
abrechnung

Release 0.12.1

Michael Loipführer, Jonas Jelten, Michael Enßlin

Jan 25, 2024

CONTENTS:

1	Installation	1
1.1	Debian Bullseye and Bookworm	1
1.2	Ubuntu Jammy	1
1.3	Docker Compose	1
2	Configuration	3
2.1	Database	3
2.2	General Options	3
2.3	API Config	4
2.4	E-Mail Delivery	4
2.5	User Registration	4
2.6	Configuration via Environment Variables	5
2.7	Frontend Configuration	5
3	Contributing	7
3.1	As Users	7
3.2	As Developers	7
4	Setup	9
4.1	Installation	9
4.2	Database Setup	10
4.3	Database migrations	10
4.4	Running tests and linters	10
4.5	Frontend Development	11
4.6	Documentation	11
5	Architecture	13
6	Packaging	15
6.1	Debian and Ubuntu	15
7	API	17
7.1	transactions	17
7.2	groups	19
7.3	auth	23
7.4	accounts	25
7.5	common	26
8	Indices and tables	27
	HTTP Routing Table	29

INSTALLATION

1.1 Debian Bullseye and Bookworm

This is the recommended installation method as it also installs the prebuilt abrechnung web app.

Simply go to the [github release page](#) and download the latest debian package matching your debian version for the latest release.

Install them via

```
sudo apt install ./abrechnung_<version>.deb
```

All dependencies, systemd services, config files in `/etc/abrechnung`, nginx config in `/etc/nginx/sites-available/abrechnung` as well as static web assets in `/usr/share/abrechnung_web` are installed.

The only remaining work to be done is to setup the database and customize the configuration (see [Configuration](#)).

1.2 Ubuntu Jammy

Follow the installation instructions for [Debian](#), just make sure to choose the correct `.deb` package file.

1.3 Docker Compose

We provide prebuilt docker containers for the api and the web frontend under <https://quay.io/abrechnung>.

To use our docker compose clone the github repository

```
git clone https://github.com/SFTtech/abrechnung.git
```

Then copy the `.env.example` file to `.env` and configure it to your liking

```
cd abrechnung
cp .env.example .env
vim .env
```

For production setups we recommend running an external postgres database but if you feel adventurous you can adapt the docker-compose file to also run a postgres container (which we definitely do not recommend). In case of using an external postgres database make sure to follow [the configuration instructions](#) on how to create a database. Afterwards make sure to include the database configuration parameters in the `.env` configuration file.

Then a simple simple

```
docker-compose -f docker-compose.prod.yaml up
```

Should suffice to get you up and running.

CONFIGURATION

After any change to the configuration make sure to apply it by restarting the respective service.

2.1 Database

The first step after installing the **abrechnung** is to setup the database. Due to the use of database specific features we only support **PostgreSQL** with versions ≥ 13 . Other versions might work but are untested.

First create a database with an associated user

```
$ sudo -u postgres psql
> create user abrechnung with password '<some secure password>';
> create database abrechnung owner abrechnung;
```

Enter the information into the config file in `/etc/abrechnung/abrechnung.yaml` under the section database as

```
database:
  host: "localhost"
  user: "abrechnung"
  dbname: "abrechnung"
  password: "<password>"
```

Apply all database migrations with

```
abrechnung db migrate
```

2.2 General Options

Some options need to be configured globally such as the base https endpoints (api and web ui) to be used in mail delivery and proper api resource URLs. If `abrechnung.example.lol` is your domain adjust the `service` section as follows. In case you change your API endpoint to another domain, port or subpath make sure to also change it here.

```
service:
  url: "https://abrechnung.example.lol"
  api_url: "https://abrechnung.example.lol/api"
  name: "Abrechnung"
```

The name is used to populate the email subjects as `[<name>] <subject>`.

2.3 API Config

Typically the config for the http API does not need to be changed much apart from one critical setting! In the `api` section make sure to insert a newly generated secret key, e.g. with

```
pwgen -S 64 1
```

The config will then look like

```
api:
  secret_key: "<your secret key>"
  host: "localhost"
  port: 8080
  id: default
```

In most cases there is no need to adjust either the `host`, `port` or `id` options.

