ILMO Documentation

Release 0.0.1

Julian-Samuel Gebühr

Jul 19, 2023

CONTENTS:

1 Users guide **1**
- 1.1 Registration . . . 1
- 1.2 Lending . . . 1
- 1.3 Login . . . 1
- 1.4 E-Mail . . . 1

2 Administration **3**
- 2.1 Create user . . . 3
- 2.2 Lending . . . 3
- 2.3 Returning . . . 3
- 2.4 Opening hours . . . 4
- 2.5 Adding Items . . . 4
- 2.6 Monitoring . . . 5

3 Installation, customization and contributing **9**
- 3.1 Deployment . . . 9
- 3.2 Contributing . . . 14
- 3.3 Release . . . 14
- 3.4 Backup & Restore . . . 15

4 API Documentation **21**
- 4.1 API Access . . . 21
- 4.2 Access Control . . . 21

CHAPTER

ONE

USERS GUIDE

1.1 Registration

To register you have to visit the library. An librarian will then set up an account for you. You will need to provide an valid E-Mail Address and a password.

1.2 Lending

You can check the available books or material online with your account.

To borrow any books or material you have to visit the library and search for the item you want to borrow.

An administrator will then check the label of the item. The item will then be marked as borrowed by you (visible only for librarians.

You have to return the item according to the rules of the library. If you don't, there will be a reminder.

1.3 Login

After your *Registration* you are able to log into your account with your E-Mail Address or user-ID and your password.

You can then access:

- your borrowed items
- the available books and items
- the opening hours

If you want to make changes to your profile ask an librarian.

1.4 E-Mail

With your *Registration* you provide a valid E-Mail address. For every lending you will be reminded via E-Mail. The interval of the reminder is defined by the library.

CHAPTER

TWO

ADMINISTRATION

2.1 Create user

Users are created by the administrators manually. Select *Users->Add* in the admin interface.

You will need to provide the following information:

- Username: unique, required
- password: rquired
- Language: optional

Although it is not required it is strongly recommended to then further edit the user and add

- Forename
- Surname
- E-Mail
- Permissions

2.2 Lending

As a user wants to lend an item you need to know the following:

- Item ID [e.g. `CH42 d`]
- The users first or last name

Search for the item with it's label via the *Search* option in the navigation bar. Select the appropriate item and click *Borrow*. Then search for your user and click *Borrow* again. The loan is now successfully registered.

2.3 Returning

To return an item either visit the page *All loans* and search for the loan there or you search for the item via *Search*.

If you found the loan, you can simply click on the button *Return* and you are finished.

2.4 Opening hours

The opening hours can be changed by selecting the page *Opening hours* in the navigation menu. You can not change an entry, simply delete it and create a new one.

Note: It is advised to fill empty time cells with a "-".

2.5 Adding Items

2.5.1 General Structure

ILMO uses a model of `Books` and `BookInstances` or *Material* ` and `MaterialInstances`. A book contains general information about a book

```
Book:
    title: string, required
    author: Author object, required
    genre: Genre object
    summary: String
    isbn: String
    language: Language object
```

As you can see there are some objects that refer to other models like `Author`. To create a book you need to create these first (or add them later, not recommended).

The physical books that are in the library are then represented in `BookInstance`. It has the following properties

```
BookInstance:
    label: String, required
    book: Author object, required
    loan_status: available | maintenance | On Loan | Reserved, required
    imprint: String
```

2.5.2 Add Books

ILMO currently only supports adding books via the admin interface.

ID System

As most libraries do often have multiple copies of the same book an identification system is proposed that accounts for that. Feel free to use your own, this is just an inspiration!

For books the ID format therefore consists of a `stem` of category and number and a `numbering` with characters of copies.

```
CC[number] [ii]

"CH42 c"    # Titel 42 in category Chemistry, copy number 3
```

(continues on next page)

(continued from previous page)

```
"XY132 af"  # Titel 132 in category XY, copy number 33

"a16792 c"  # bad example as categories should be two capital letters and the
            # titel number should be ascending
```

- **CC:**
  Category abbreviation e. `CH` for chemistry. It is advised to use a two letter abbreviation, but there is no technical limit.
- **Number:**
  Indicates which title the book has. The number is ascending incremented for every title added to the category.
- **ii:**
  index of the copy. For the first copy the index is `a` for the second it is `b` and for the 27th copy it is `aa`.

