@stack.gallery/registry

the stack.gallery registry

- readme.md for @stack.gallery/registry
- changelog.md for @stack.gallery/registry

readme.md for @stack.gallery/registry

A self-hosted, multi-protocol package registry built with Deno and TypeScript. Run your own private **NPM**, **Docker/OCI**, **Maven**, **Cargo**, **PyPI**, **Composer**, and **RubyGems** registry — all behind a single binary with a modern web UI.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

□ Features

- □□ **7 Protocol Support** — NPM, OCI/Docker, Maven, Cargo, PyPI, Composer, RubyGems via `@push.rocks/smartregistry`
- □□ **Organizations & Teams** — Hierarchical access control: orgs → teams → repositories
- □□ **Flexible Authentication** — Local JWT auth, OAuth/OIDC, and LDAP with JIT user provisioning
- □□ **Scoped API Tokens** — Per-protocol, per-scope tokens (`srg_` prefix) for CI/CD pipelines
- □□ **RBAC Permissions** — Reader → Developer → Maintainer → Admin per repository
- □□ **Upstream Caching** — Transparently proxy and cache packages from public registries
- □□ **Audit Logging** — Full audit trail on every action for compliance
- □□ **Modern Web UI** — Web Components dashboard built with `@design.estate/dees-catalog`, bundled into the binary
- ⚡ **Single Binary** — Cross-compiled with `deno compile` for Linux and macOS (x64 + ARM64)
- □□ **MongoDB + S3** — Metadata in MongoDB, artifacts in any S3-compatible store

□□ Quick Start

Prerequisites

- **MongoDB** >= 4.4
- **S3-compatible storage** (MinIO, AWS S3, Cloudflare R2, etc.)

Install from Binary

```
# One-liner install (latest version)
curl -sSL https://code.foss.global/stack.gallery/registry/raw/branch/main/install.sh | sudo bash

# Install specific version
curl -sSL https://code.foss.global/stack.gallery/registry/raw/branch/main/install.sh | sudo bash -s -- --version v1.8.0

# Install + set up systemd service
curl -sSL https://code.foss.global/stack.gallery/registry/raw/branch/main/install.sh | sudo bash -s -- --setup-service
```

The installer:

- Detects your platform (Linux/macOS, x64/ARM64)
- Downloads the pre-compiled binary from Gitea releases
- Installs to `/opt/stack-gallery-registry/` with a symlink in `/usr/local/bin/`
- Optionally creates and enables a systemd service

Run from Source

```
# Clone
git clone https://code.foss.global/stack.gallery/registry.git
cd registry

# Install Node dependencies (for tsbundle/tsdeno build tools)
pnpm install

# Development mode (hot reload, reads .nogit/env.json)
deno task dev

# Production mode
```

```
deno task start
```

The registry is available at `http://localhost:3000`.

⚙ Configuration

Configuration is loaded from **environment variables** (production) or from `.nogit/env.json` when using the `--ephemeral` flag (development).

Variable	Default	Description
`MONGODB_URL`	`mongodb://localhost:27017`	MongoDB connection string
`MONGODB_DB`	`stackgallery`	Database name
`S3_ENDPOINT`	`http://localhost:9000`	S3-compatible endpoint
`S3_ACCESS_KEY`	`minioadmin`	S3 access key
`S3_SECRET_KEY`	`minioadmin`	S3 secret key
`S3_BUCKET`	`registry`	S3 bucket name
`S3_REGION`	—	S3 region
`HOST`	`0.0.0.0`	Server bind address
`PORT`	`3000`	Server port
`JWT_SECRET`	`change-me-in-production`	JWT signing secret
`AUTH_ENCRYPTION_KEY`	*(ephemeral)*	64-char hex for AES-256-GCM encryption of OAuth/LDAP secrets
`STORAGE_PATH`	`packages`	Base path in S3 for artifacts
`ENABLE_UPSTREAM_CACHE`	`true`	Cache packages from upstream registries
`UPSTREAM_CACHE_EXPIRY`	`24`	Cache TTL in hours

