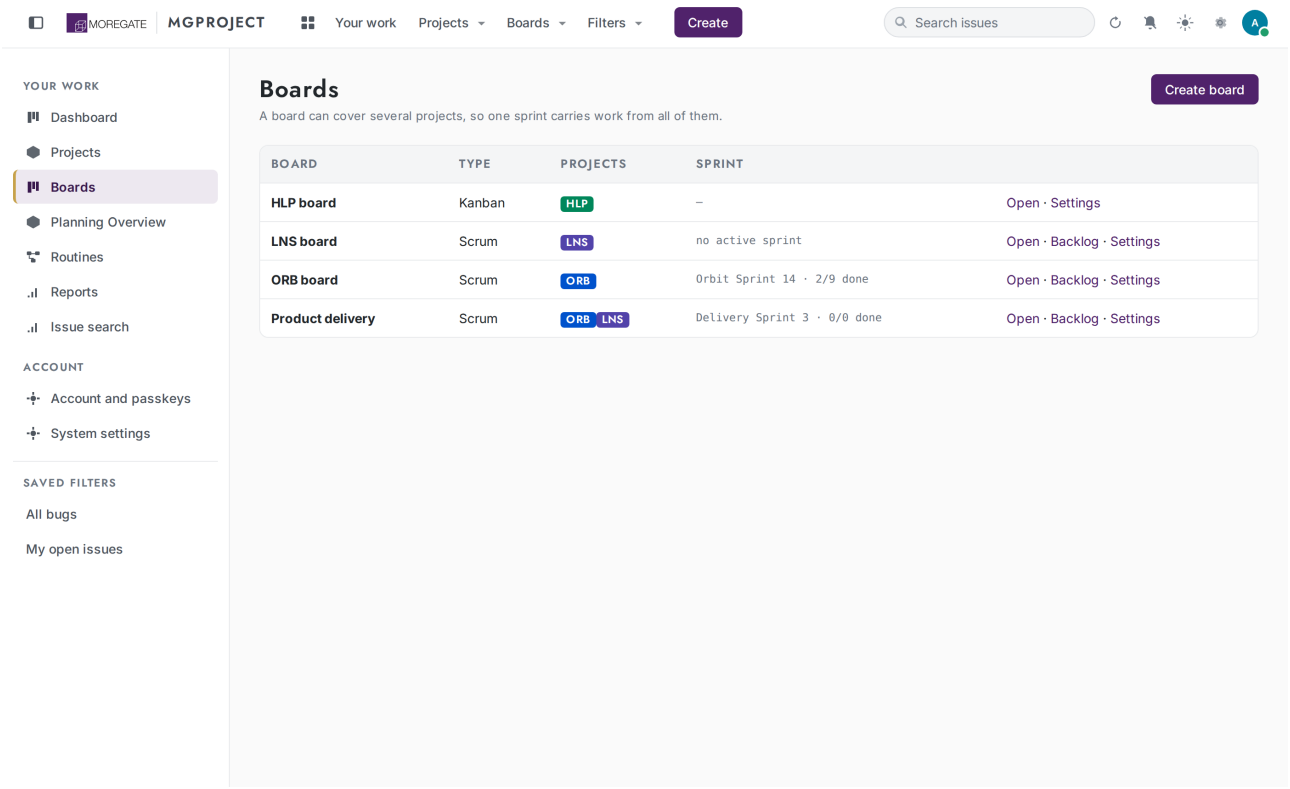
06

Projects and boards



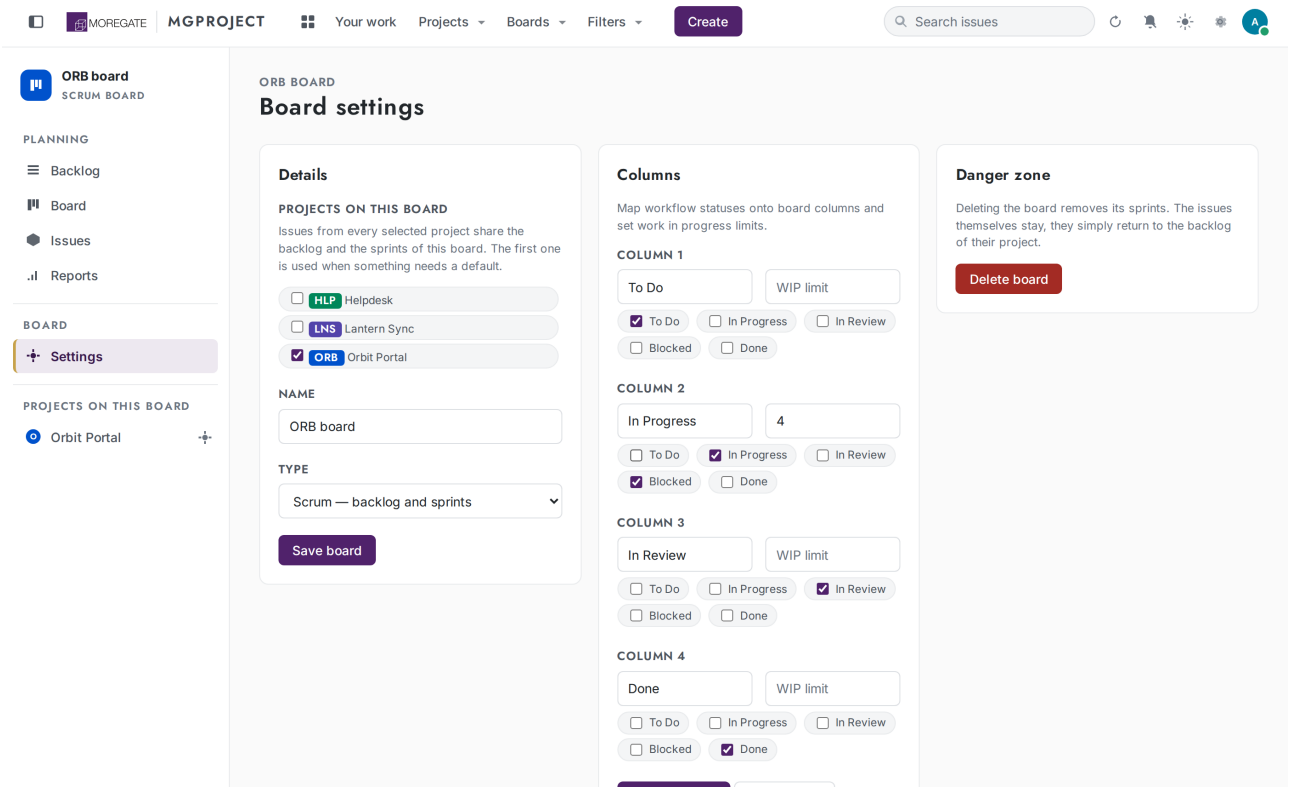
The projects, with their key and lead.

A project holds issues and carries a key - the letters in front of every issue number. Choose it carefully: it is what people type, say and search for, and it does not change afterwards.

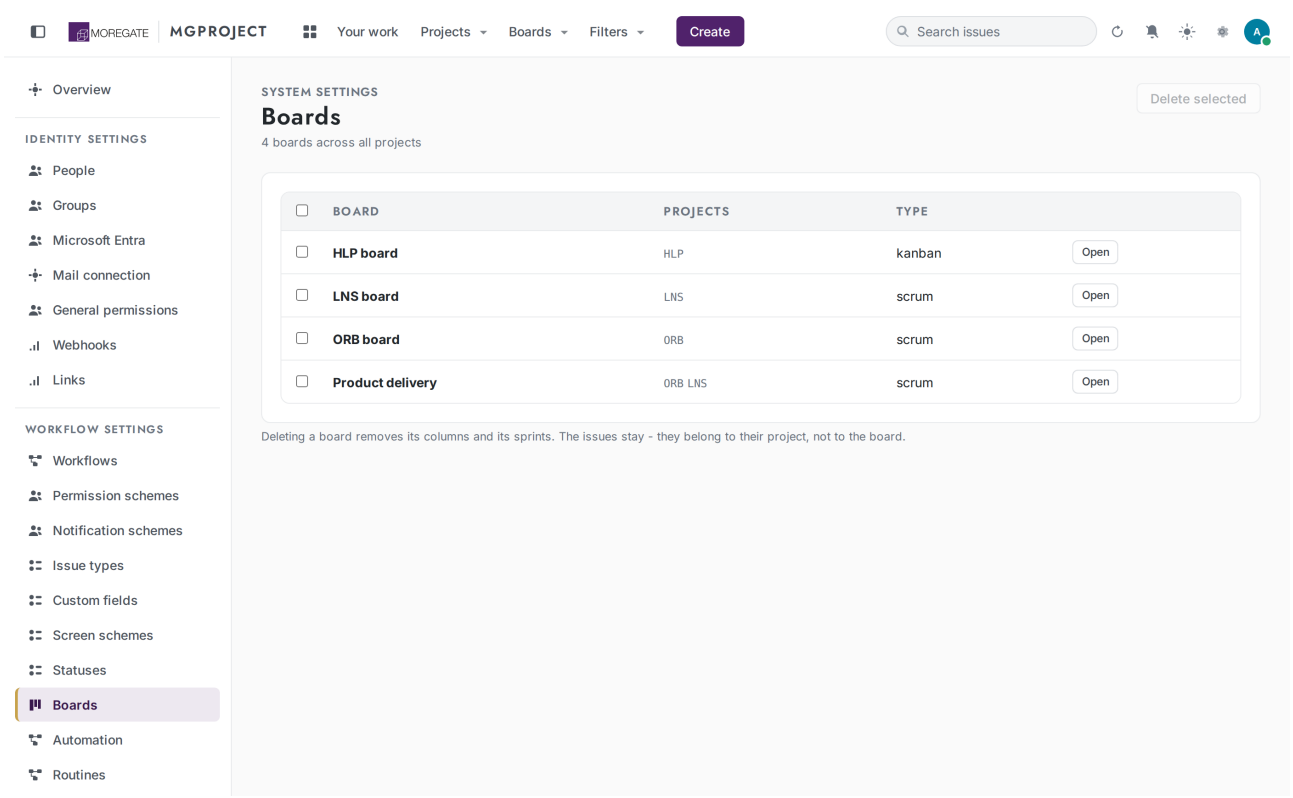


The boards, each with its projects.

A board carries one or several projects. A new project usually belongs on a board that is already running: either from the project, under Project settings › Boards › Add to an existing board, or from the board, under Board settings › Projects on this board.



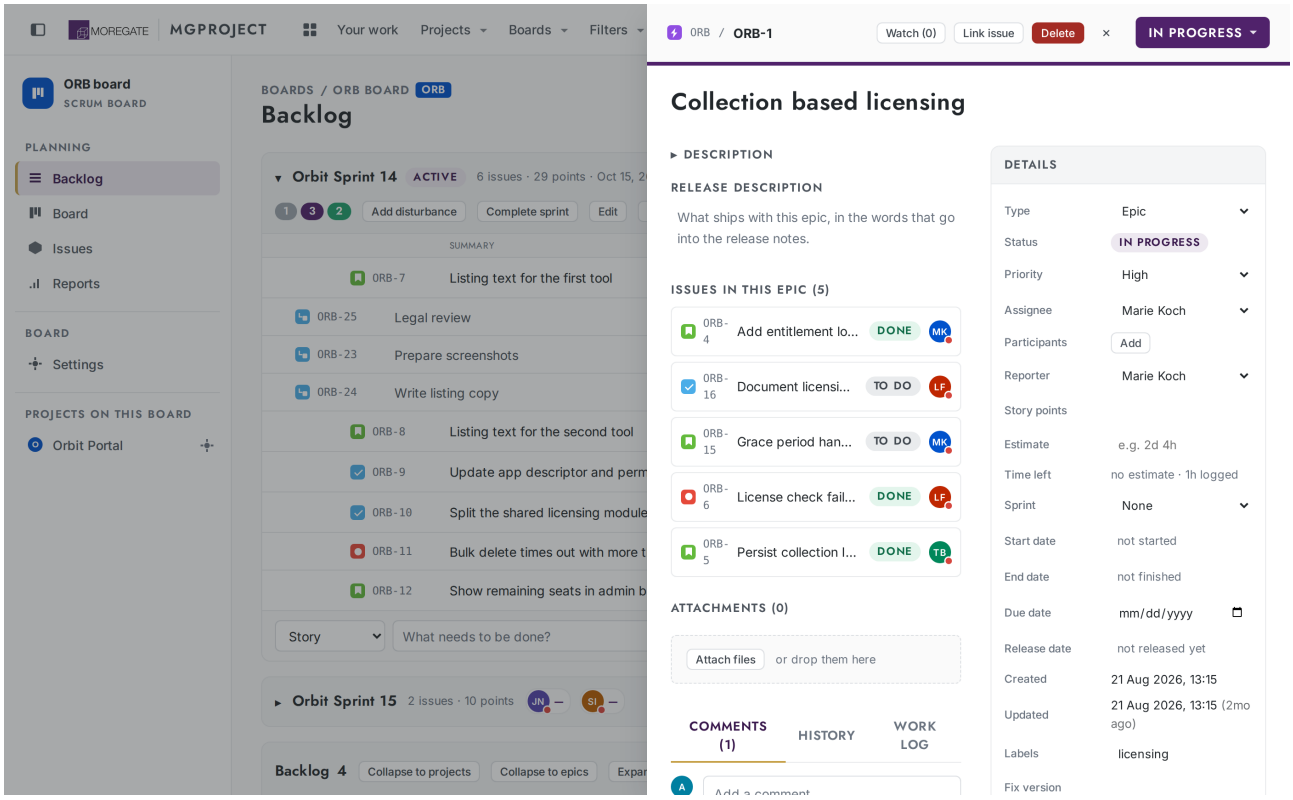
Board settings: columns, projects, and what is shown in the sprint.



System settings Boards: every board of the instance at a glance.

07

Working with issues



An issue: description and comments on the left, the fields on the right.

An issue has a type, a status, a priority and somebody it is assigned to. Summary and description are edited in place - click, type, click away. Web addresses become links once saved.

An issue opens over the page it was opened from. The arrow in the corner leads back through issues opened from one another; the cross closes. Ctrl- or middle-click opens it in a tab of its own on purpose.

Created and Updated are fields of their own on the right; every comment carries its date and time. Screenshots paste straight into a comment or the description and can be resized by dragging a corner. Typing @ in a comment offers the people who may see the project - whoever is named is told.

Assignee and participants

The assignee leads the issue and is the one who moves its status. Participants help: as many people as needed, chosen from those who may work in that project. They book their own time on the issue and find it under “My work in the sprint” - the status stays the assignee's to move. In MQL they are found with participant = currentUser().

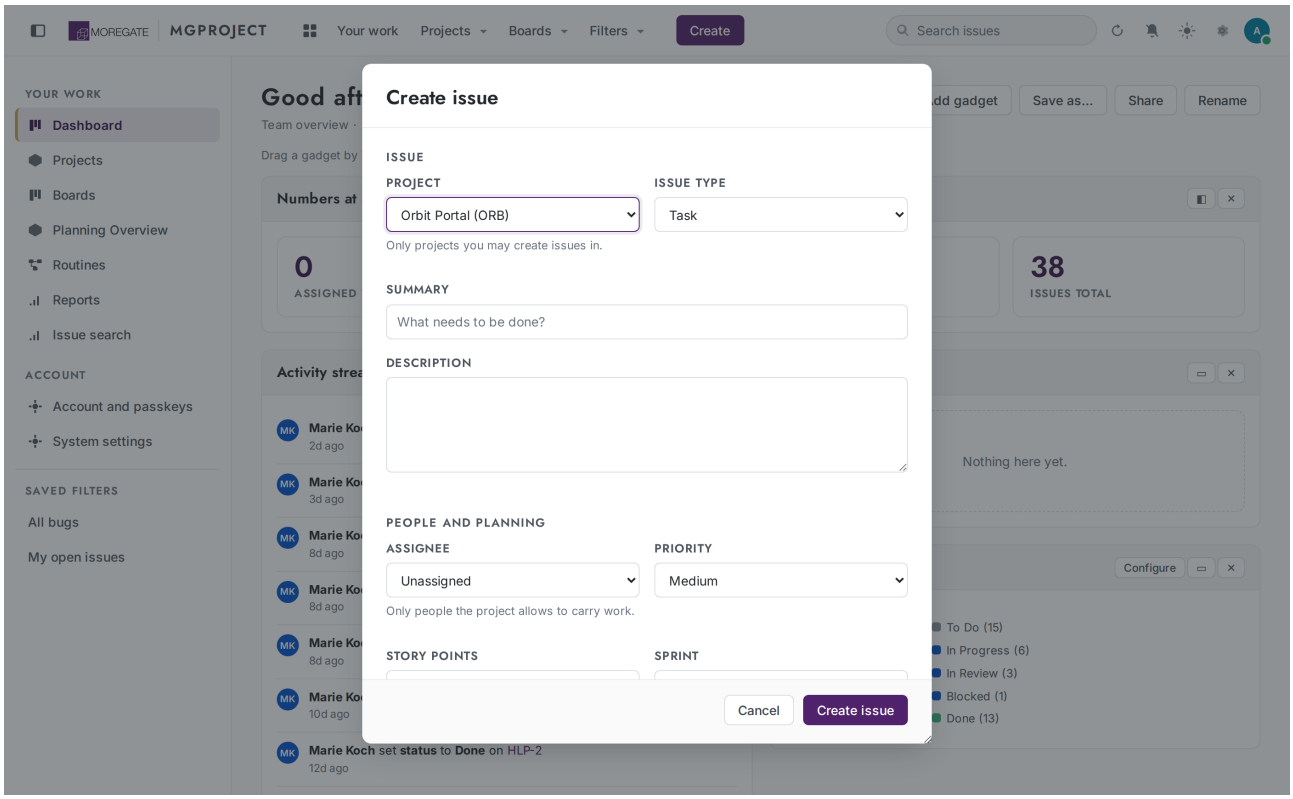
Comments, history and attachments

Three tabs sit under the description: comments, history and work log. The history names every change with its author and time. Attachments hang on the issue and can be dropped in.