2.5.3 Add material

ILMO currently only supports adding material via the admin interface.

ID System

Material has a different ID system. It consists of an abbreviation and an index.

```
AA [index]

"LC 42"           # Labcoat number 42
"SG 132"          # Safety glasses number 132

"Labcoat 16792"   # bad example as abbreviation should be two capital letters and index
                  # number should be ascending
```

- **AA:**
  Abbreviation of the items name
- **Index:**
  Ascending number. The highest index of an item is the number of items of this name.

2.6 Monitoring

ILMO should, like every other software, be easy to monitor. Therefore a basic metrics are exposed to *https://example.com/library/metrics*. The data is encoded in JSON format and is therefore suitable to bea read by humans and it is easy to use it as data source for further processing.

2.6.1 Exposed Metrics

```
users: number of users (all roles combined)
staff: number of users with staff status
books: number of books
book_instances: number of physical copies of books
book_instances_available: number of available physical copies
materials: number of materials
material_instances: number of physical copies of material
material_instances_available: number of available physical copies
authors: number of authors
loans: number of loans
unreturned_loans: number of unreturned loans
reminder_sent_today: number of reminders that were sent to users on this day
```

2.6.2 Example workflow

To use the exposed metrics you will usually need a time series database and a visualization tool.

As time series database we will utilize InfluxDB, the visualization tool will be Grafana.

InfluxDB and Telegraf

First we install InfluxDB (e.g. with docker, be aware of the security risks!).

```
# Pull the image
$ sudo docker pull influxdb

# Start influxdb
$ sudo docker run -d -p 8086:8086 -v influxdb:/var/lib/influxdb --name influxdb influxdb

# Start influxdb console
$ docker exec -it influxdb influx
Connected to http://localhost:8086 version 1.8.3
InfluxDB shell version: 1.8.3
> create database monitoring
> create user "telegraf" with password 'mypassword'
> grant all on monitoring to telegraf
```

Note: When creating the user telegraf check the double and single quotes for username an password.

Now install telegraf and configure *etc/telegraf/telegraf.conf*. Modify the domain and your password for the InfluxDB database.

```
# Global tags can be specified here in key="value" format.
[global_tags]


# Configuration for telegraf agent
[agent]
```

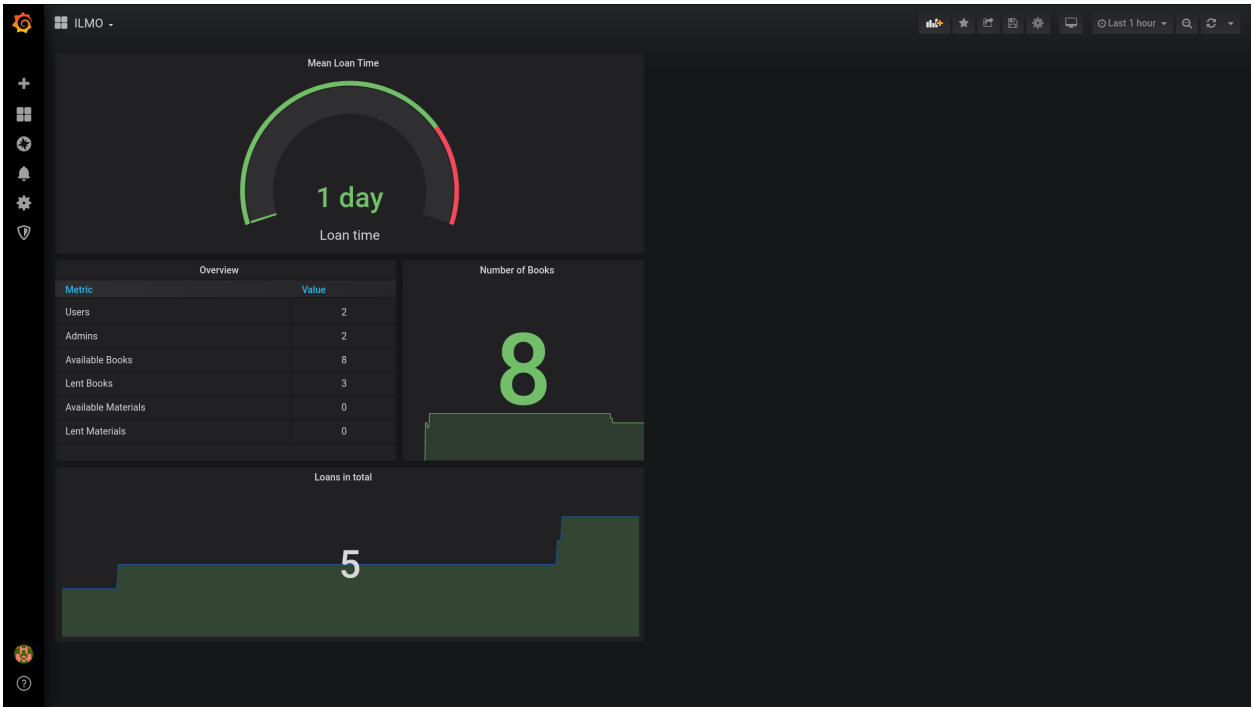
(continues on next page)