2.4 E-Mail Delivery

To setup E-Mail delivery adjust the `email` config section to fit your use case. An example might look like

```
email:
  address: "abrechnung@example.lol"
  host: "localhost"
  port: 587
  mode: "smtp-starttls"
  auth:
    username: "abrechnung"
    password: "<verysecret>"
```

Currently supported mode options are

- `local`, uses `lmtp` on `localhost`
- `smtp-ssl`, uses `smtp` with forced `ssl`
- `smtp-starttls`, uses `smtp` with `starttls`
- if mode is not given plain `smtp` is used

The `auth` section is optional, if omitted the mail delivery daemon will try to connect to the mail server without authentication.

2.5 User Registration

This section allows to configure how users can register at the `abrechnung` instance. By default open registration is disabled.

When enabling registration without any additional settings any user will be able to create an account and use it after a successful email confirmation.

E-mail confirmation can be turned off by setting the respective config variable to `false`.


```
registration:
  enabled: true
  require_email_confirmation: true
```

Additionally open registration can be restricted adding domains to the `valid_email_domains` config variable. This will restrict account creation to users who possess an email from one of the configured domains. To still allow outside users to take part the `allow_guest_users` flag can be set which enables users to create a “guest” account when in possession of a valid group invite link. Guest users will not be able to create new groups themselves but can take part in groups they are invited to normally.

```
registration:
  enabled: true
  require_email_confirmation: true
  valid_email_domains: ["some-domain.com"]
  allow_guest_users: true
```

2.6 Configuration via Environment Variables

All of the configuration options set in the config yaml file can also be set via environment variables. The respective environment variable name for a config variable is in the pattern `ABRECHNUNG_<config section>__<variable name in capslock>`.

E.g. to set the email auth username from the config yaml as below we'd use the environment variable `ABRECHNUNG_EMAIL__AUTH__USERNAME`.

```
email:
  auth:
    username: "..."
```

2.7 Frontend Configuration

The frontend also has some configuration options. The javascript application expects a json file `config.json` under `https://<domain>/config.json`. If you want to overwrite the default config please configure your webserver to serve your designated `config.json` at that url.

This also enables server administrators to show information banners / messages on top via the `messages` config key to e.g. announce maintenance.

Possible config options are

```
{
  "messages": [
    {
      "type": "<'error'|'info'|'warning'|'success'>",
      "title": "<string, optional>",
      "body": "<string>"
    }
  ],
  "imprintURL": "<string, optional>",
  "sourceCodeURL": "https://github.com/SFTtech/abrechnung"
}
```


CONTRIBUTING

3.1 As Users

In case you encounter bugs or have ideas for new and useful features don't hesitate to open an [issue](#) and start a discussion.

3.2 As Developers

Get started by setting up a local *development environment* and reading through the *Architecture* overview and then start hacking. [Pull requests](#) are always welcome!

Table of Contents

- *Setup*
 - *Installation*
 - *Database Setup*
 - *Database migrations*
 - *Running tests and linters*
 - *Frontend Development*
 - *Documentation*

4.1 Installation

Fork and clone the repository

```
git clone https://github.com/SFTtech/abrechnung.git
cd abrechnung
```

Then install the package in local development mode as well as all required dependencies. Make sure to have `flit` installed first. Installing the dependencies can be two ways:

Setup a virtual environment and install the packages via pip (straightforward way)

```
virtualenv -p python3 venv
source venv/bin/activate
pip install flit
flit install -s --deps develop
```

Or install the dependencies through your package manager (useful for distribution packaging)

- arch linux (slight chance some dependencies may be missing here)

```
sudo pacman -S python-flit python-yaml python-aiohttp python-aiohttp-cors python-asyncpg
python-sphinx python-schema python-email-validator python-bcrypt python-pyjwt python-
python-aiohttp python-pytest python-pytest-cov python-black python-mypy python-pylint python-
python-apispec python-marshmallow python-webargs
```

Afterwards install the package without dependencies

```
flit install -s --deps none
```

4.2 Database Setup

Have a running **PostgreSQL** database server. Create the database (in a psql prompt):

```
create role someuser with login password 'somepassword';
create database somedatabasename owner someuser;
```

- exit the psql prompt
- Copy config/abrechnung.yaml to the project root folder and adjust it, e.g. enter someuser, somepassword and somedatabasename
- Populate database by applying all migrations:

```
abrechnung -c abrechnung.yaml db migrate
```

- Launch `abrechnung -c abrechnung.yaml api`
- Launch `abrechnung -c abrechnung.yaml mailer` to get mail delivery (working mail server in config file required!)