Time

Time is logged on the issue, in the shape it is spoken: 3h 30m, 2d, 45m. Own bookings are always visible; other people's are for the worklog_admins group.

Creating one

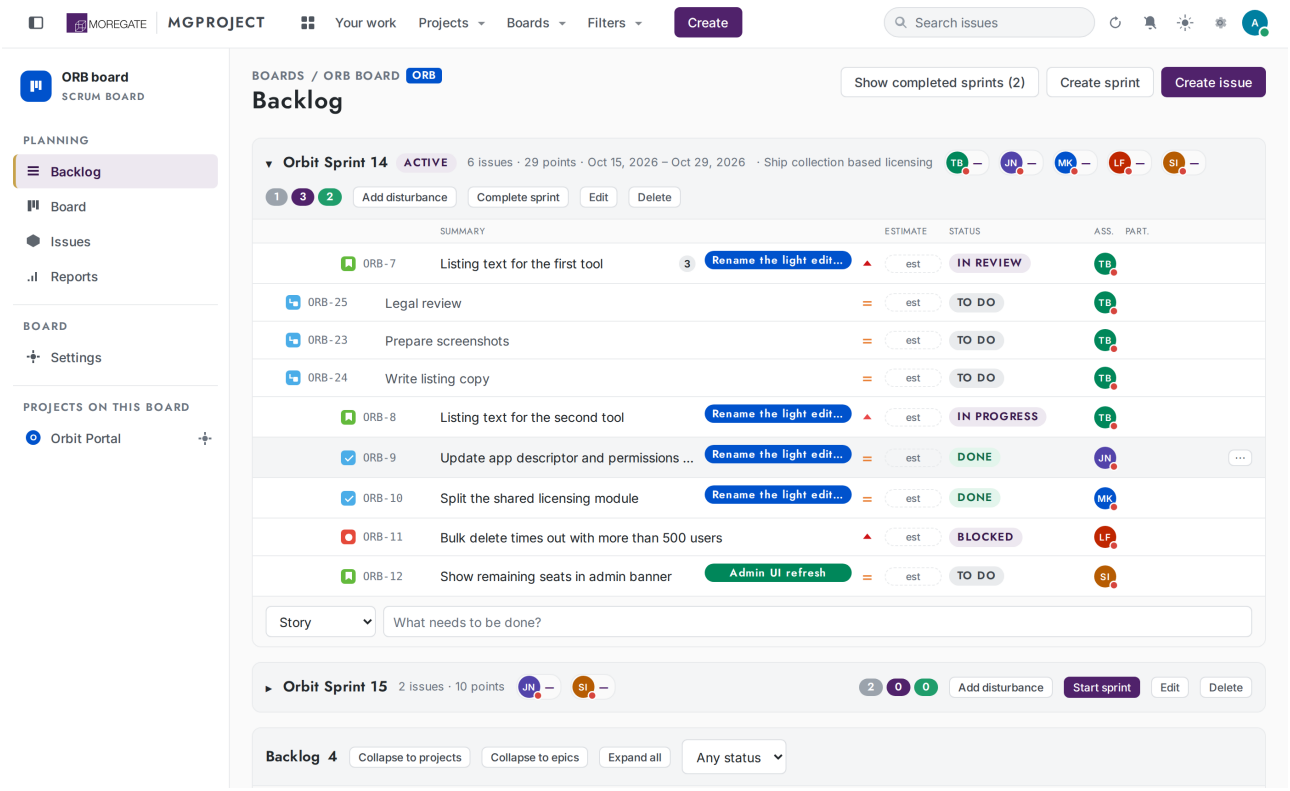


The create dialogue: project, type and the fields of the screen.

Which fields appear is decided by the screen for that issue type in that project. Sub-tasks sit under an issue, epics above it.

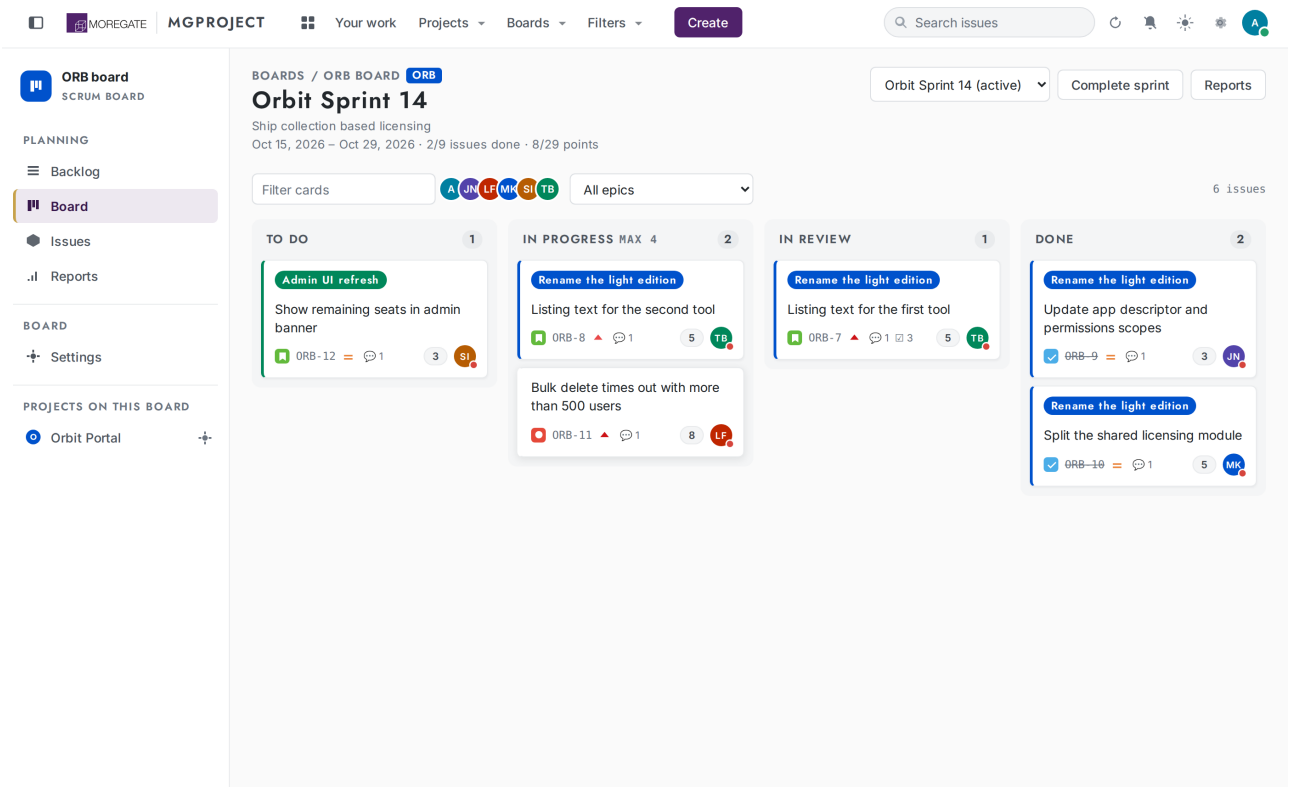
08

Boards, backlog and sprints



The backlog with the running sprint above it.

The backlog lists what is not yet in a sprint. Drag an issue into the sprint above it, or several at once. A new sprint runs one working week by default; the dates can be changed when it is created.



The board: one column per status, moved by dragging.

Completing a sprint

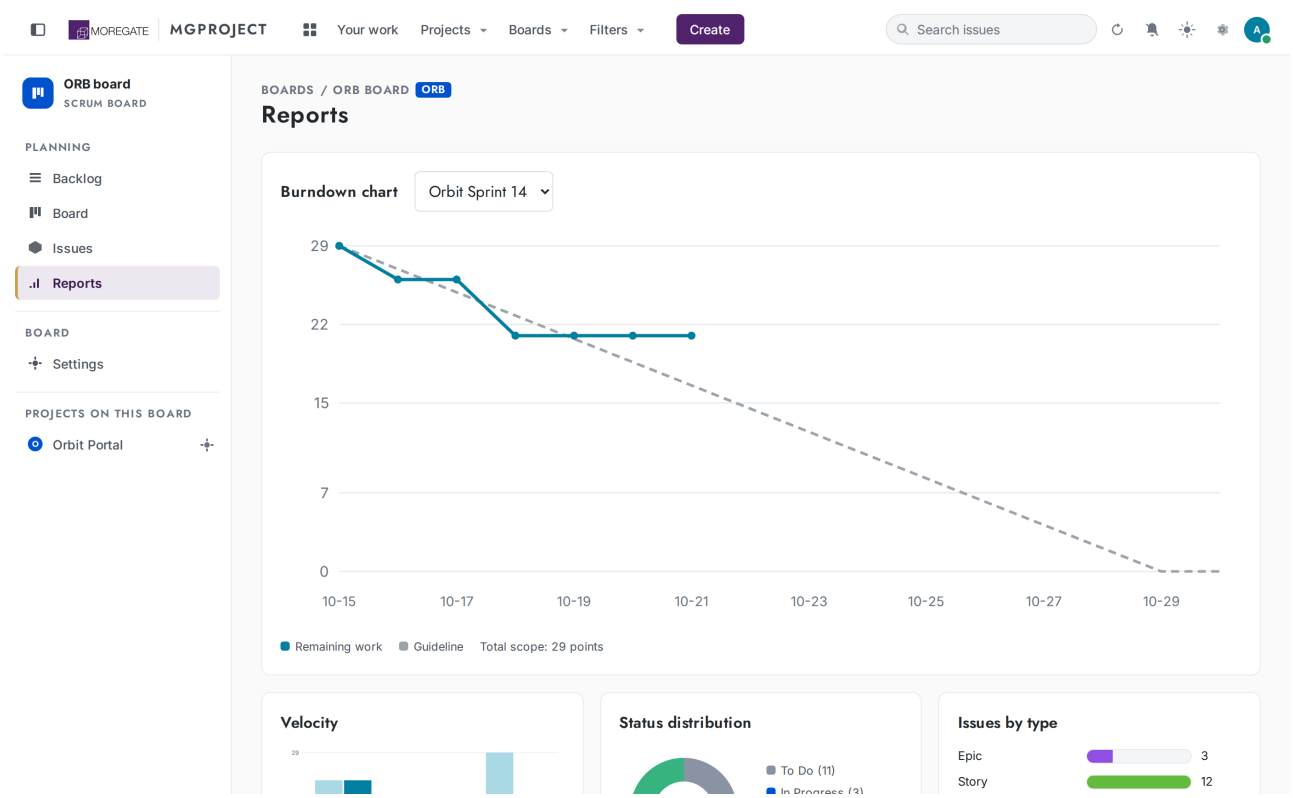
Completing one asks what should happen to whatever is unfinished: into the next sprint, back to the backlog, or split - the work already done stays on the old issue and the remainder moves on as a new one, keeping its service desk ticket.

Filtering by person

A click on the avatars in a sprint's header shows only those people's issues; a second click takes the filter back. Several people can be chosen at once.

Disturbances

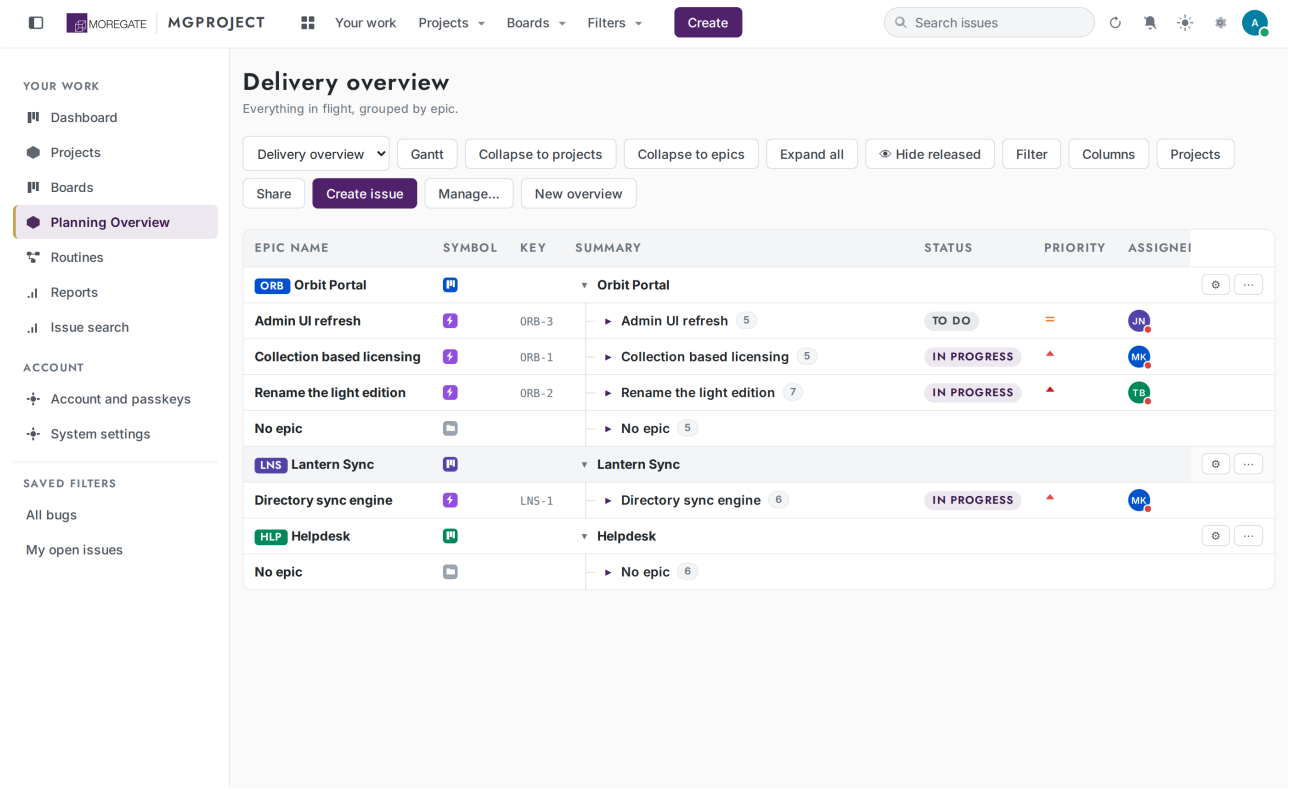
Whatever arrives unplanned is taken into the sprint as a disturbance rather than quietly spoiling the plan. The sprint report then says how much of it came in afterwards - the number that explains why a sprint did not add up.



Board Reports: burndown, velocity, status distribution and types.