(continued from previous page)

```
  interval = "10s"
  round_interval = true
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"

# Configuration for sending metrics to InfluxDB
[[outputs.influxdb]]
  urls = ["http://:::8086"]
  database = "telegraf"
  skip_database_creation = true
  username = 'telegraf'
  password = 'yourpassword'

[[inputs.http]]
  urls = ["https://example.com/library/metrics/"]
  name_override = "ilmo"
  #Data from HTTP in JSON format
  data_format = "json"

```

Graphana

Now we can simply use the InfluxDB as data source in Grafana and configure until you have beautiful plots!



CHAPTER

THREE

INSTALLATION, CUSTOMIZATION AND CONTRIBUTING

3.1 Deployment

There are different ways to deploy ILMO. We support an ansible+docker based deployment and manual installation.

3.1.1 Ansible deployment

ILMO can be deployed with the ilmo-ansible-role that is based on the official ILMO docker image. This role will only install ilmo itself. If you want a complete setup that includes a database and a webserver with minimal configuration you can use the mash-playbook by following it's documentation on ILMO.

3.1.2 Manual Deployment

This guide describes the installation of a installation of ILMO from source. It is inspired by this great guide from pretix.

> **Warning:** Even though this guide tries to make it as straightforward to run ILMO, it still requires some Linux experience to get it right. If you're not feeling comfortable managing a Linux server, check out a managed service.

This guide is tested on **Ubuntu20.04** but it should work very similar on other modern systemd based distributions.

Requirements

Please set up the following systems beforehand, it will not be explained here in detail (but see these links for external installation guides):

- A SMTP server to send out mails, e.g. Postfix on your machine or some third-party server you have credentials for
- A HTTP reverse proxy, e.g. nginx or Traefik to allow HTTPS connections
- A PostgreSQL database server

Also recommended is, that you use a firewall, although this is not a ILMO-specific recommendation. If you're new to Linux and firewalls, it is recommended that you start with ufw.

Note: Please, do not run ILMO without HTTPS encryption. You'll handle user data and thanks to Let's Encrypt SSL certificates can be obtained for free these days.

Unix user

As we do not want to run ilmo as root, we first create a new unprivileged user:

```
# adduser ilmo --disabled-password --home /var/ilmo
```

In this guide, all code lines prepended with a # symbol are commands that you need to execute on your server as `root` user (e.g. using `sudo`); all lines prepended with a $ symbol should be run by the unprivileged user.

Database

Having the database server installed, we still need a database and a database user. We can create these with any kind of database managing tool or directly on our database's shell. Please make sure that UTF8 is used as encoding for the best compatibility. You can check this with the following command:

```
# sudo -u postgres psql -c 'SHOW SERVER_ENCODING'
```

For PostgreSQL database creation, we would do:

```
# sudo -u postgres createuser ilmo
# sudo -u postgres createdb -O ilmo ilmo
# su ilmo
$ psql
> ALTER USER ilmo PASSWORD 'strong_password';
```

Package dependencies

To build and run ilmo, you will need the following debian packages:

```
# apt-get install git build-essential python-dev python3-venv python3 python3-pip \
                  python3-dev
```

Config file

We now create a config directory and config file for ilmo:

```
# mkdir /etc/ilmo
# touch /etc/ilmo/ilmo.cfg
# chown -R ilmo:ilmo /etc/ilmo/
# chmod 0600 /etc/ilmo/ilmo.cfg
```

