BibMgR

A BibTeX reference manager for collecting, validating, editing, and exporting bibliography data.

Published: Aug 20, 2026

BibMgR is a BibTeX reference manager for collecting, validating, editing, and exporting bibliography data. It provides a shared authenticated web library with revision history, a CLI, and a Socket Mode Slack app built on the same source-preserving Rust core.

Prerequisites

Developing, testing, or building this repository from source requires the following tools:

- `uv`: provisions the Python environment, installs locked dependencies, and runs Poe tasks; the application supports Python 3.11–3.13, while `.python-version` selects 3.12 only as the default contributor version
- Rust 1.86 or later (`rustc` and `cargo`): builds the CLI and Python native extension
- Node.js 22.12 or later in the Node.js 22 release line, with npm 10 or later: manages frontend dependencies, the development server, tests, and production builds

Setup

Install the locked dependencies:

```
uv sync
uv run poe setup
```

List the available tasks with:

```
uv run poe --help
```

CLI installation (optional)

To use BibTeX validation from the command line, install the CLI directly from GitHub without cloning the repository:

```
cargo install --git https://github.com/EhimeNLP/bibmgr.git --locked bibmgr-cli
bibmgr --version
```

Add `--force` to replace an existing Cargo installation:

```
cargo install --git https://github.com/EhimeNLP/bibmgr.git --locked --force bibmgr-cli
```

Alternatively, Linux and macOS users can install a prebuilt binary without requiring Rust:

```
curl -LsSf https://github.com/EhimeNLP/bibmgr/releases/latest/download/bibmgr-installer.sh | sh
```

The installer verifies the release archive with SHA-256 and writes the `bibmgr` executable to `~/.local/bin` by default. If that directory is not already on `PATH`, add it in your shell configuration:

```
export PATH="$HOME/.local/bin:$PATH"
```

Set `BIBMGR_INSTALL_DIR` to select another destination, or `BIBMGR_VERSION` to install a specific release. A version may be written with or without the leading `v`.

```
curl -LsSf https://github.com/EhimeNLP/bibmgr/releases/latest/download/bibmgr-installer.sh |
  BIBMGR_INSTALL_DIR="$HOME/bin" BIBMGR_VERSION=v0.1.0 sh
```

Run the installer again to update an existing installation. Windows x86_64 users can download `bibmgr-x86_64-pc-windows-msvc.zip` from the latest release, extract `bibmgr.exe`, and place it on `PATH`.

If you clone the repository for development or inspection, you can install from the local path:

```
git clone https://github.com/EhimeNLP/bibmgr.git
cd bibmgr
cargo install --locked --path crates/bibmgr-cli
```

To update an installation from a local checkout, update the checkout and reinstall:

```
git pull
cargo install --locked --force --path crates/bibmgr-cli
```

To build a release binary without installing it, use the Poe task:

```
uv run --frozen poe build-cli
./target/release/bibmgr --version
```

CLI usage

The following examples cover the basic commands. Use `bibmgr COMMAND --help` for detailed options, and see `docs/cli.md` for exit codes and the JSON output contract.

```
# Lint
bibmgr lint references.bib

# Emit JSON for CI
bibmgr lint references.bib --format json

# Preview safe, source-preserving fixes
bibmgr fix references.bib --safe --dry-run

# Export a separate file using the default modern profile
bibmgr export references.bib --output references.exported.bib

# Return the export result as a JSON DTO
bibmgr export references.bib --profile classical-bst --format json

# Inspect the semantic AST
bibmgr inspect references.bib --ast
```

Slack app

The Slack app accepts a BibTeX entry in a channel mention or direct message, asks the user to choose an export profile, applies safe fixes, and returns the formatted entry with any remaining warnings. It runs in Socket Mode and is deployed separately from the web application.

Create the Slack app from the bundled manifest, then use the interactive Docker task for local operation:

```
uv run poe slack
```

Bot messages default to English. Select Japanese at startup with:

```
BIBMGR_SLACK_LANGUAGE=ja uv run poe slack
```

For an unattended production deployment, build the image separately and start it with both Slack tokens supplied by the deployment environment:

```
uv run poe slack-build
uv run poe slack-up
```

See `docs/slack.md` for the Slack website setup, token options, usage, and deployment-only profiles.

Initialization pipelines

The optional out-of-band initialization pipeline is separate from the application dependency graph. See the [pipeline overview](pipeline/README.md) for the two-stage workflow, shared contracts, and boundaries; the stage-specific guides cover [metadata extraction](pipeline/metadata_extraction/README.md) and [BibTeX reconstruction](pipeline/bibtex_reconstruction/README.md). Neither stage registers records in the shared BibMgR library automatically.