09

Planning Overview



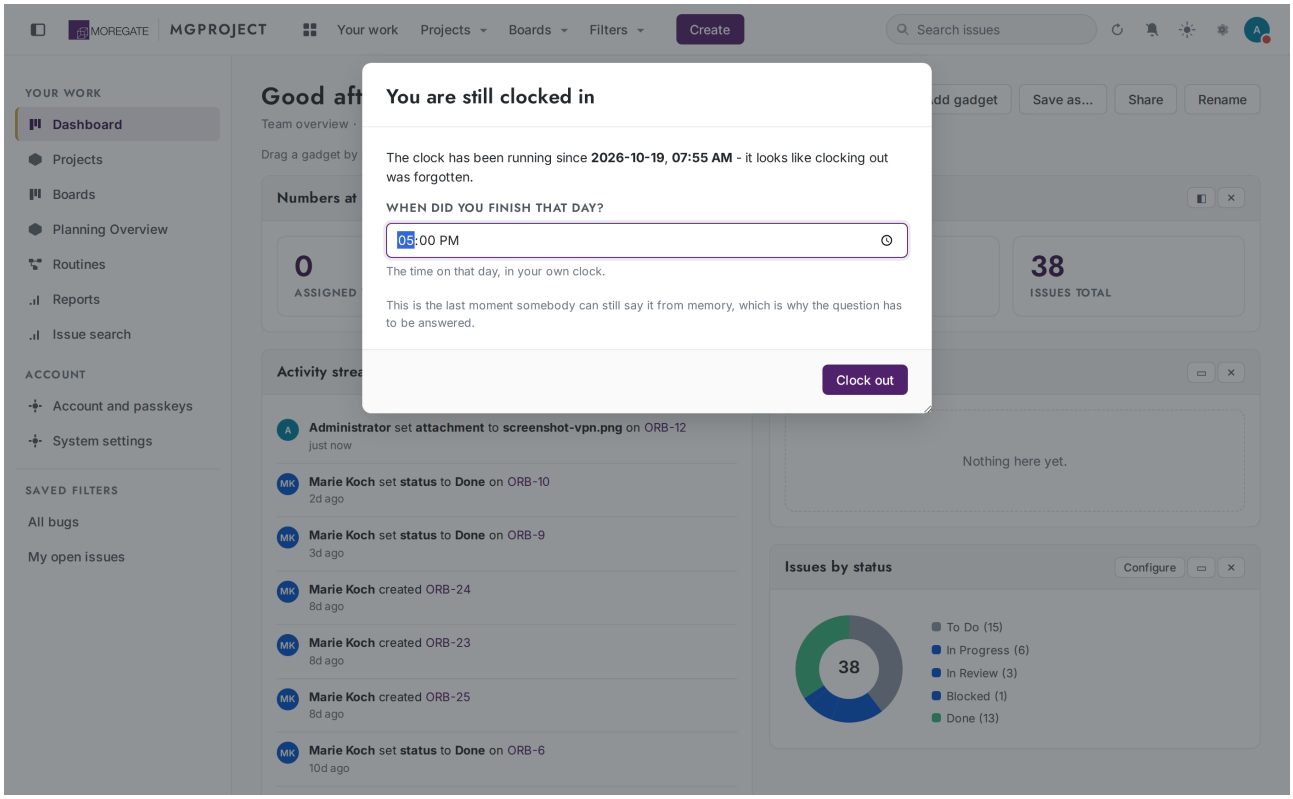
Epics with their issues underneath, in columns you choose.

The Planning Overview shows what sits under what: epics, their issues, sub-tasks below those. Choose the columns under “Columns”. A click on a heading sorts by it, a second reverses, a third returns to the original order - and the tree is kept, so an issue never leaves its epic.

Several overviews can be kept side by side, shared, and seen as a Gantt chart. Who may build them is decided by the structure-admin group.

10

Time recording

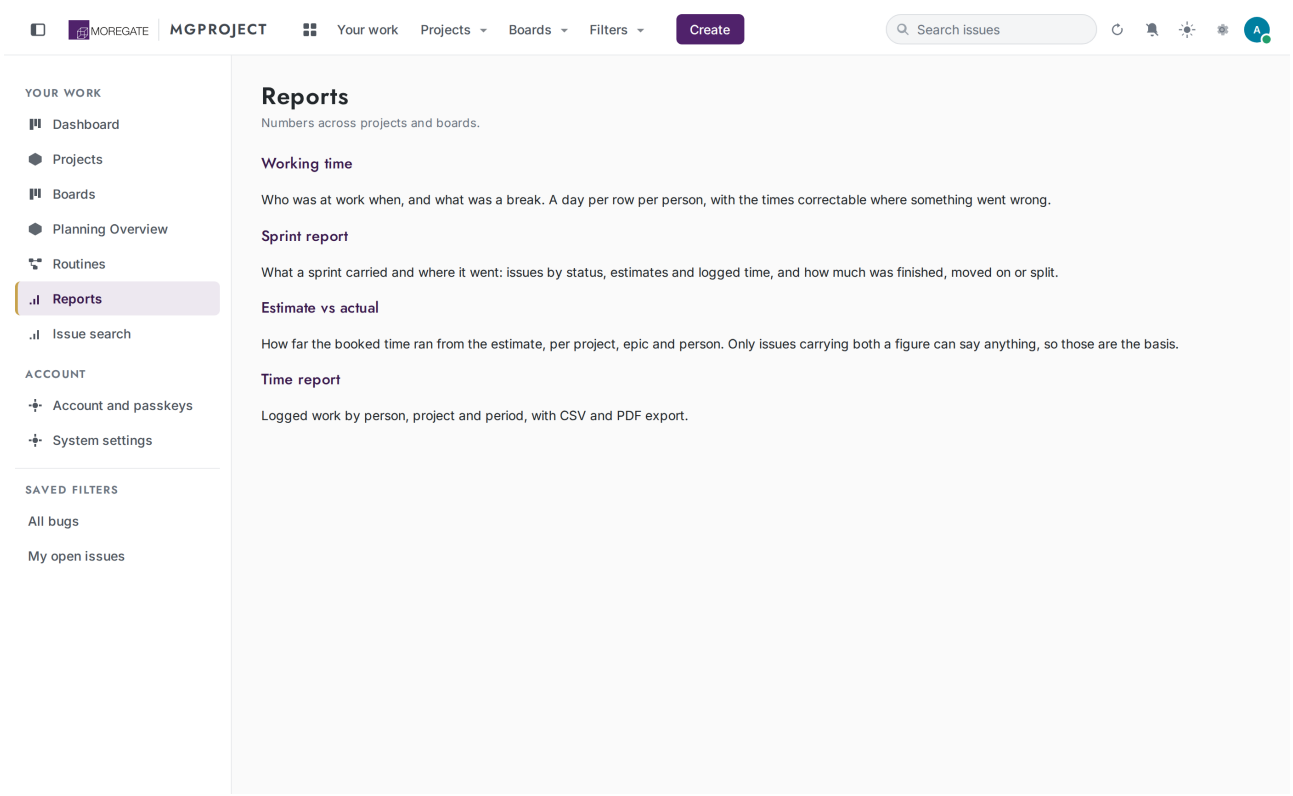


The first question at sign-in when yesterday's clock is still running.

Beside booking time on issues there is a clock: in, break, out. The “Who is at work” gadget shows everybody's state, and a small dot on each avatar says it throughout the interface.

Where an account is set for it, two questions appear at sign-in, and both have to be answered. If the clock is still running from an earlier day, the first asks when that day ended - the next morning is the last moment somebody can still say it from memory. The second asks whether the clock should start now.

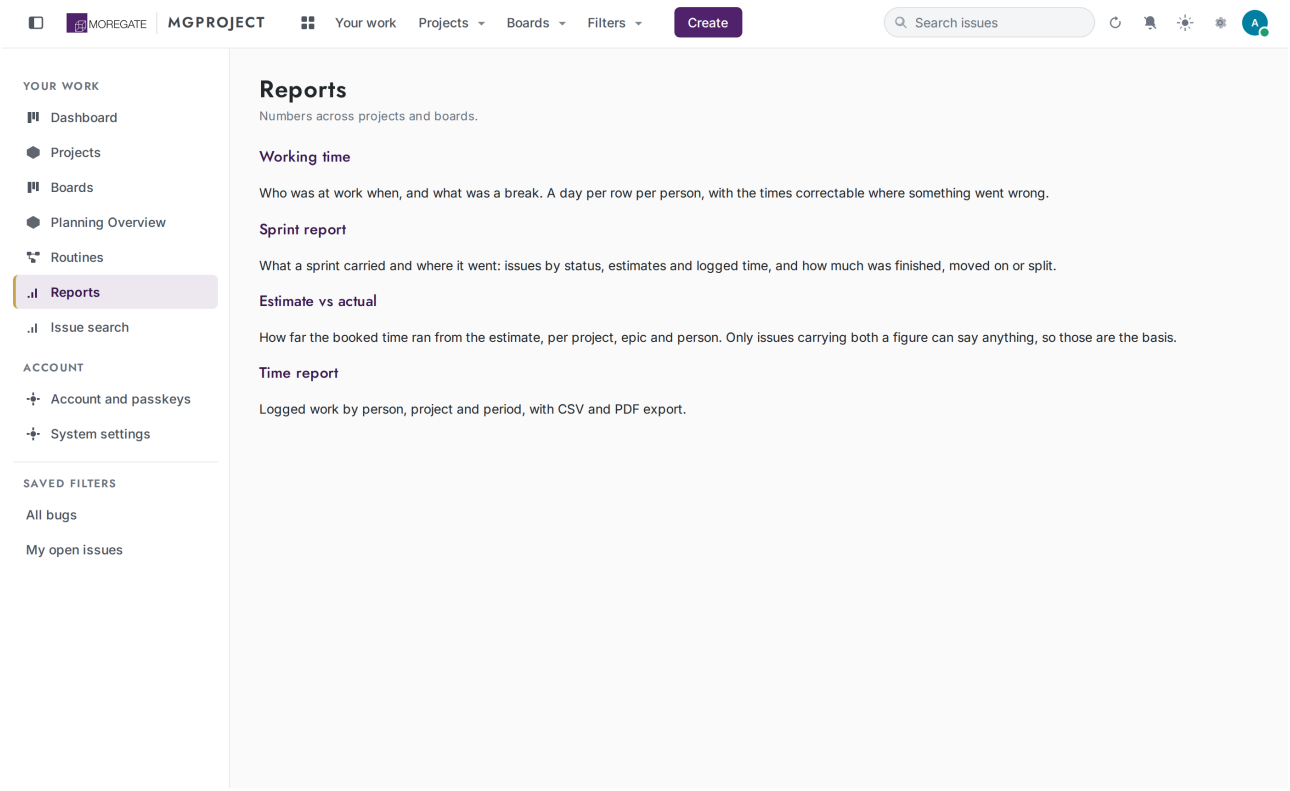
> Because neither can be dismissed, every booking is one a person made. Nothing is closed automatically and nothing has to be marked as estimated.



The time report: hours by person, project and period.

11

Reports

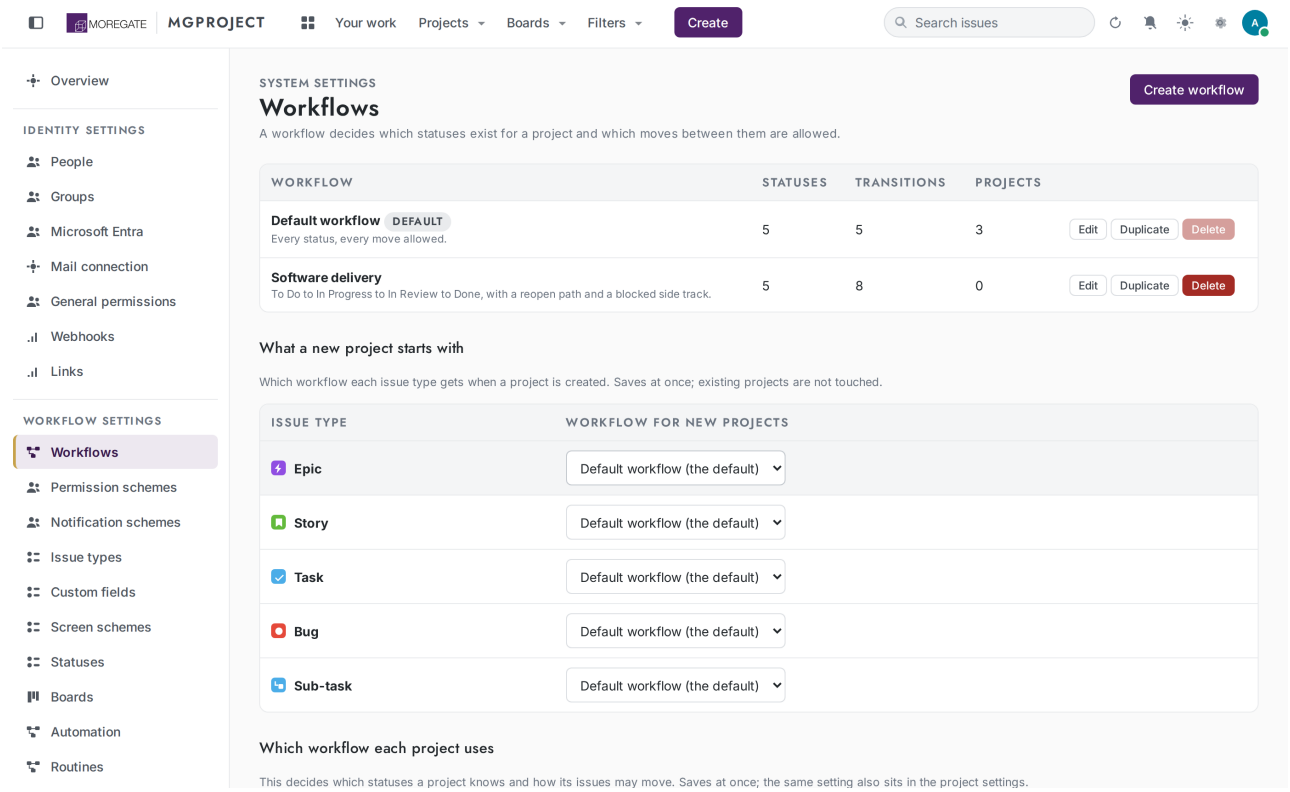


The reports, each with a PDF export.

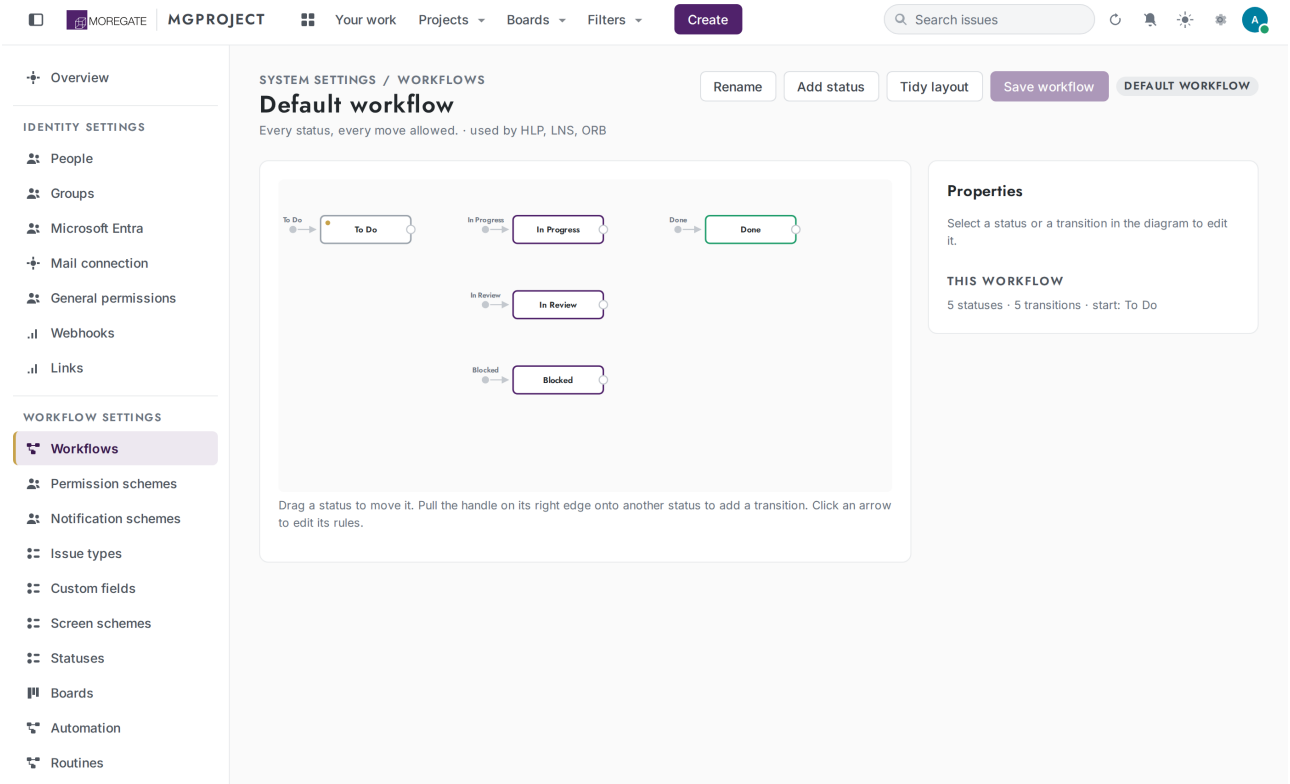
- Sprint report - what a sprint carried and where it went: issues by status, estimates against logged time, and how much was finished, moved on or split
- Estimate vs actual - estimates against time booked, per person, per project and per epic. Only issues carrying both figures are the basis
- Working time - who was at work when, with an Excel export; a day per row per person, correctable
- Time booked - hours on issues, by month, by day and by sprint, with CSV and PDF export