It is always possible wipe and rebuild the database with

```
abrechnung -c abrechnung.yaml db rebuild
```

4.3 Database migrations

In case a new features requires changes to the database schema create a new migration file with

```
./tools/create_revision.py <revision_name>
```

In case you did not install the abrechnung in development mode it might be necessary to add the project root folder to your PYTHONPATH.

4.4 Running tests and linters

To run the tests a dedicated **PostgreSQL** instance is required. The tests assume defaults for the name, user and password as

- username: `abrechnung-test`
- database: `abrechnung-test`
- password: `asdf1234`

In case you want to use a different database / user they can be overwritten using environment variables:

- `TEST_DB_USER`
- `TEST_DB_HOST`

- TEST_DB_PASSWORD
- TEST_DB_DATABASE

Make sure the database user has owner permissions on the `public` schema of the database as dropping and recreating is used as a means to wipe and repopulate the database between tests.

```
alter schema public owner to "<your user>"
```

Finally run the tests via

```
make test
```

Run the linters via

```
make lint
```

4.5 Frontend Development

Working on the frontend is quite easy, simply

```
cd web  
yarn install  
yarn start
```

and you are good to go!

4.6 Documentation

To build the documentation locally simply run

```
make docs
```

The html docs can then be found in `docs/_build`.

ARCHITECTURE

TODO

- PostgreSQL DB, exposed as websocket and http REST API
- various python services (e.g. mail delivery, database cleanup)
- web-client

PACKAGING

The main `abrechnung` python package is automatically built and uploaded to [Test Pypi](#) and [Pypi](#) on every tag on master by the CI pipeline.

6.1 Debian and Ubuntu

Debian and Ubuntu packages are built by the CI on every push to master and are attached to a new release on every tag to master.

To build the packages locally run:

```
./tools/build_debian_packages.py
```

Docker is a requirement as the packages are built inside a matching docker container for each distribution.

7.1 transactions

GET /api/v1/groups/{group_id}/transactions

list all transactions in a group

Parameters

- **group_id** (*integer*) –

Query Parameters

- **min_last_changed** ({'date-time', 'null'}) –
- **transaction_ids** ({'null', 'string'}) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/transactions

create a new transaction

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/transactions/{transaction_id}

get transaction details

Parameters

- **transaction_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/transactions/{transaction_id}

update transaction details

Parameters

- **transaction_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

DELETE /api/v1/transactions/{transaction_id}

delete a transaction

Parameters

- **transaction_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/transactions/{transaction_id}/positions

update transaction positions

Parameters

- **transaction_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/files/{file_id}/{blob_id}

fetch the (binary) contents of a transaction attachment

Parameters

- **file_id** (*integer*) –
- **blob_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

7.2 groups

POST /api/v1/groups/preview

preview a group before joining using an invite token

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/join

join a group using an invite token

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/groups

list the current users groups

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found

POST /api/v1/groups

create a group

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/groups/{group_id}

fetch group details

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}

update group details

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

DELETE /api/v1/groups/{group_id}

delete a group

Parameters

- **group_id** (*integer*) –

Status Codes

- 204 No Content – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found

- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/leave

leave a group

Parameters

- **group_id**(*integer*) –

Status Codes

- 204 No Content – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/groups/{group_id}/members

list all members of a group

Parameters

- **group_id**(*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/members

update the permissions of a group member

Parameters

- **group_id**(*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/groups/{group_id}/logs

fetch the group log

Parameters

- **group_id**(*integer*) –

Status Codes

- 200 OK – Successful Response

- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/send_message

post a message to the group log

Parameters

- **group_id** (*integer*) –

Status Codes

- 204 No Content – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/groups/{group_id}/invites