Development

The reference library API uses PostgreSQL 18. Mailpit provides the development inbox for email authentication. If Docker is available, start both services and apply the database migrations:

```
uv run poe dev-services-up
uv run poe db-migrate
```

Open the Mailpit inbox at `http://127.0.0.1:8025/`. Unless configured otherwise, the development backend sends authentication email through `127.0.0.1:1025`.

Docker is optional. On macOS, PostgreSQL 18 and Mailpit can run directly through Homebrew. See `docs/local-development.md` for instructions covering local database creation, the first account, BibTeX registration, and service shutdown.

To reset the development database to an empty current schema, run the following task and type the displayed database name to confirm. The task refuses to reset a remote database.

```
uv run poe db-reset
```

Set `BIBMGR_DATABASE_URL` to change the database connection. It defaults to `postgresql+psycopg://bibmgr:bibmgr@127.0.0.1:5432/bibmgr`. The server-owned `BIBMGR_REGISTRATION_POLICY` selects the registration policy and defaults to the source-preserving `archive` policy. Default analysis and export use `modern`; deployment-specific output conventions belong in a custom export profile. `BIBMGR_AUTH_EMAIL_DOMAIN` selects the exact permitted login domain; local development defaults to the reserved domain `example.test`, while production requires an explicit value.

```
BIBMGR_DATABASE_URL=postgresql+psycopg://user:password@db.example/bibmgr \
BIBMGR_REGISTRATION_POLICY=archive \
uv run poe db-migrate
```

Start the backend and frontend development servers together:

```
uv run poe dev
```

- Frontend: `http://127.0.0.1:5173/`
- Backend: `http://127.0.0.1:8000/`
- Health check: `http://127.0.0.1:8000/healthz`
- Readiness check: `http://127.0.0.1:8000/readyz`
- API documentation: `http://127.0.0.1:8000/docs`
- Development email inbox: `http://127.0.0.1:8025/`

After login, the `History` view shows sequential per-reference revisions, including edits and deletions. Deleted references remain in the history index and can be reviewed and restored from a selected earlier state. A restore appends a new revision without rewriting existing history.

To permit a user outside the configured login domain, add their complete email address to `BIBMGR_AUTH_ALLOWED_EMAILS`. Domain entries and wildcards are not accepted.

To run the servers separately, use two terminals:

```
uv run poe dev-backend
uv run poe dev-frontend
```

The listening addresses and ports can be changed with environment variables:

```
HOST=127.0.0.1 PORT=8000 \
FRONTEND_HOST=127.0.0.1 FRONTEND_PORT=5173 \
uv run poe dev
```

Run the complete test suite and the integrated checks, including formatting, linting, type checking, lockfile, schema, Markdown, and fuzz/benchmark builds, with:

```
uv run poe test
uv run poe check
```

Production deployment

Build the CLI, native extension wheel, backend wheel, and frontend static files from the locked dependencies:

```
uv run --frozen poe build
```

The artifacts are written to:

- CLI: `target/release/bibmgr`
- Native wheel: `dist/native/*.whl`
- Backend wheel: `dist/backend/*.whl`
- Frontend static files: `frontend/dist/`

To run the backend from a source checkout, synchronize the environment without modifying the lockfile, then use the production-oriented task that disables automatic reload:

```
uv sync --frozen
HOST=0.0.0.0 PORT=8000 uv run --frozen poe start-backend
```

To deploy only the wheels to a runtime host, install the native extension and backend wheels into a supported Python 3.11–3.13 environment and start the application. The following example uses Python 3.11, matching the production container:

```
uv venv --python 3.11 .venv-runtime
uv pip install \
  --python .venv-runtime/bin/python \
  dist/native/*.whl \
  dist/backend/*.whl
.venv-runtime/bin/python -m uvicorn bibmgr_backend.app:app \
  --host 0.0.0.0 \
  --port 8000
```

Serve `frontend/dist/` from a static file server or CDN, and configure a reverse proxy that forwards `/api/` to the backend.

`compose.production.yaml` contains the shared production configuration for PostgreSQL 18, the migration job, the backend, and Vue/Caddy. Choose either `compose.production.direct.yaml`, where Caddy terminates TLS directly, or `compose.production.proxy.yaml`, where an external reverse proxy terminates TLS.

```
cp .env.production.example .env.production

# Publish HTTPS directly through Caddy
uv run poe prod-direct-config
uv run poe prod-direct-up

# Run behind an external reverse proxy
uv run poe prod-proxy-config
uv run poe prod-proxy-up
```

The production backend requires `BIBMGR_ENV=production`, a secret file, SMTP connection settings, secure cookies, and HTTPS. See `docs/operations.md` for account management, periodic authentication-data cleanup, monitoring, backup/restore, and systemd timers. See `docs/authentication.md` for the authentication contract.

License

The BibMgR application and libraries are distributed under the MIT License.