12

Workflows and statuses

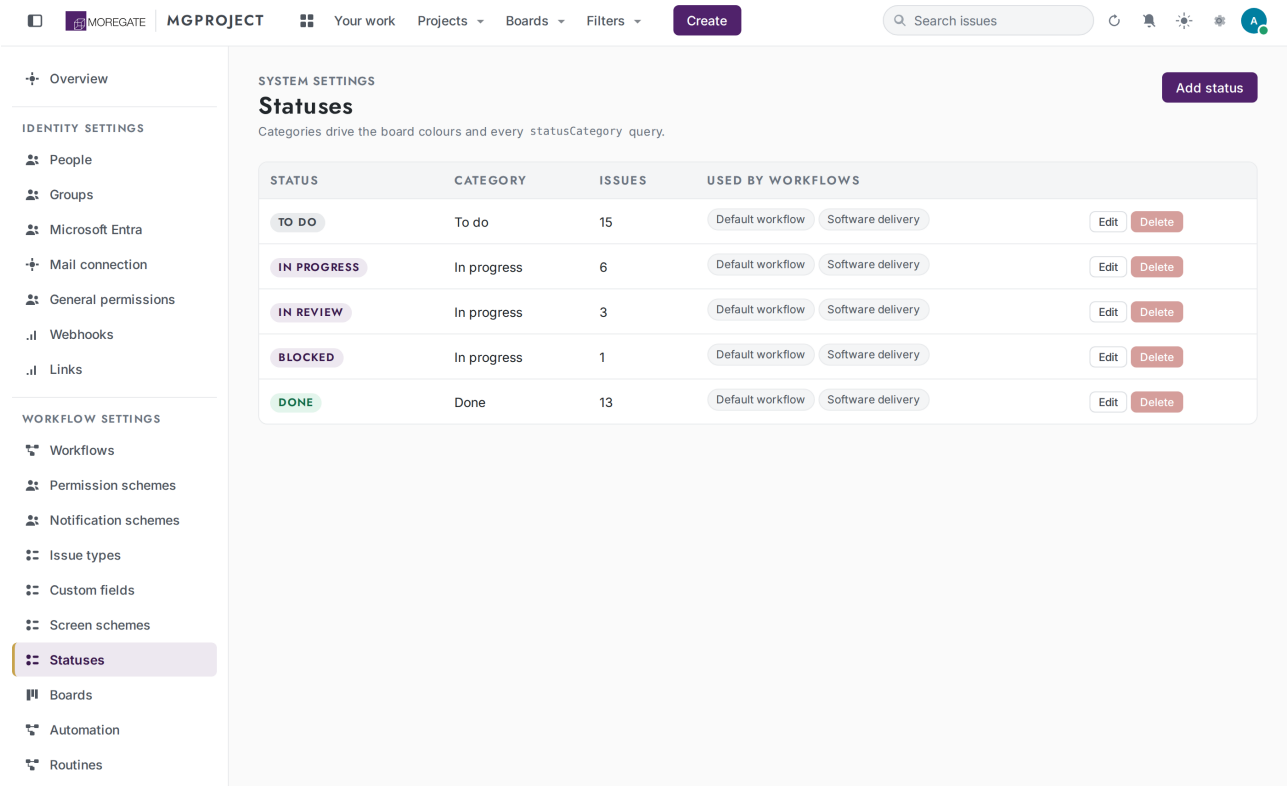


System settings Workflows: which statuses exist and which moves are allowed.



The editor: statuses, transitions, and who may take them.

A workflow decides which statuses a project knows and which transitions are allowed. For each transition you can say who may take it and what should happen along the way. The overview also says what a new project starts with - one workflow per issue type - and which workflow each existing project uses.

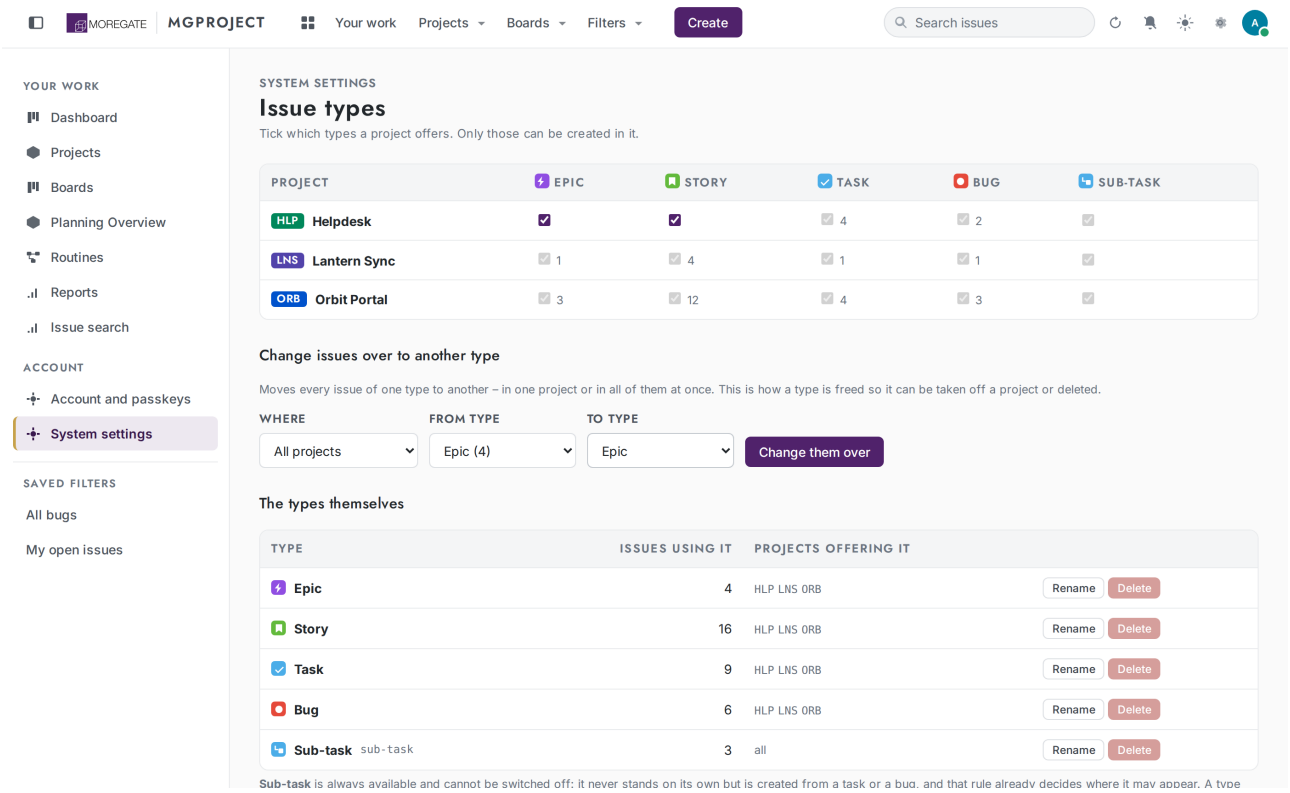


System settings Statuses: name, colour and category.

A status belongs to one of three categories - to do, in progress, done. The board, the reports and `statusCategory` in MQL hang off that, which makes the category more important than the name.

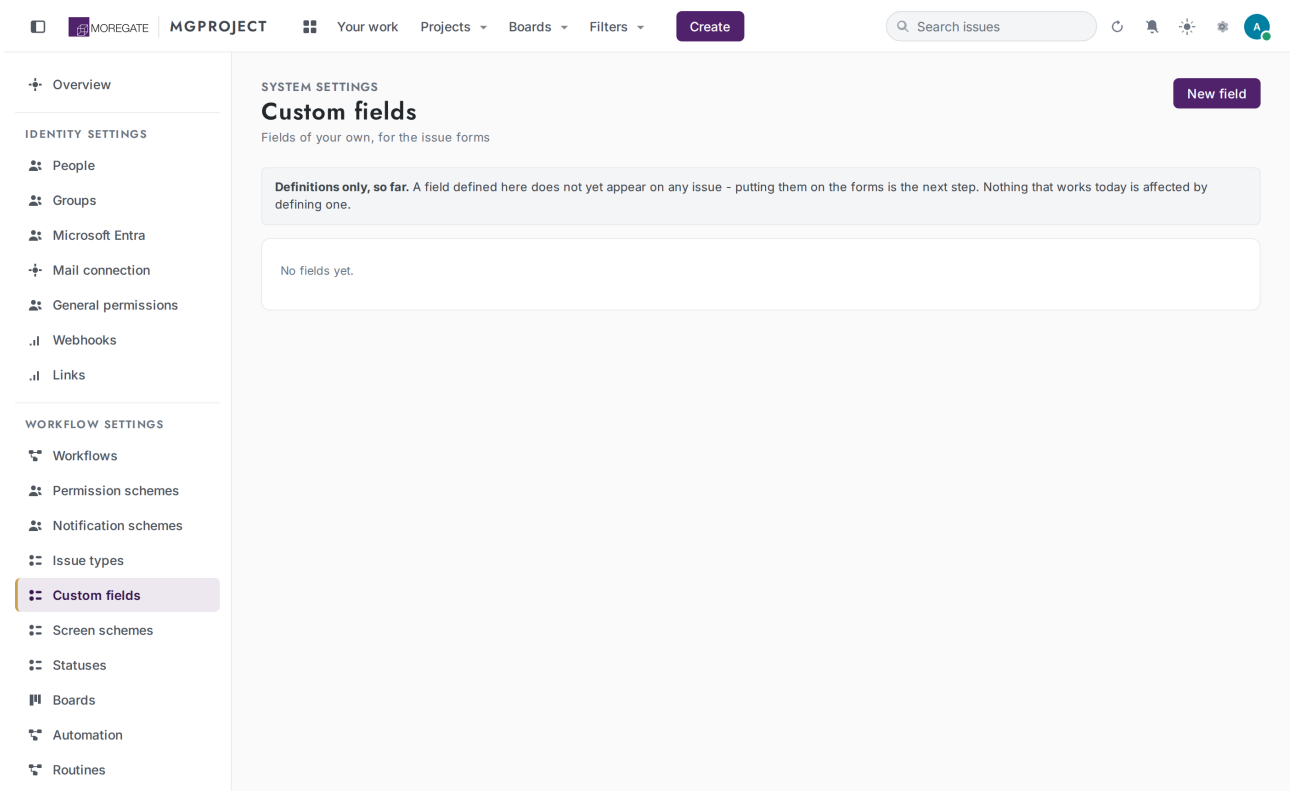
13

Issue types, fields and screens



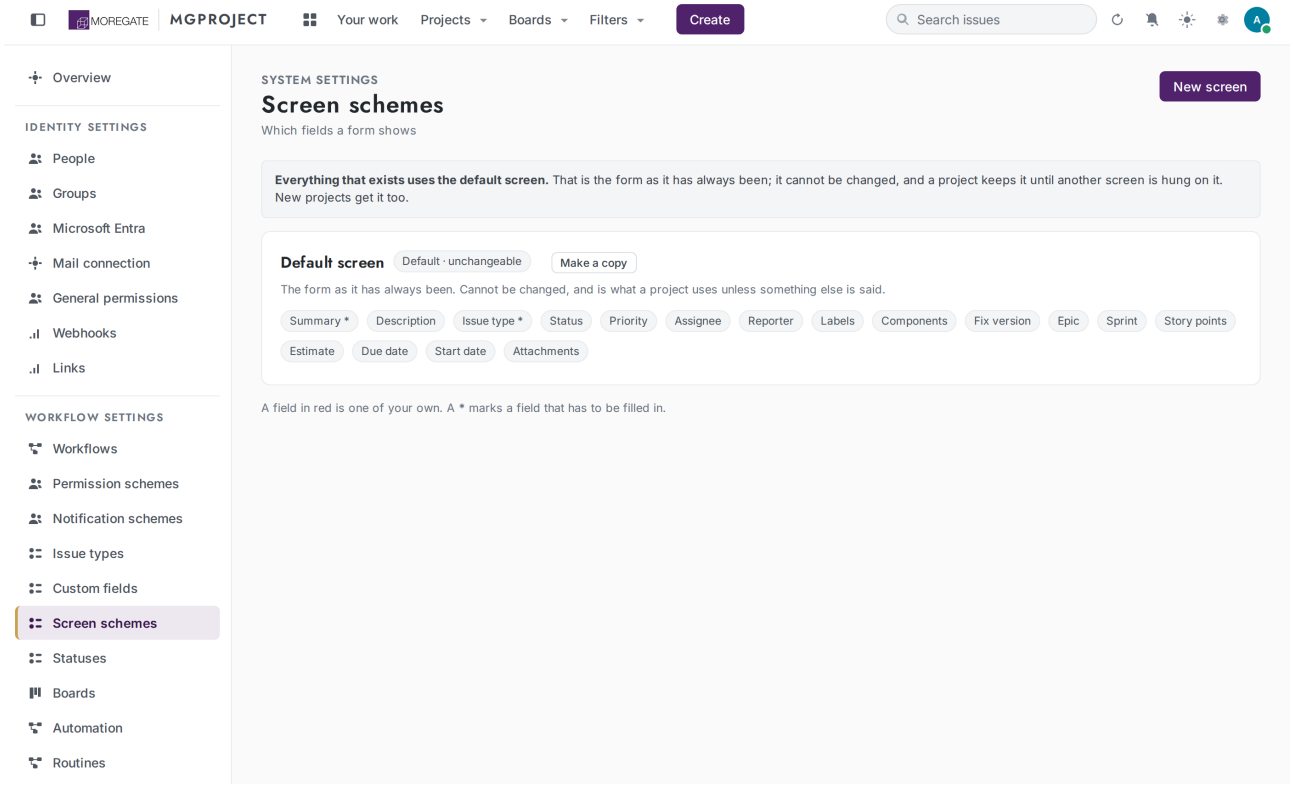
System settings Issue types: which types a project offers.

Every type carries an icon and a colour that come back everywhere. Sub-tasks are always available; the other types are ticked per project.



System settings Custom fields: fields of your own and their kind.

Custom fields come as text, number, date, select, multi-select, checkbox and person. They can be placed on screens, taken into lists as a column, and searched in MQL.