Fill the configuration file `/etc/ilmo/ilmo.cfg` with the following content (adjusted to your environment):

```
[ilmo]
instance_name=My library
url=https://ilmo.example.com

[database]
backend=postgresql
name=ilmo
user=ilmo
```

(continues on next page)

(continued from previous page)

```
[locations]
static=/var/ilmo/static

[mail]
; See config file documentation for more options
; from=ilmo@example.com
; host=127.0.0.1
; user=ilmo
; password=foobar
; port=587

[security]
; See https://securitytxt.org/ for reference
;Contact=
;Expires=
;Encryption=
;Preferred-Languages=
;Scope=
;Policy=
```

Install ilmo as package

Now we will install ilmo itself. The following steps are to be executed as the `ilmo` user. Before we actually install ilmo, we will create a virtual environment to isolate the python packages from your global python installation:

```
$ python3 -m venv /var/ilmo/venv
$ source /var/ilmo/venv/bin/activate
(venv)$ pip3 install -U pip setuptools wheel
```

We now clone and install ilmo, its direct dependencies and gunicorn:

```
(venv)$ git clone https://github.com/moan0s/ILMO2
(venv)$ cd ILMO2/src/
(venv)$ pip3 install -r requirements.txt
(venv)$ pip3 install -e .
```

Note that you need Python 3.6 or newer. You can find out your Python version using `python -V`.

Finally, we compile static files and create the database structure:

```
(venv)$ ./manage.py collectstatic
(venv)$ ./manage.py migrate
(venv)$ django-admin compilemessages --ignore venv
```

Start ilmo as a service

You should start ilmo using systemd to automatically start it after a reboot. Create a file named `/etc/systemd/system/ilmo-web.service` with the following content:

```
[Unit]
Description=ilmo web service
After=network.target

[Service]
User=ilmo
Group=ilmo
Environment="VIRTUAL_ENV=/var/ilmo/venv"
Environment="PATH=/var/ilmo/venv/bin:/usr/local/bin:/usr/bin:/bin"
ExecStart=/var/ilmo/venv/bin/gunicorn ilmo.wsgi \
                      --name ilmo --workers 5 \
                      --max-requests 1200  --max-requests-jitter 50 \
                      --log-level=info --bind=127.0.0.1:8345
WorkingDirectory=/var/ilmo
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

You can now run the following commands to enable and start the services:

```
# systemctl daemon-reload
# systemctl enable ilmo-web
# systemctl start ilmo-web
```

SSL

The following snippet is an example on how to configure a nginx proxy for ilmo:

```
server {
        listen 80;
        listen [::]:80;

        if ($scheme = http) {
                return 301 https://$server_name$request_uri;
        }

        #
        listen 443 ssl;
        listen [::]:443 ssl;
        ssl_certificate     /etc/letsencrypt/live/ilmo.example.com/cert.pem;
        ssl_certificate_key /etc/letsencrypt/live/ilmo.example.com/privkey.pem;
        ssl_protocols       TLSv1.2 TLSv1.3;
        ssl_ciphers         HIGH:!aNULL:!MD5;


    # Set header
    add_header X-Clacks-Overhead "GNU Terry Pratchett";
```

(continues on next page)

(continued from previous page)

```
    add_header Permissions-Policy interest-cohort=(); #Anti FLoC
    add_header Referrer-Policy same-origin;
    add_header X-Content-Type-Options nosniff;

        server_name ilmo.example.com;
    location / {
        proxy_pass http://localhost:8345;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header Host $http_host;
    }

    location /static/ {
        alias /var/ilmo/static/;
        access_log off;
        expires 365d;
        add_header Cache-Control "public";
    }
}
```

We recommend reading about setting strong encryption settings for your web server.

Next steps

Yay, you are done! You should now be able to reach ilmo at https://ilmo.example.com/

Updates

> **Warning:** While we try hard not to break things, **please perform a backup before every upgrade**.

To upgrade to a new ilmo release, pull the latest code changes and run the following commands:

```
$ source /var/ilmo/venv/bin/activate
(venv)$ git pull
(venv)$ pg_dump ilmo > ilmo.psql
(venv)$ python manage.py migrate
(venv)$ django-admin compilemessages --ignore venv