list all invite links of a group

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/invites

create a new group invite link

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

DELETE /api/v1/groups/{group_id}/invites/{invite_id}

delete a group invite link

Parameters

- **group_id** (*integer*) –
- **invite_id** (*integer*) –

Status Codes

- 204 No Content – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

7.3 auth

POST /api/v1/auth/token

login with username and password

Status Codes

- 200 OK – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/login

login with username and password

Status Codes

- 200 OK – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/logout

sign out of the current session

Status Codes

- 204 No Content – Successful Response

POST /api/v1/auth/register

register a new user

Status Codes

- 200 OK – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/confirm_registration

confirm a pending registration

Status Codes

- 204 No Content – Successful Response

- 422 Unprocessable Entity – Validation Error

GET /api/v1/profile

fetch user profile information

Status Codes

- 200 OK – Successful Response

POST /api/v1/profile/change_password

change password

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/profile/change_email

change email

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/confirm_email_change

confirm a pending email change

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/recover_password

recover password

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/confirm_password_recovery

confirm a pending password recovery

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/delete_session

delete a given user session

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

POST /api/v1/auth/rename_session

rename a given user session

Status Codes

- 204 No Content – Successful Response
- 422 Unprocessable Entity – Validation Error

7.4 accounts

GET /api/v1/groups/{group_id}/accounts

list all accounts in a group

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

POST /api/v1/groups/{group_id}/accounts

create a new group account

Parameters

- **group_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

GET /api/v1/accounts/{account_id}

fetch a group account

Parameters

- **account_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found

- 422 Unprocessable Entity – Validation Error

POST /api/v1/accounts/{account_id}

update an account

Parameters

- **account_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

DELETE /api/v1/accounts/{account_id}

delete an account

Parameters

- **account_id** (*integer*) –

Status Codes

- 200 OK – Successful Response
- 401 Unauthorized – unauthorized
- 403 Forbidden – forbidden
- 404 Not Found – Not found
- 422 Unprocessable Entity – Validation Error

7.5 common

GET /api/version

Get Version

Status Codes

- 200 OK – Successful Response

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

HTTP ROUTING TABLE

/api

GET /api/v1/accounts/{account_id}, 25
GET /api/v1/files/{file_id}/{blob_id}, 19
GET /api/v1/groups, 19
GET /api/v1/groups/{group_id}, 20
GET /api/v1/groups/{group_id}/accounts, 25
GET /api/v1/groups/{group_id}/invites, 22
GET /api/v1/groups/{group_id}/logs, 21
GET /api/v1/groups/{group_id}/members, 21
GET /api/v1/groups/{group_id}/transactions,
17
GET /api/v1/profile, 24
GET /api/v1/transactions/{transaction_id}, 17
GET /api/version, 26
POST /api/v1/accounts/{account_id}, 26
POST /api/v1/auth/confirm_email_change, 24
POST /api/v1/auth/confirm_password_recovery,
24
POST /api/v1/auth/confirm_registration, 23
POST /api/v1/auth/delete_session, 24
POST /api/v1/auth/login, 23
POST /api/v1/auth/logout, 23
POST /api/v1/auth/recover_password, 24
POST /api/v1/auth/register, 23
POST /api/v1/auth/rename_session, 24
POST /api/v1/auth/token, 23
POST /api/v1/groups, 19
POST /api/v1/groups/join, 19
POST /api/v1/groups/preview, 19
POST /api/v1/groups/{group_id}, 20
POST /api/v1/groups/{group_id}/accounts, 25
POST /api/v1/groups/{group_id}/invites, 22
POST /api/v1/groups/{group_id}/leave, 21
POST /api/v1/groups/{group_id}/members, 21
POST /api/v1/groups/{group_id}/send_message,
22
POST /api/v1/groups/{group_id}/transactions,
17
POST /api/v1/profile/change_email, 24
POST /api/v1/profile/change_password, 24
POST /api/v1/transactions/{transaction_id},
18

POST /api/v1/transactions/{transaction_id}/positions,
18
DELETE /api/v1/accounts/{account_id}, 26
DELETE /api/v1/groups/{group_id}, 20
DELETE /api/v1/groups/{group_id}/invites/{invite_id},
22
DELETE /api/v1/transactions/{transaction_id},
18