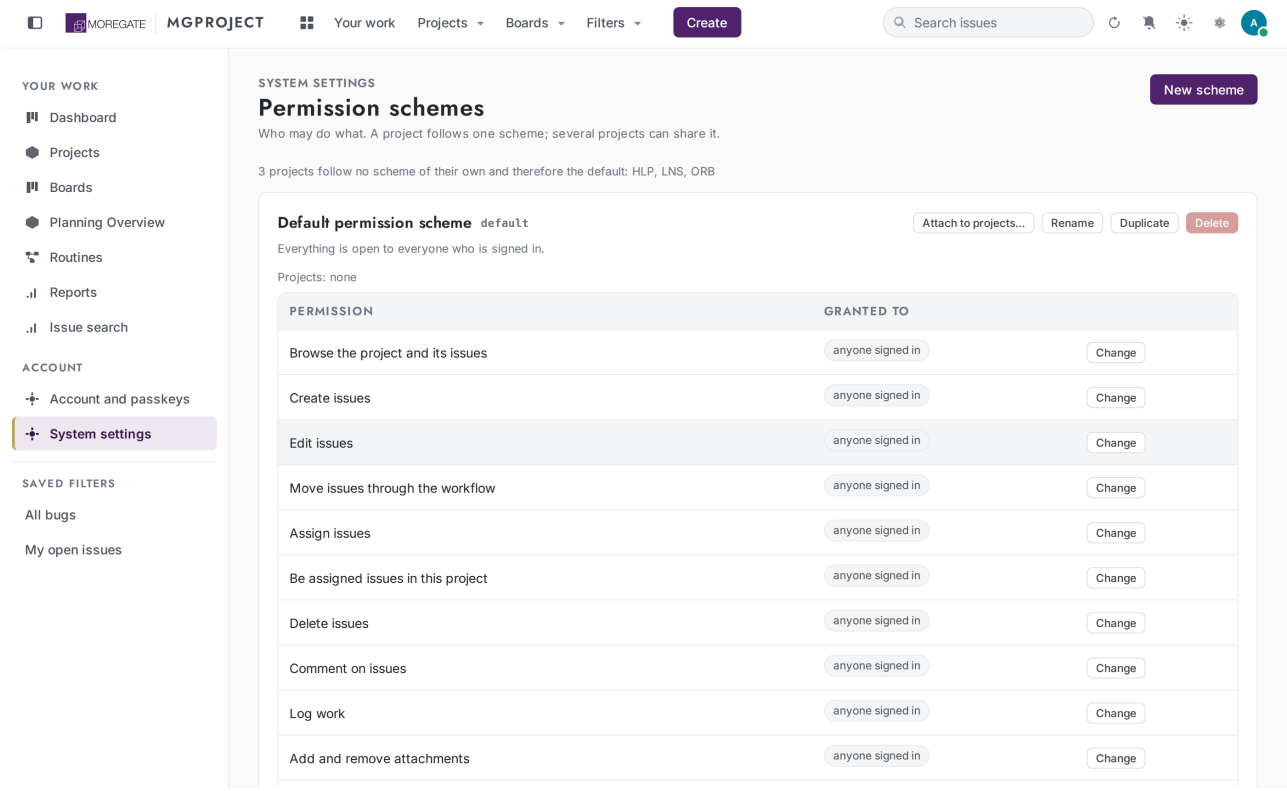


System settings Screen schemes: which fields appear when.

A screen decides which fields are there when creating, when editing and when viewing - per issue type.

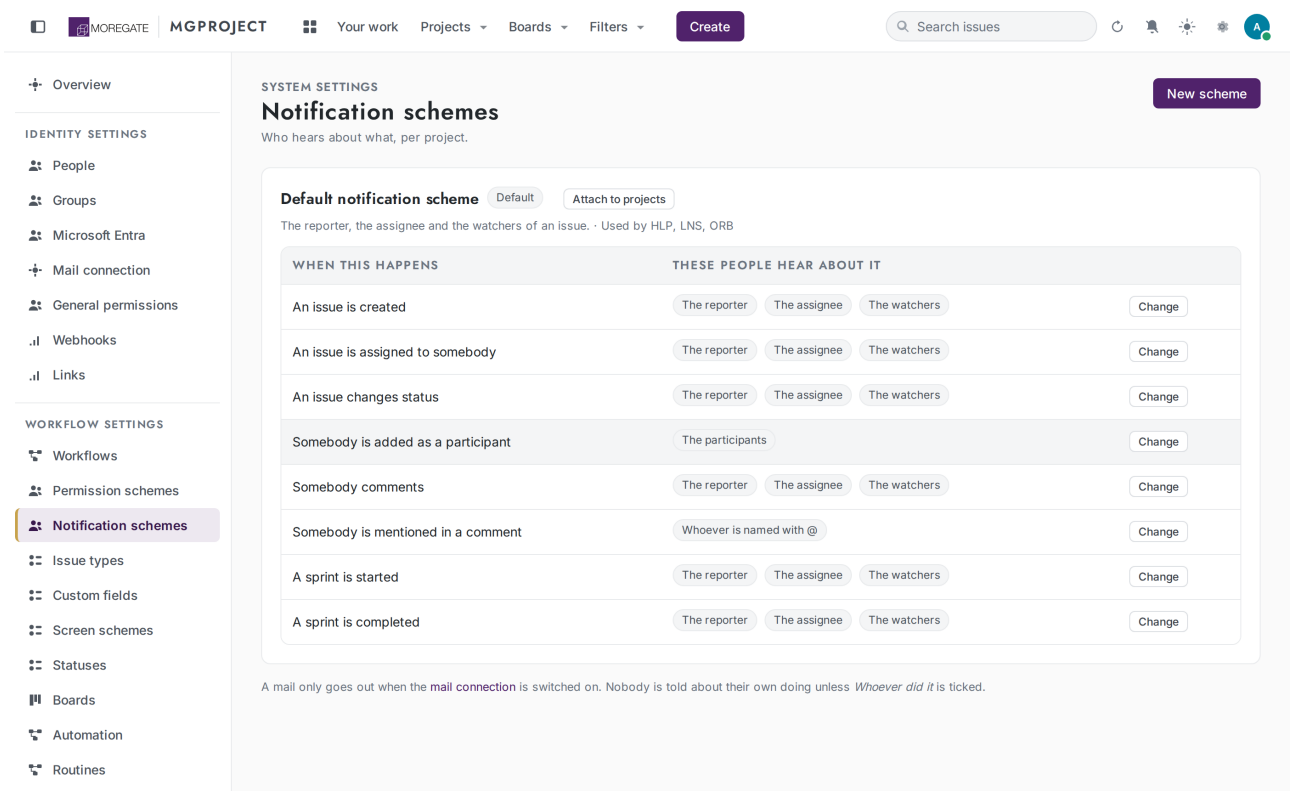
14

Permissions and notifications



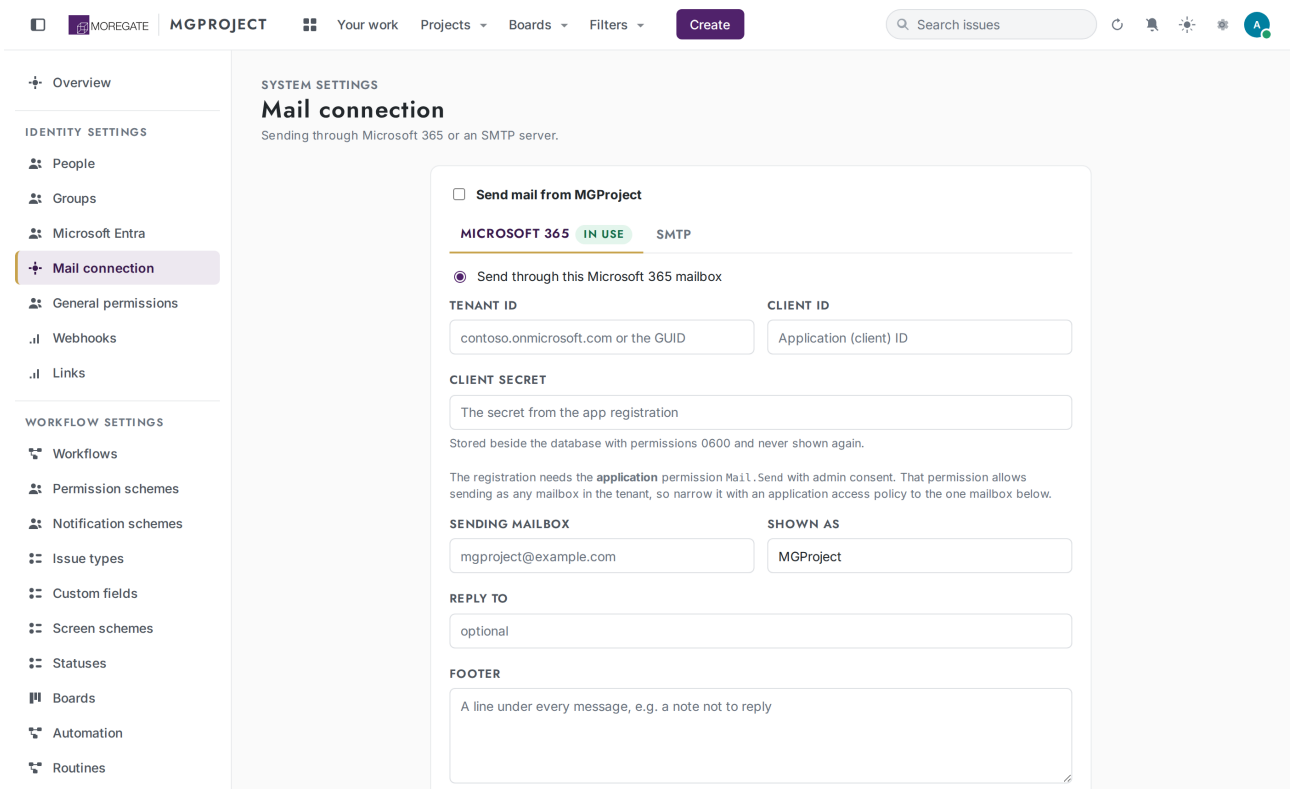
System settings Permission schemes: who may do what in a project.

A permission is granted to a group, to a named person or to a role - the assignee, the reporter, a participant, the project lead, or anybody signed in. Where nothing is granted, everybody qualifies: an empty scheme is an open one.



System settings Notification schemes: who hears about what.

A notification scheme answers, for every event - created, assigned, commented, resolved - who it reaches. Every project points at a scheme, and several may share one.

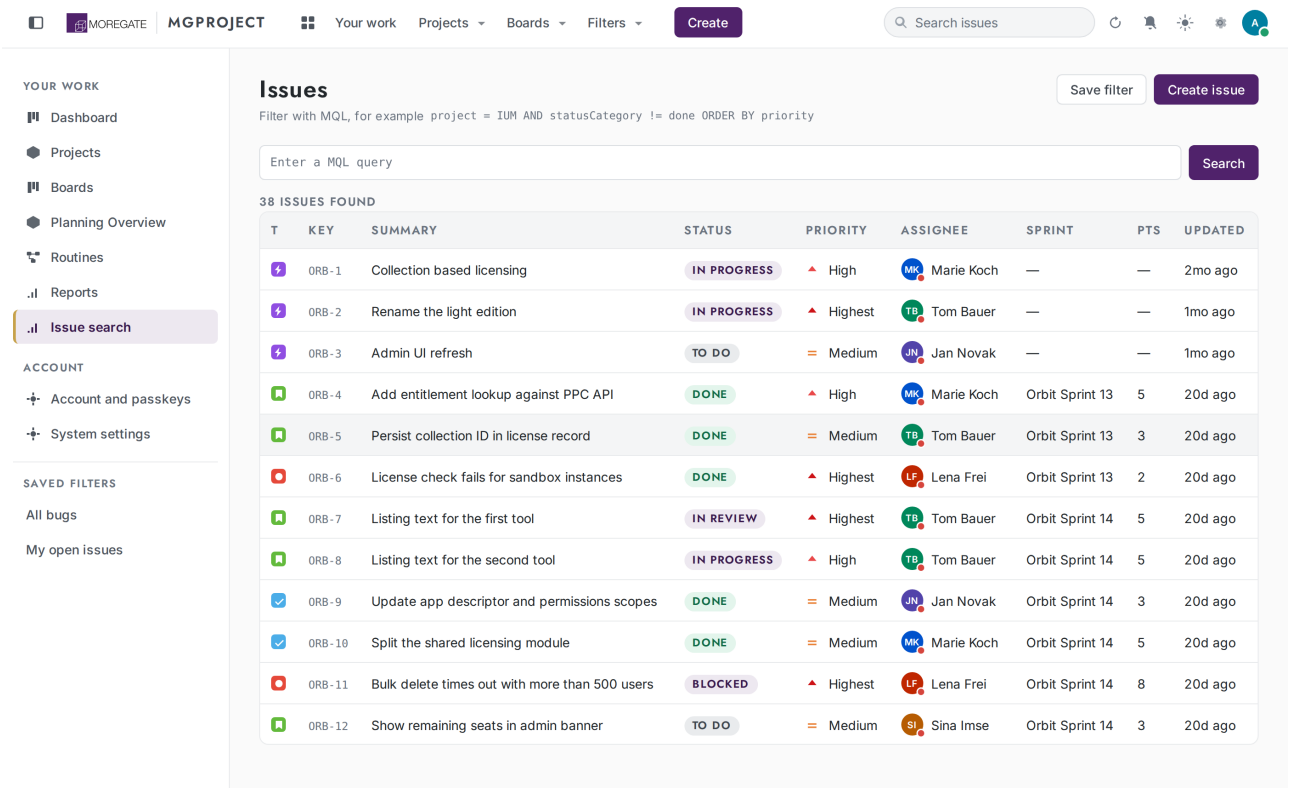


System settings Mail connection: what MGProject sends through.

Mail goes out through Microsoft 365 or an ordinary mail server. A button sends a test message before it matters.

15

Searching with MQL



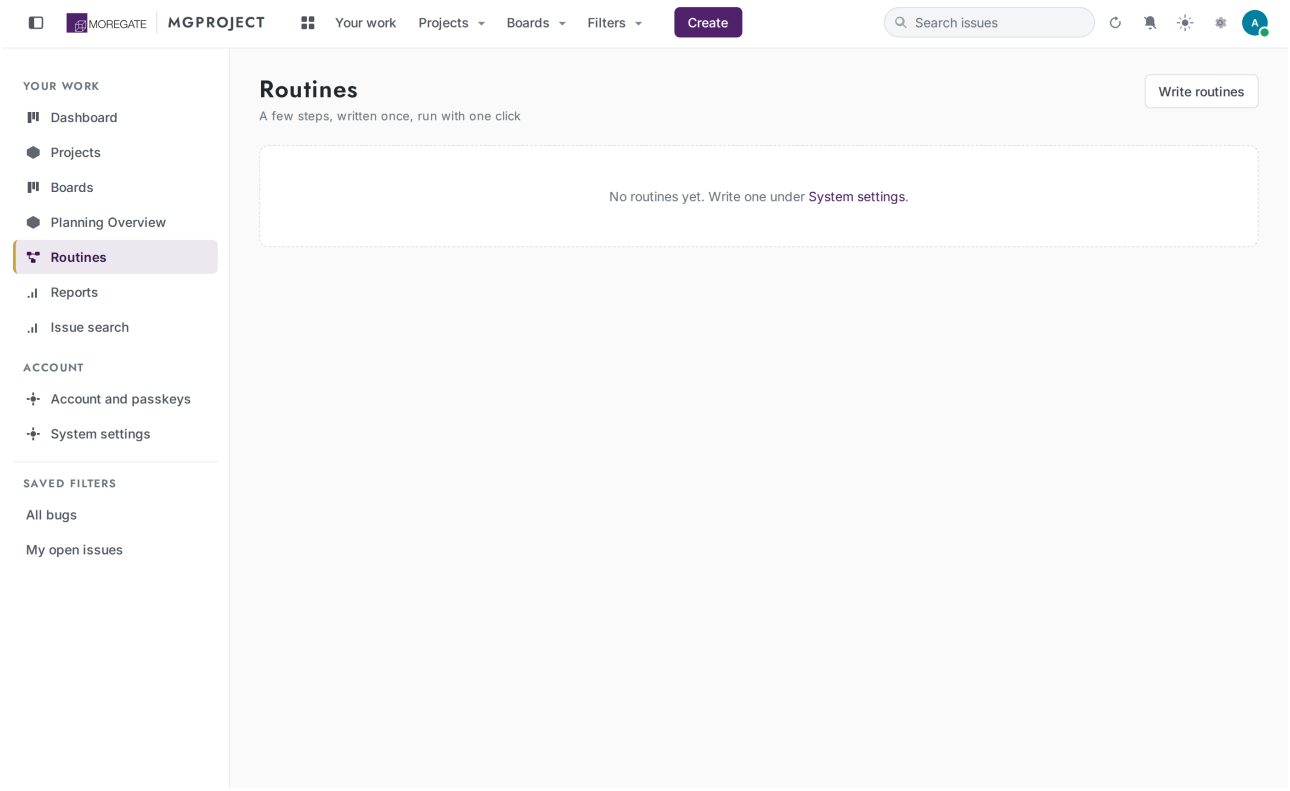
Search: a query you can save and share.

```
project = ORB AND statusCategory != done
AND participant = currentUser()
ORDER BY priority DESC
```

A query can be saved as a filter; saved filters sit in the sidebar and can be shared. The same language sits behind the board columns and the gadgets.