# systemctl restart ilmo-web
```

3.2 Contributing

3.2.1 Report a bug

To report a bug, file an issue on Github

Try to include the following information:

- The information needed to reproduce the problem
- What you would expect to happen
- What did actually happen
- Error messages

You are also invited to include:

- Screenshots
- Which browser you are using
- The URL of the site
- How urgent it is
- Any additional information you consider useful

3.2.2 Get involved!

To contribute simply clone the directory, make your changes and file a pull request.

If you want to know what can be done, have a look at the current Issues.

3.2.3 Get in touch!

If you have questions, want to contribute or want to message me regarding something else you can find contact information at https://hyteck.de/about/ or directly write an E-Mail

3.3 Release

3.3.1 What qualifies as release?

A new release should be announced when a significant number functions, bugfixes or other improvements to the software is made. Usually this indicates a minor release. Major releases are yet to be determined.

3.3.2 What should be done before a release?

Tested basic functions

Run **`pytest`** and manually test

- Installation
- User, book and material management (create, view, change, delete)
- Logging (e.g. mail log)
- E-Mails
- Monitoring

Test upgrade on a copy of a production database

> **Warning:** You have to prevent e-mails from being sent, otherwise users could receive duplicate e-mails!

- Ensure correct conversion if necessary
- Views correct?

3.3.3 Release

After testing everything you are good to go. Open the file `src/setup.py` with a text editor you can adjust the version number:

Do a final commit on this change, and tag the commit as release with appropriate version number.

```
git tag -a v1.0.0 -m "Releasing version v1.0.0"
git push origin v1.0.0
```

Make sure the tag is visible on GitHub and celebrate

3.4 Backup & Restore

If you do no heavy modification of the code you should be fine with backing up `/etc/ilmo/` and the database. Assuming you used a PostgreSQL database the following solution might help you with backups and restores.

3.4.1 Backup

The following code is a modification of this script licensed under the postgresql_license.

You will first need to create a backup configuration at `/var/ilmo/pg_backup.config`.

```
##############################
## POSTGRESQL BACKUP CONFIG ##
##############################
```

(continues on next page)

(continued from previous page)

```
# Optional system user to run backups as.  If the user the script is running as doesn't␣
↪match this
# the script terminates.  Leave blank to skip check.
BACKUP_USER=ilmo

# Optional hostname to adhere to pg_hba policies.  Will default to "localhost" if none␣
↪specified.
HOSTNAME=localhost

# Optional username to connect to database as.  Will default to "postgres" if none␣
↪specified.
USERNAME=ilmo

# This dir will be created if it doesn't exist.  This must be writable by the user the␣
↪script is
# running as.
BACKUP_DIR=/var/ilmo/backups/postgresql

# Enter database to backup
DATABSE=ilmo


#### SETTINGS FOR ROTATED BACKUPS ####

# Which day to take the weekly backup from (1-7 = Monday-Sunday)
DAY_OF_WEEK_TO_KEEP=7

# Number of days to keep daily backups
DAYS_TO_KEEP=7

# How many weeks to keep weekly backups
WEEKS_TO_KEEP=5

######################################
```

And then add the script that will do the actual backup at `/var/ilmo/backup_rotate.sh`

```
#!/bin/bash

############################
####### LOAD CONFIG #######
############################

while [ $# -gt 0 ]; do
        case $1 in
                -c)
                        CONFIG_FILE_PATH="$2"
                        shift 2
                        ;;
                *)
                        ${ECHO} "Unknown Option \"$1\"" 1>&2
                        exit 2
```

(continues on next page)

(continued from previous page)

```
                        ;;
        esac
done

if [ -z $CONFIG_FILE_PATH ] ; then
        SCRIPTPATH=$(cd ${0%/*} && pwd -P)
        CONFIG_FILE_PATH="${SCRIPTPATH}/pg_backup.config"
fi

if [ ! -r ${CONFIG_FILE_PATH} ] ; then
        echo "Could not load config file from ${CONFIG_FILE_PATH}" 1>&2
        exit 1
fi

source "${CONFIG_FILE_PATH}"

############################
#### PRE-BACKUP CHECKS ####
############################

# Make sure we're running as the required backup user
if [ "$BACKUP_USER" != "" -a "$(id -un)" != "$BACKUP_USER" ] ; then
        echo "This script must be run as $BACKUP_USER. Exiting." 1>&2
        exit 1
fi