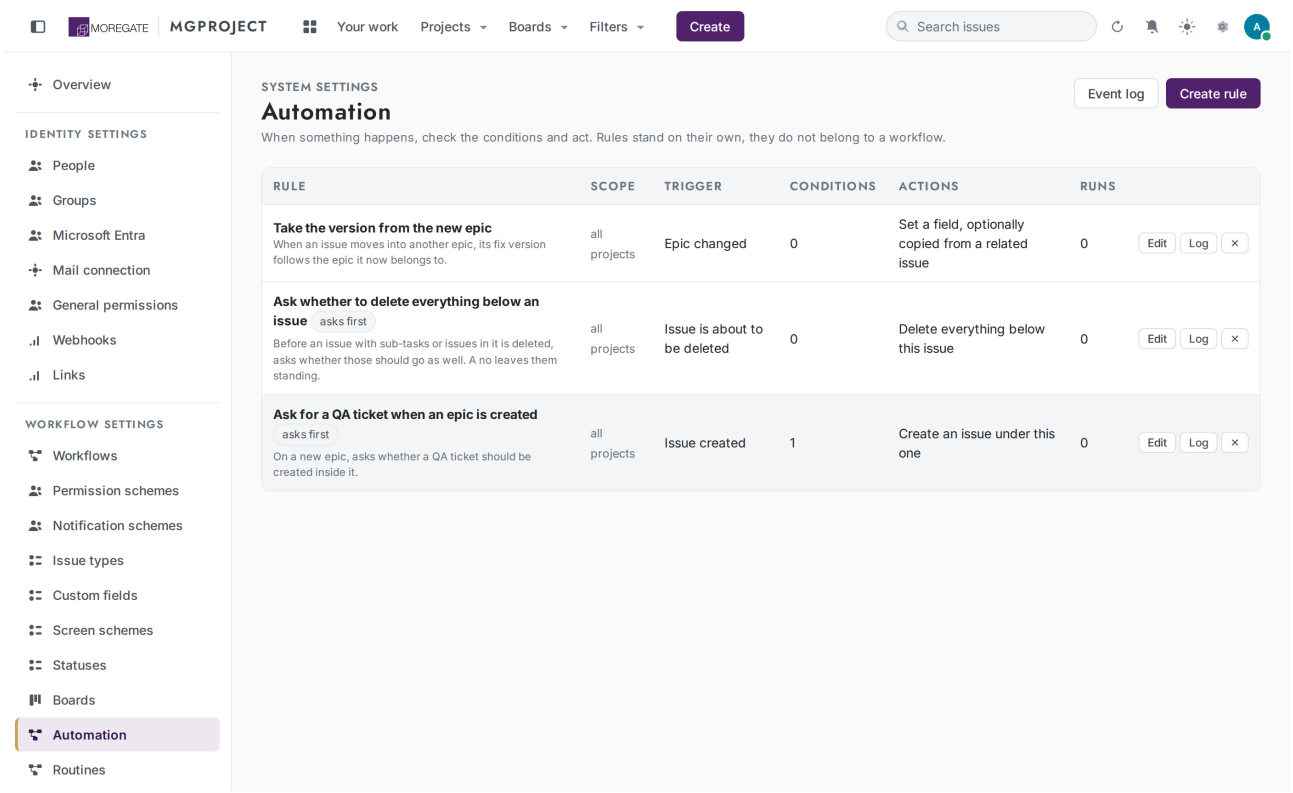
16

Routines and automation



Routines: recurring issues that appear when they are due.

A routine creates issues on a schedule - the monthly backup check, the quarterly review of permissions. It carries project, type, assignee and text with it. Writing and editing them belongs to the routine-admin group.

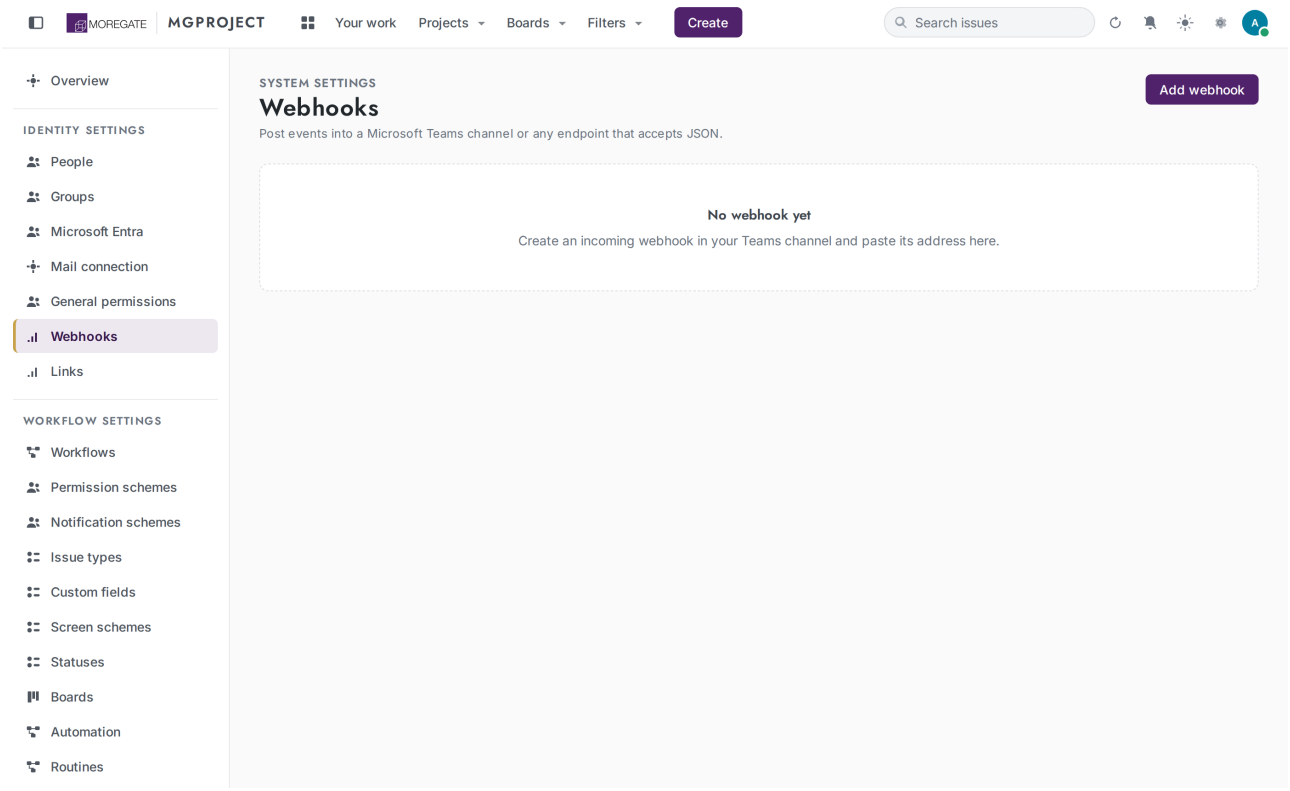


System settings Automation: what happens by itself.

Rules take over the handgrips somebody would otherwise do every day: assign when an issue of a type is created; close when nothing more comes; tell somebody when a date draws near. What a rule does appears in the issue's history with the rule as the author.

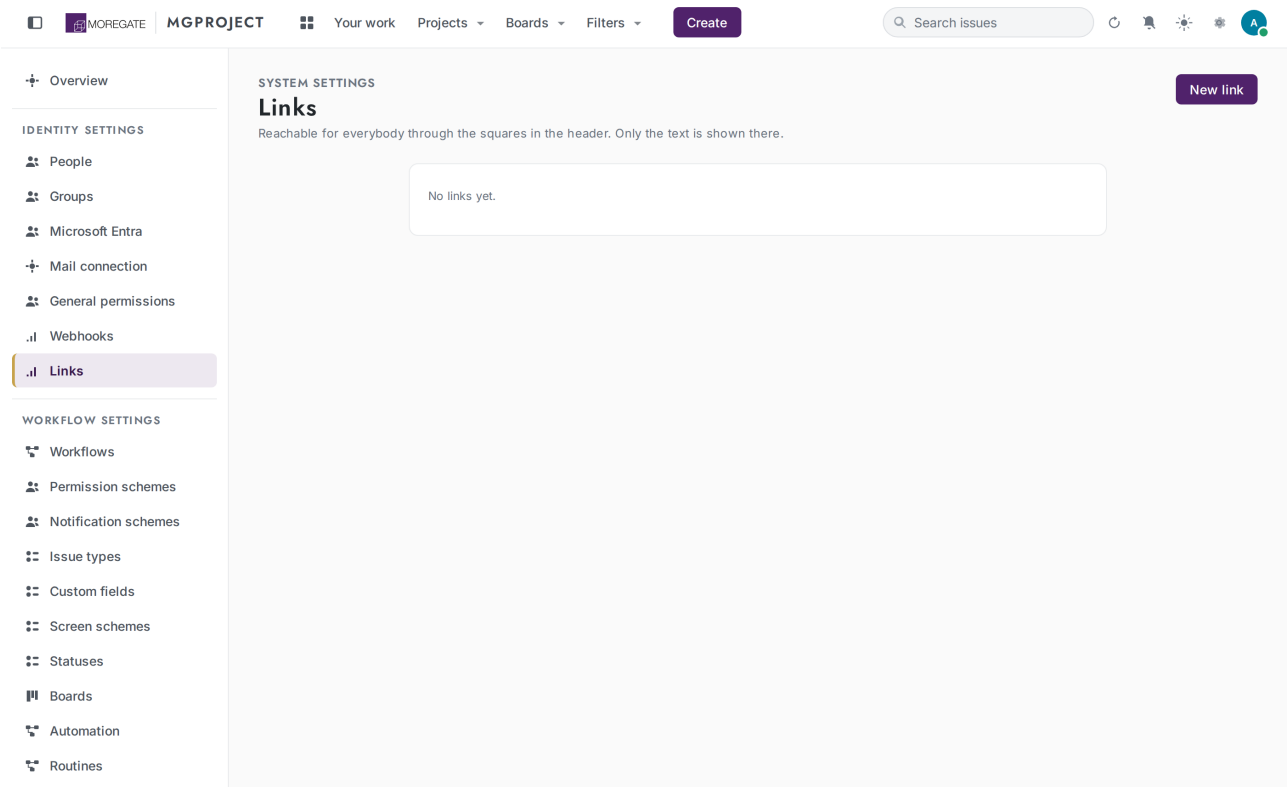
17

Webhooks and links



System settings Webhooks: events sent to another service.

A webhook posts an event as JSON to an address of your choosing - a Teams channel, or anything that accepts JSON. You pick the events, and a button sends a test call.

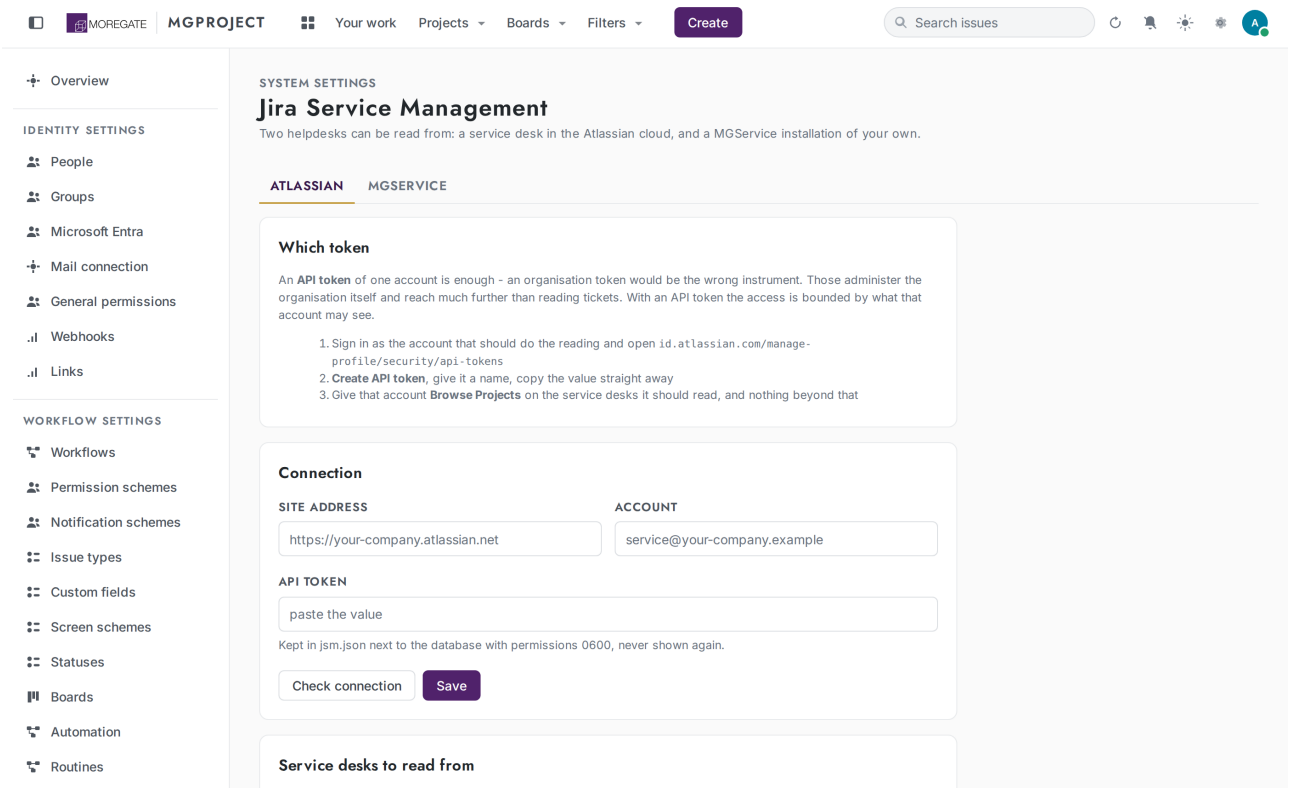


System settings Links: the shelf in the top bar.

Links sit in the header for everybody - the wiki, the monitoring, whatever the house needs. Entered once, rather than in every browser.

18

Service Management

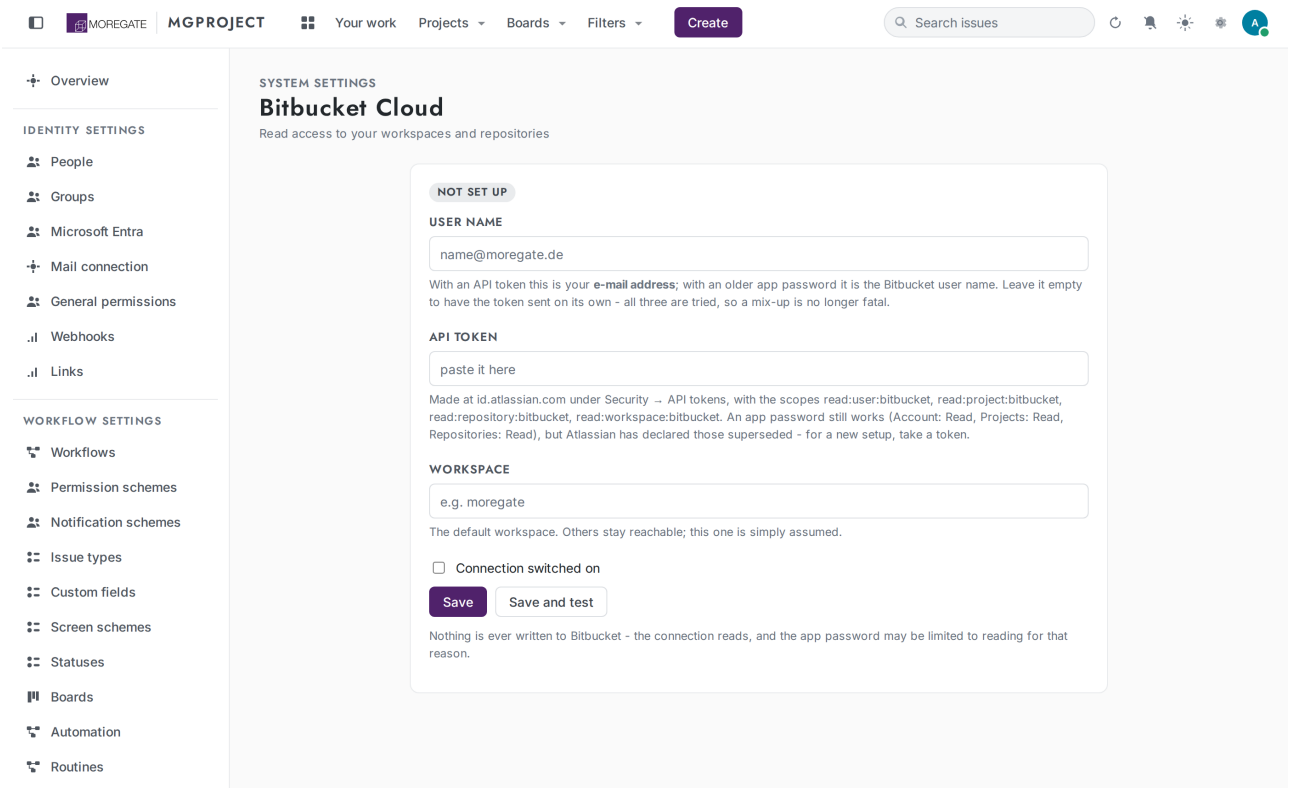


System settings Service Management: the connection to the service desk.

MGProject can be connected to a service desk - to MGService or to Jira Service Management in the cloud. A ticket becomes an issue, and the issue carries where it came from: split at the end of a sprint, the remainder takes the ticket with it. Whoever works in the service desk belongs in the helpdesk-agent group.

19

Bitbucket Cloud



System settings Bitbucket Cloud: the connection, set up once.

MGProject reads from Bitbucket Cloud: workspaces, projects and repositories. Nothing is ever written back, so the token may be limited to reading.

```
read:user:bitbucket
read:workspace:bitbucket
read:project:bitbucket
read:repository:bitbucket
```

> With an API token the user name is your e-mail address; with an older app password it is the Bitbucket user name. Mixing the two up is the usual reason a connection is refused.

An epic then carries the Bitbucket project and the repository its work lives in. Issues in the epic inherit the project and choose only the repository, so a version can be traced to the code it came from.

20

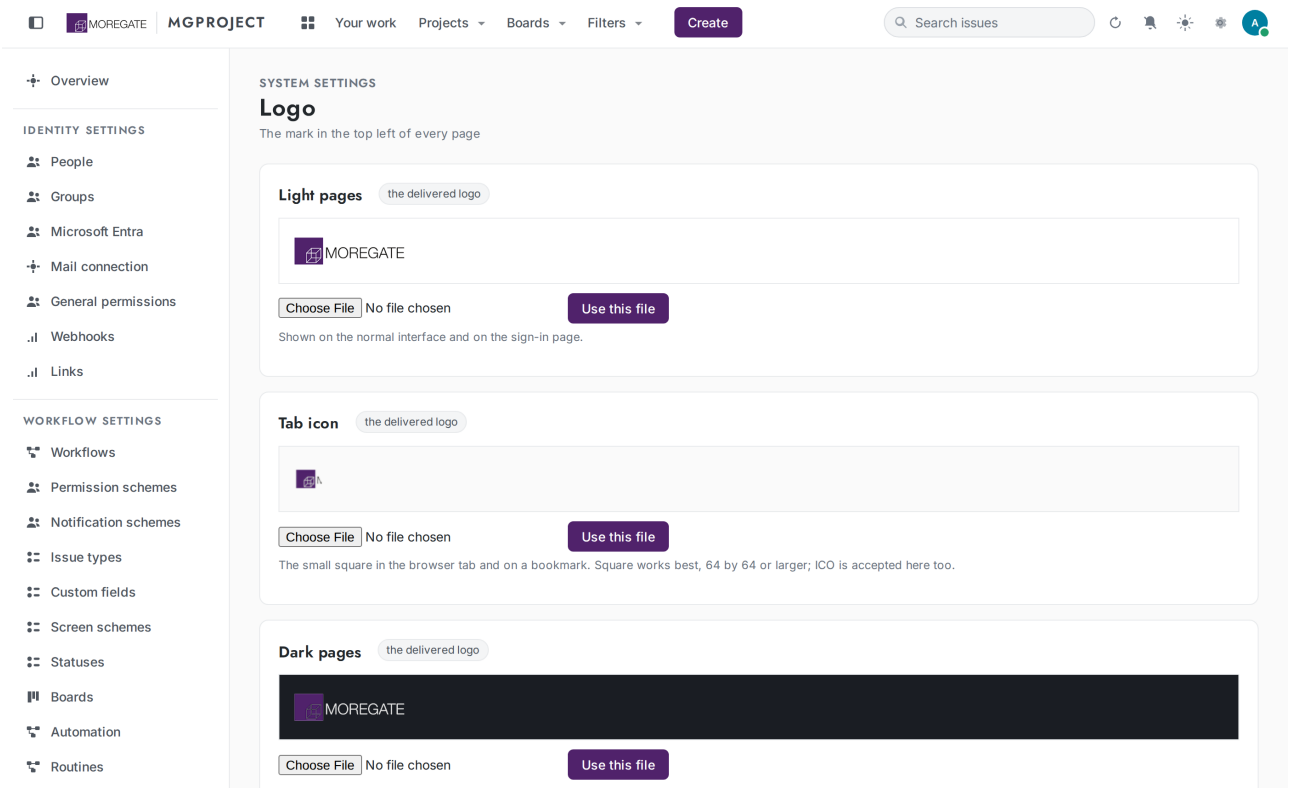
Importing from Jira

Projects can be taken over from a Jira Data Center instance with their issues, comments, attachments, sprints and worklogs. It reads through the API; nothing is ever written back.

You need the address of the instance, credentials and the projects that should come along. The run reports what was taken over and what was not - and whatever was left out is listed with its reason. A run that stopped half way can be repeated without creating anything twice.

21

Your own look, and announcements



System settings Logo: the mark, the tab icon and the accent colour.

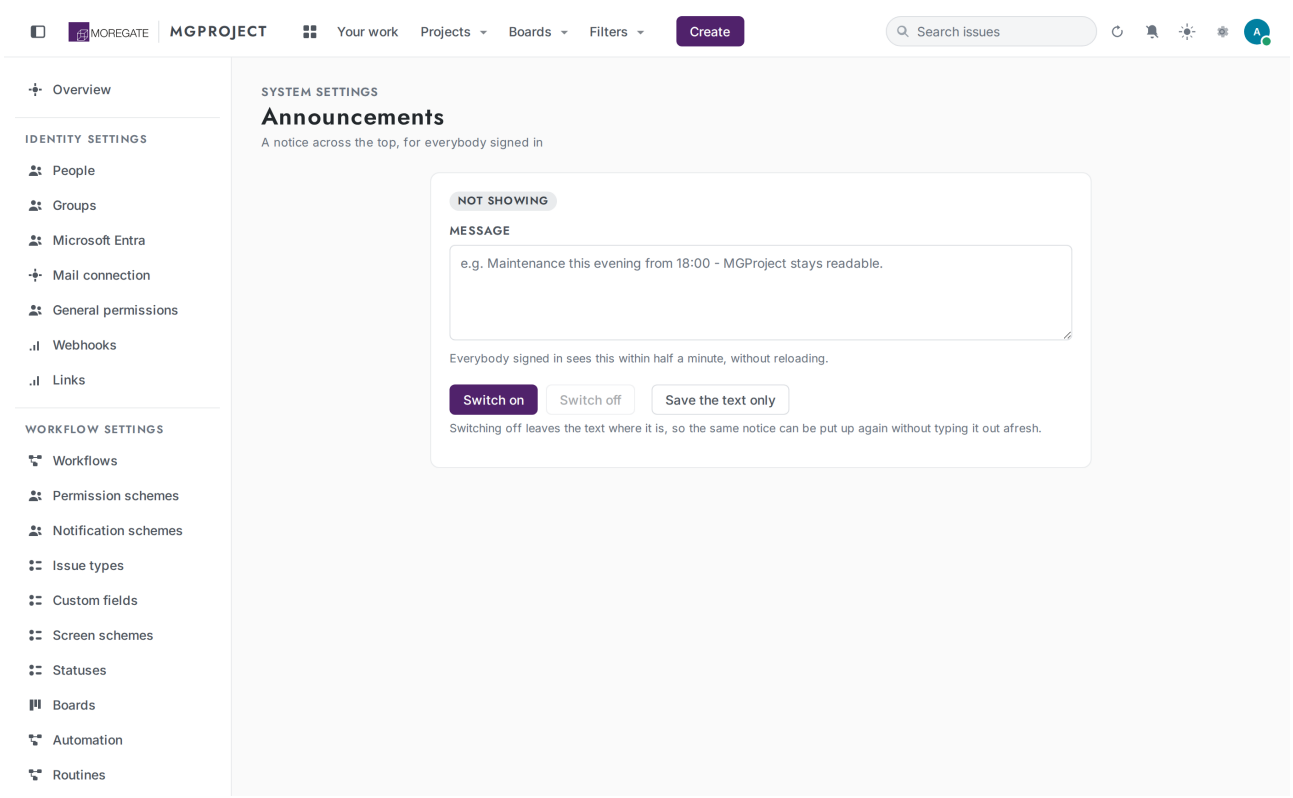
Sidebar and the installation's name

The switch at the very front of the top bar folds the sidebar away; the content then uses the full width. It is remembered per account, and without a choice of your own the sidebar is open.

Under System settings › Instance the installation can be given a name. It replaces the product name in the top bar and the browser tab, so several installations on one server can be told apart. Empty brings the product name back.

Replace the logo for light and for dark pages and the icon in the browser tab. PNG, SVG, WEBP and JPG up to 3 MB. The files live in the data directory and survive every update - a deployment writes the delivered logos only on the very first installation. The accent colour is taken from the logo and can be set by hand; on dark pages a lightened version is used.

> Files are read in the order .svg, .png, .webp, .jpg. A leftover logo.svg therefore beats a freshly uploaded logo.png. Uploading through the settings removes the other spellings; a file copied in by hand does not.

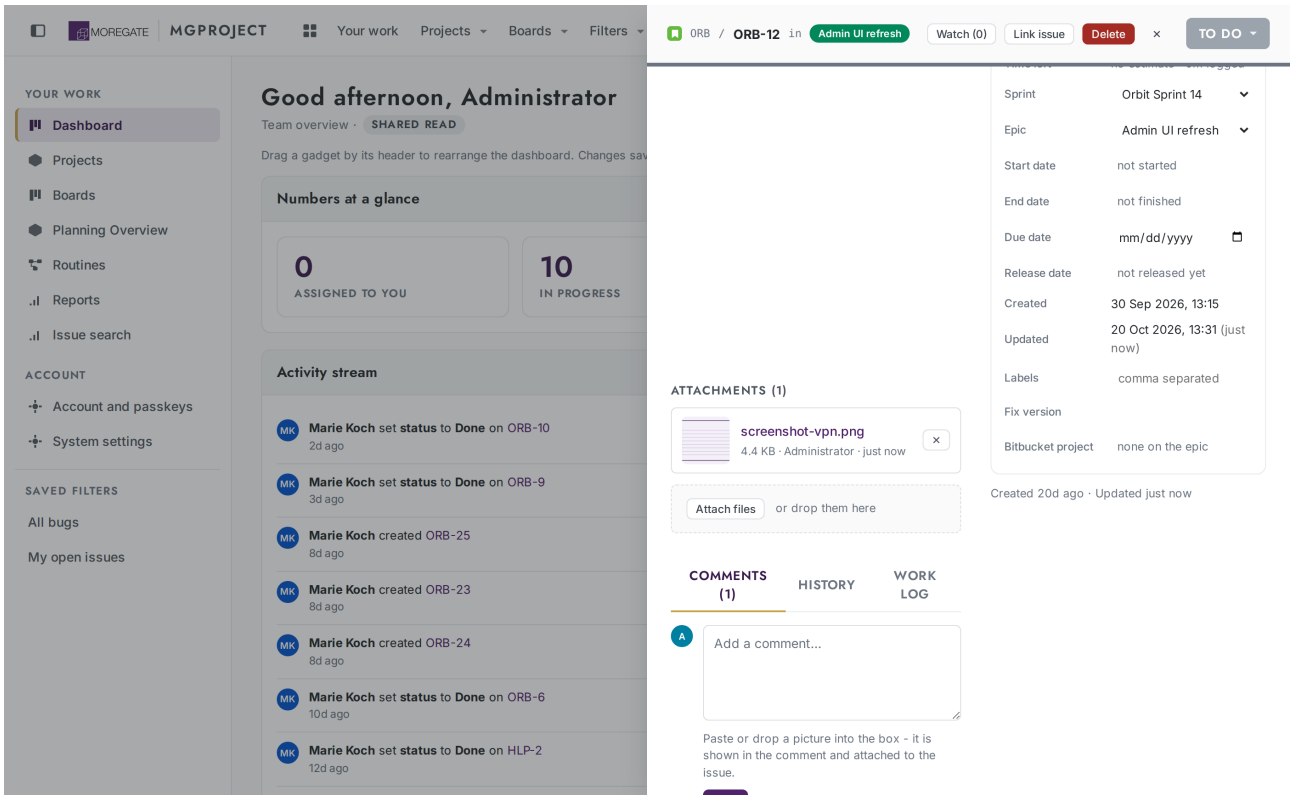


System settings Announcements: a band across every page.

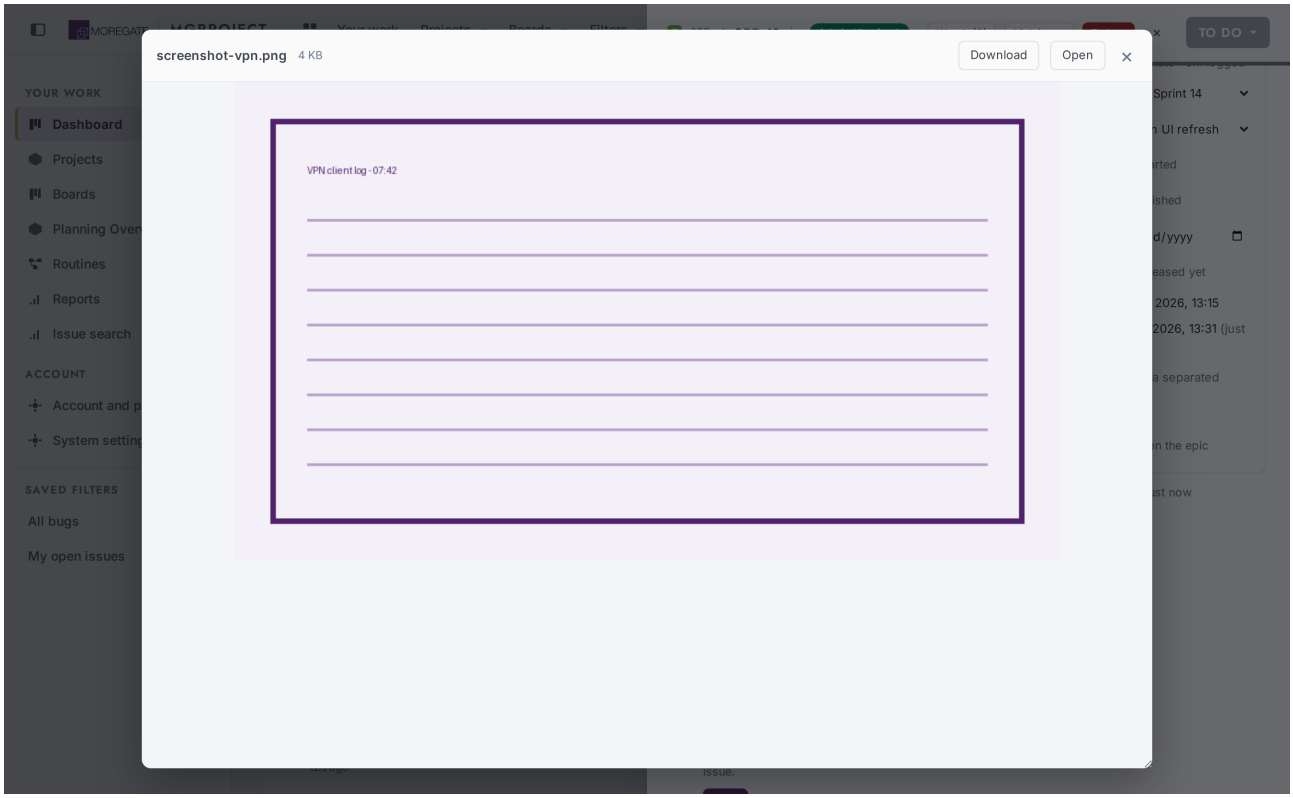
22

Attachments and the viewer

Attachments open in a viewer without leaving the browser: images, video, audio, PDF and text - and Word, Excel and other office documents, which LibreOffice converts to PDF on the server. The conversion happens once and is kept beside the attachment.



An attachment on the issue - one click opens it in the viewer.