############################
### INITIALISE DEFAULTS ###
############################

if [ ! $HOSTNAME ]; then
        HOSTNAME="localhost"
fi;

if [ ! $USERNAME ]; then
        USERNAME="postgres"
fi;


############################
#### START THE BACKUPS ####
############################

function perform_backups()
{
        SUFFIX=$1
        FINAL_BACKUP_DIR=$BACKUP_DIR"`date +\%Y-\%m-\%d`$SUFFIX/"

        echo "Making backup directory in $FINAL_BACKUP_DIR"

        if ! mkdir -p $FINAL_BACKUP_DIR; then
```

(continues on next page)

(continued from previous page)

```
                echo "Cannot create backup directory in $FINAL_BACKUP_DIR. Go and fix it!
↪" 1>&2
                exit 1;
        fi;

        ########################
        ### GLOBALS BACKUPS ###
        ########################

        echo -e "\n\nPerforming backup"
        echo -e "-------------------------------------------\n"

        echo "Backup"

        set -o pipefail
        if ! pg_dump $DATABASE | gzip > $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress;␣
↪then
                echo "[!!ERROR!!] Failed to produce globals backup" 1>&2
        else
                mv $FINAL_BACKUP_DIR"$DATABASE".sql.gz.in_progress $FINAL_BACKUP_DIR"
↪$DATABSE".sql.gz
        fi
        set +o pipefail

        echo -e "\nAll database backups complete!"
}

# MONTHLY BACKUPS

DAY_OF_MONTH=`date +%d`

if [ $DAY_OF_MONTH -eq 1 ];
then
        # Delete all expired monthly directories
        find $BACKUP_DIR -maxdepth 1 -name "*-monthly" -exec rm -rf '{}' ';'

        perform_backups "-monthly"

        exit 0;
fi

# WEEKLY BACKUPS

DAY_OF_WEEK=`date +%u` #1-7 (Monday-Sunday)
EXPIRED_DAYS=`expr $((($WEEKS_TO_KEEP * 7) + 1))`

if [ $DAY_OF_WEEK = $DAY_OF_WEEK_TO_KEEP ];
then
        # Delete all expired weekly directories
        find $BACKUP_DIR -maxdepth 1 -mtime +$EXPIRED_DAYS -name "*-weekly" -exec rm -rf
↪'{}' ';'
```

(continues on next page)

(continued from previous page)

```
        perform_backups "-weekly"

        exit 0;
fi

# DAILY BACKUPS

# Delete daily backups 7 days old or more
find $BACKUP_DIR -maxdepth 1 -mtime +$DAYS_TO_KEEP -name "*-daily" -exec rm -rf '{}' ';'

perform_backups "-daily"
```

You should make the script executable test it and automate the execution with **crontab**

```
$ chmod +x backup_rotate.sh
$ ./backup_rotate.sh
$ crontab -e
# enter the following to backup every day at 3am
0 3 * * * /var/ilmo/backup_rotate.sh
```

3.4.2 Restore

If you for any reason want to restore a backup you can use the following:

```
$ sudo systemctl stop ilmo-web
$ pg_dump ilmo > ilmo_YYYY_MM_DD-hh_mm.psql # Make a backup for later analysis
$ dropdb ilmo
$ cd /path/to/backup
$ gzip -d ilmo.sql.gz
$ sudo -u postgres createdb -O ilmo ilmo
$ psql ilmo < ilmo.sql
$ systemctl restart ilmo-web
```

CHAPTER

FOUR

API DOCUMENTATION

ILMOs API serves the purpose of supporting 3rd-person applications and anything you can think of basically.

Warning: The current API is limited in it's functionality. I you miss a specific feature please contact the developer!

4.1 API Access

4.1.1 Via browser

When a user is logged in, they can easily access the API in their browser, authenticated by their session. The API endpoint can be found at /library/api/ http://example.com:8000/library/api/book

4.1.2 Via token

All users are able to generate a token that allows them to use the API. This can be done in the user's profile. An application can then send this token in the request header for authorization.

4.2 Access Control

The API allows to query permissions to access rooms. E.g. a IoT device could query this information and decided to open a door or a key locker. Currently the user can be queried via a UID.

A typical request Looks like this: .. code-block:

```
GET /library/api/uid/1234456/room/dbc71599-a0ce-482f-a896-6f4a7dfc17ec
```

Read the manual, Image by Mike Powell (CC-BY).