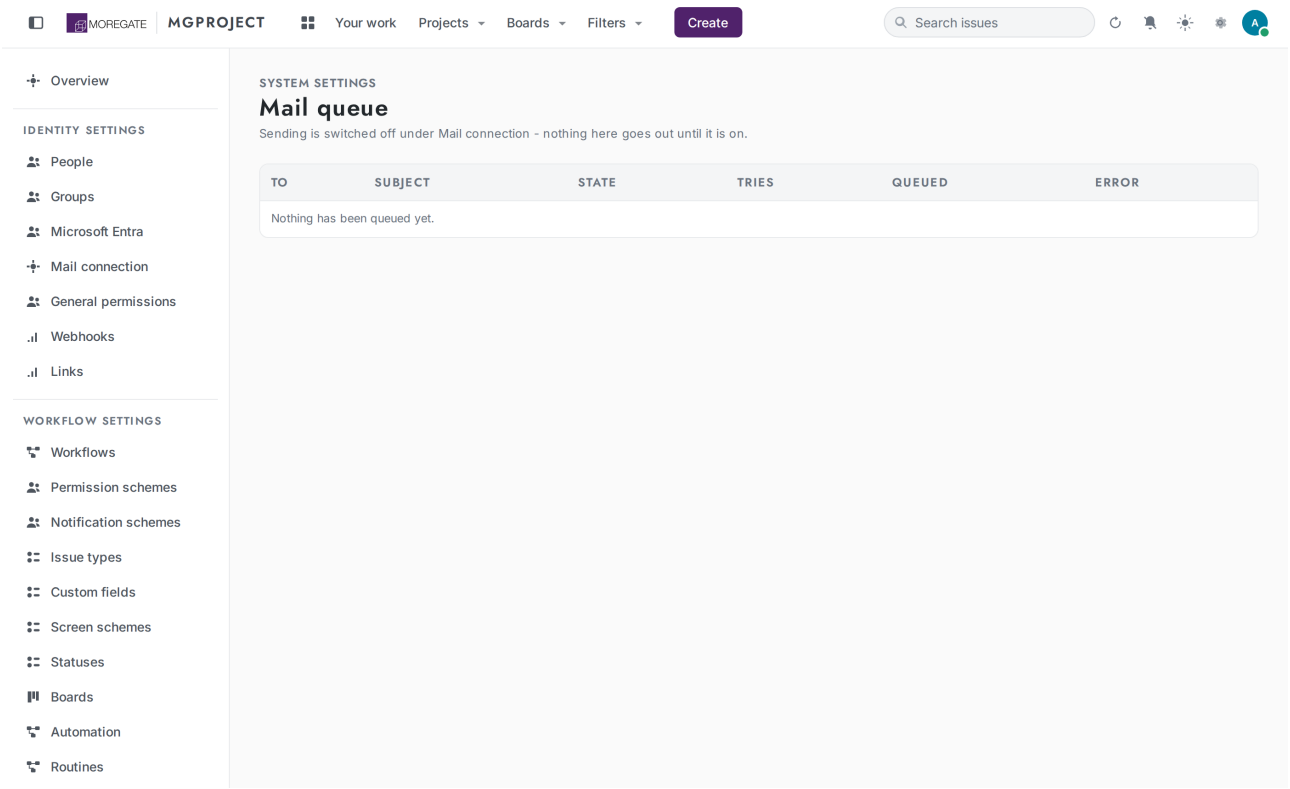
The viewer: show, download, or open in a tab of its own.

> What kind a file is comes from the file itself, not its extension. HTML and SVG are shown sandboxed - an uploaded page cannot run anything in the viewer.

Deleting an issue, a project or a space takes its files and their converted previews off the disk. A deleted number is never given out again, so nothing left over can attach itself to something new with the same number.

23

Mail queue



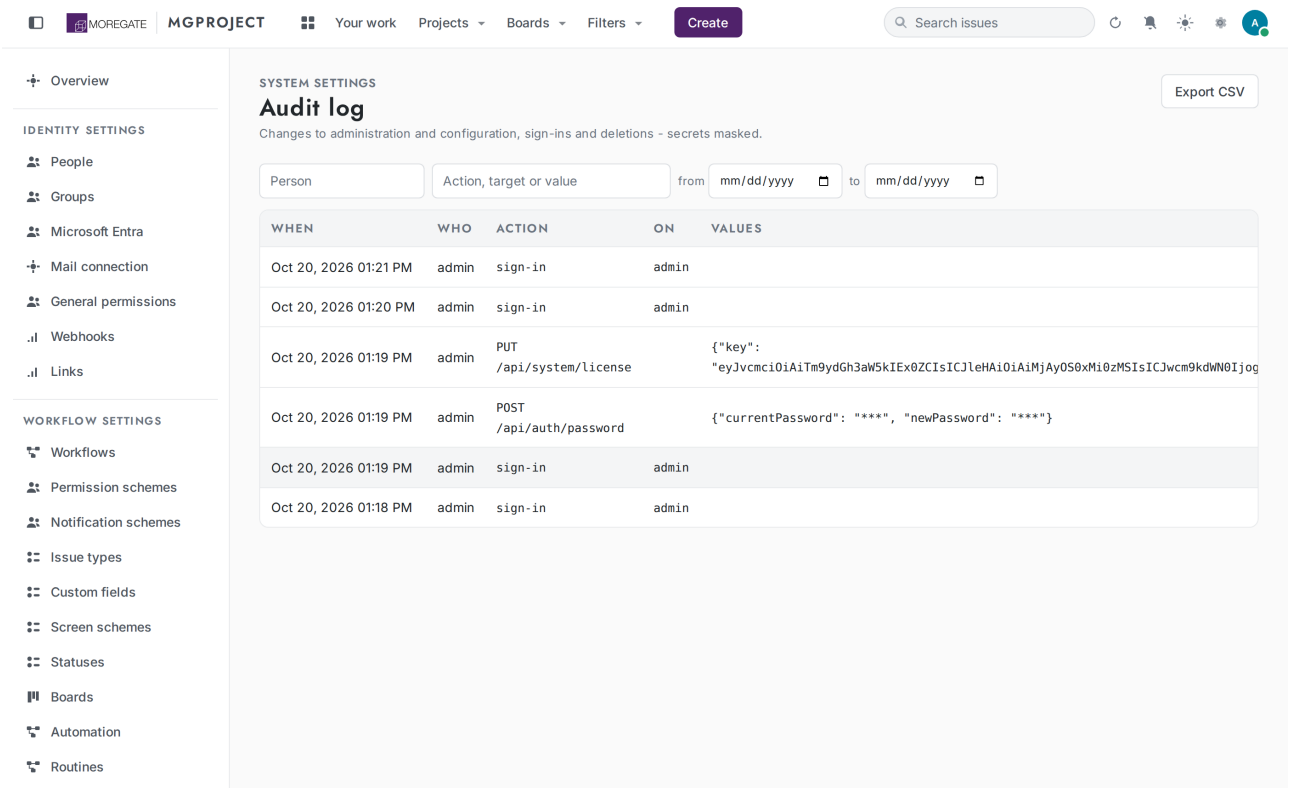
System settings Mail queue: what went, what waits, what failed.

Mail no longer goes out directly but through a queue, so saving never waits for the mailbox. A mail that fails is tried five times and then kept with the reason; from here it can be sent again.

The worker service works through the queue. The test message on the mail connection page is the one exception and goes out at once - it is meant to show whether the connection works now.

24

Audit log



System settings Audit log: who changed what, with filters and a CSV export.

Recorded are every change to administration, every sign-in and every deletion - with the person, the time, and what applied before and after. Secrets such as passwords and keys appear masked. Filter by person, kind and period; the export is CSV.

What an automation rule does is recorded as Automation - not as the person who happened to trigger it.

25

Keeping it running

Two services

MGProject runs as two systemd services: mgproject.service answers requests through gunicorn, mgproject-worker.service does everything in the background - mail, syncs, rules. A lease in the database makes sure two workers never do the same work.

Monitoring

```
curl https://.../api/health
curl https://.../api/health/details
```

/api/health says whether the service is there; /api/health/details what a monitoring system wants - database, schema steps, background jobs, disk. With MGPROJECT_LOG_FORMAT=json MGProject logs as JSON. An unexpected error is logged with a reference the answer names - whoever reports it lands at the right line of the log at once.

Checking the database without changing it

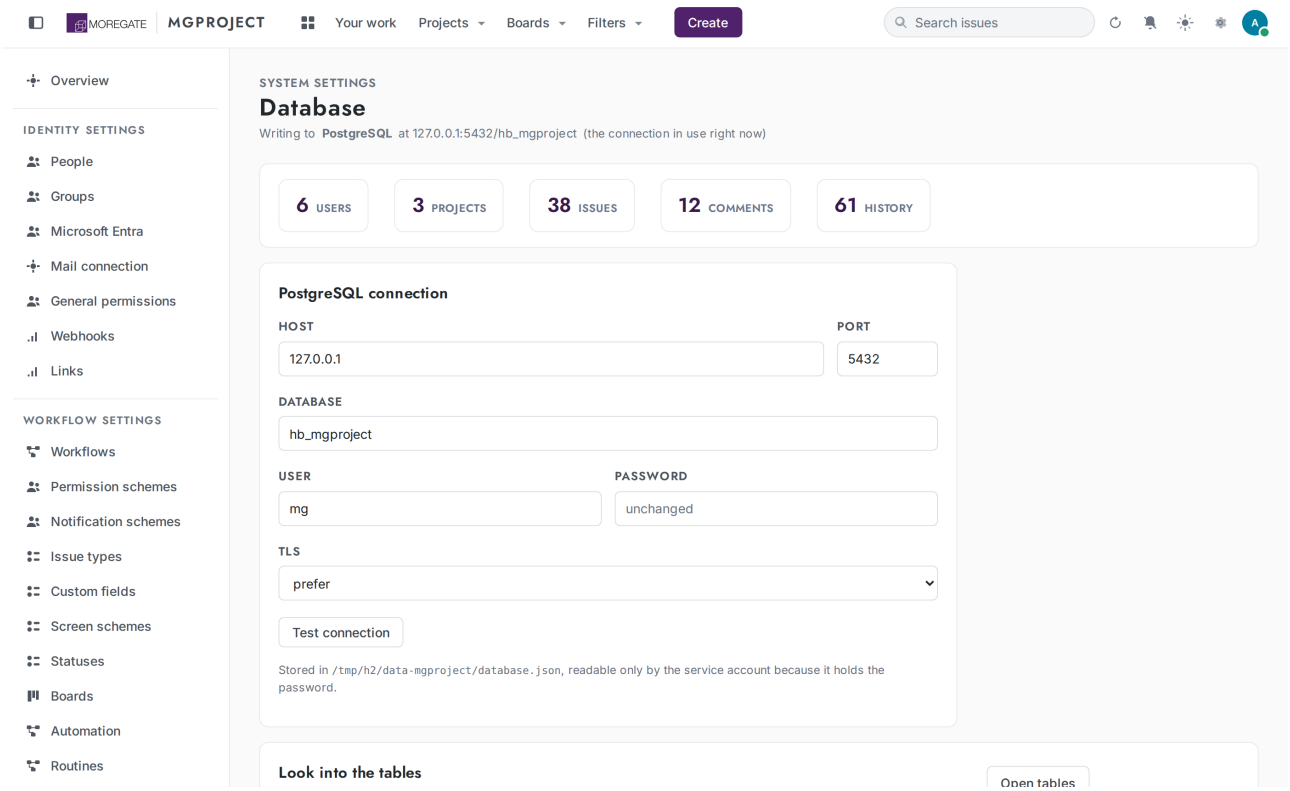
```
sudo -u mgproject /opt/mgproject/app/mgproject --manage db-check
```

Names the schema state, pending steps, missing tables and columns - and only reads. Meant also for sending us the output: no content leaves the database.

What is in the package

Every package carries a software bill of materials, sbom.cdx.json (CycloneDX 1.5): every library at its exact version with its checksum and licence, plus the Python interpreter, OpenSSL and the fonts. The build checks every package against known vulnerabilities and does not ship when one is affected.

The database



System settings Database: which database is in use and whether anything is missing.

Backups

Two things need saving: the database and the data directory. The directory holds attachments, avatars, logos and the licence - everything that is not in the database. A deployment backs up the database by itself, but only the last ten.

```
sudo -u postgres pg_dump mgproject > mgproject-$(date +%F).sql
sudo tar czf mgproject-data-$(date +%F).tar.gz /var/lib/mgproject
```

Everyday commands

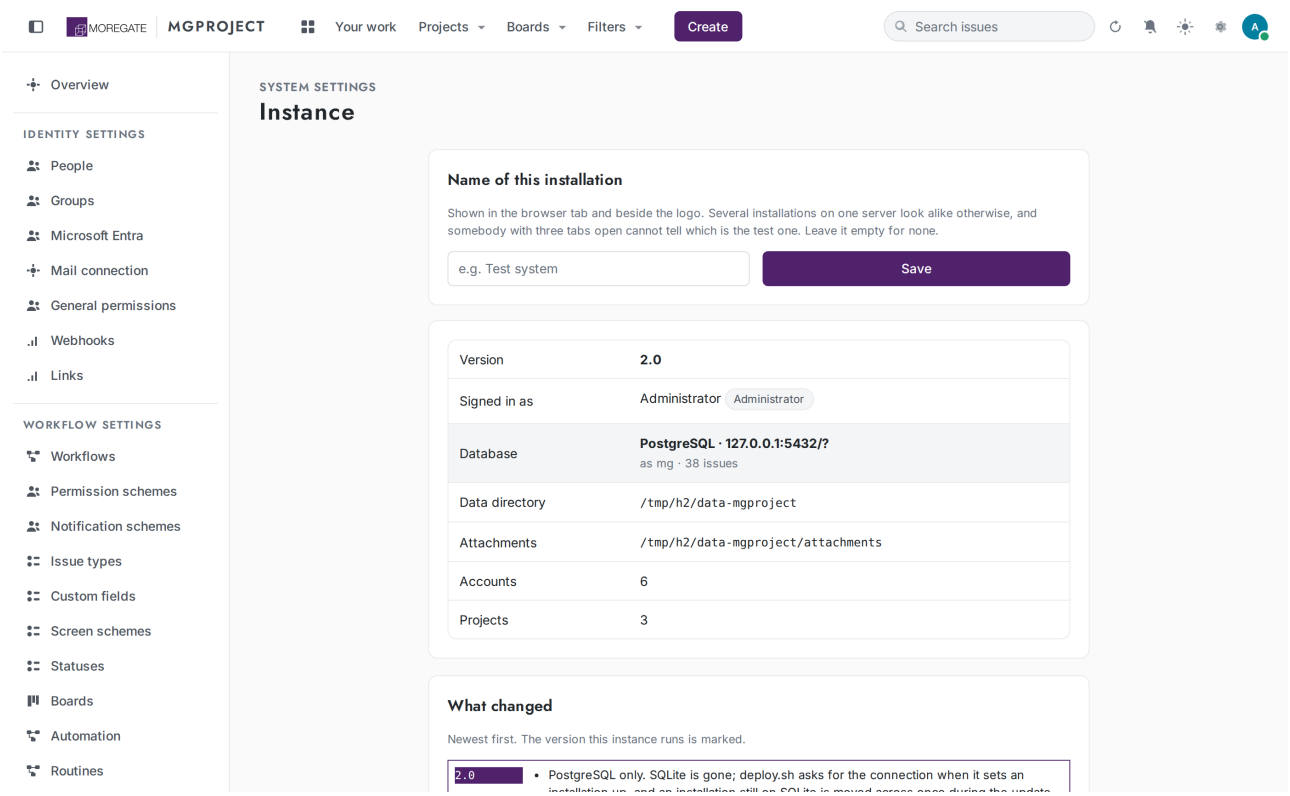
What for	Command
Status and health	sudo ./deploy.sh --status
Follow the log	journalctl -u mgproject -f
New admin password	sudo ./deploy.sh --reset-admin
Back to before the last update	sudo ./deploy.sh --rollback
Check the database, read only	/opt/mgproject/app/mgproject --manage db-check
Set a password	mgproject --manage set-password <user>
Put the delivered logos back	sudo ./deploy.sh --refresh-branding

What for	Command
Remove, keeping data	sudo ./deploy.sh --uninstall
Remove everything	sudo ./deploy.sh --uninstall --purge

--purge deletes the data directory as well: database, licence, attachments and logos. There is no way back but a backup.

When something is wrong

What you see	Where to look
502 from the proxy	The service is not running - systemctl status
413 on upload	client_max_body_size in nginx
Red band across the top	The licence, or too many accounts
Everybody signed out	MGPROJECT_SECRET was changed
No mail	The mail connection, then the log
Update refused	No valid licence



System settings Instance: version and changelog.

Every release carries its own changelog. Quoting the version from this page saves a round of questions.