Example `.nogit/env.json`:

```
{
  "MONGODB_URL": "mongodb://admin:pass@localhost:27017/stackregistry?authSource=admin",
  "MONGODB_NAME": "stackregistry",
  "S3_HOST": "localhost",
  "S3_PORT": "9000",
  "S3_ACCESSKEY": "minioadmin",
  "S3_SECRETKEY": "minioadmin",
```

```
  "S3_BUCKET": "registry",
  "S3_USESSL": false
}
```

□□ Protocol Endpoints

Each protocol is handled natively via `@push.rocks/smartregistry`. Point your package manager at the registry:

Protocol	Paths	Client Config Example
NPM	`/-/npm/{org}/*`	`npm config set registry http://registry:3000/-/npm/myorg/`
OCI/Docker	`/v2/*`	`docker login registry:3000`
Maven	`/maven2/*`	Add repository URL in `pom.xml`
Cargo	`/api/v1/crates/*`	Configure in `.cargo/config.toml`
PyPI	`/simple/*`, `/pypi/*`	`pip install --index-url http://registry:3000/simple/`
Composer	`/packages.json`, `/p/*`	Add repository in `composer.json`
RubyGems	`/api/v1/gems/*`, `/gems/*`	`gem sources -a http://registry:3000`

Authentication works with **Bearer tokens** (API tokens prefixed `srg_`) and **Basic auth** (email:password or username:token).

NPM Usage Example

```
# Configure npm to use your org's registry
npm config set @myorg:registry http://localhost:3000/-/npm/myorg/

# Authenticate
echo "//localhost:3000/-/npm/myorg/:_authToken=srg_YOUR_TOKEN" >> ~/.npmrc

# Publish & install as usual
npm publish
npm install @myorg/my-package
```

Docker/OCI Usage Example

```
# Login
docker login localhost:3000

# Tag and push
docker tag myimage:latest localhost:3000/myorg/myimage:1.0.0
docker push localhost:3000/myorg/myimage:1.0.0

# Pull
docker pull localhost:3000/myorg/myimage:1.0.0
```

□□ Authentication & Security

Local Auth

- JWT-based with **15-minute access tokens** and **7-day refresh tokens** (HS256)
- Session tracking — each login creates a session, tokens embed session IDs
- Password hashing with PBKDF2 (10,000 rounds SHA-256 + random salt)

External Auth (OAuth/OIDC & LDAP)

- **OAuth/OIDC** — Connect to any OIDC-compliant provider (Keycloak, Okta, Auth0, Azure AD, etc.)
- **LDAP** — Bind + search authentication against Active Directory or OpenLDAP
- **JIT Provisioning** — Users are auto-created on first external login
- **Auto-linking** — External identities are linked to existing users by email match
- **Encrypted secrets** — Provider client secrets and bind passwords are stored AES-256-GCM encrypted

RBAC Permissions

Access is resolved through a hierarchy:

```
Platform Admin (full access)
  └─ Organization Owner/Admin
       └─ Team Maintainer (read + write + delete on team repos)
            └─ Team Member (read + write on team repos)
                 └─ Direct Repo Permission (reader / developer / maintainer / admin)
```

```
└─ Public Repository (read for everyone)
```

Scoped API Tokens

Tokens are prefixed with `srg_` and can be scoped to:

- Specific **protocols** (e.g., npm + oci only)
- Specific **actions** (read / write / delete)
- Specific **organizations**
- Custom **expiration** dates

□□ REST API

All management endpoints live under `/api/v1/`. Authenticated via `Authorization: Bearer <jwt_or_api_token>`.

Auth

Method	Endpoint	Description
`POST`	`/api/v1/auth/login`	Login (email + password)
`POST`	`/api/v1/auth/refresh`	Refresh access token
`POST`	`/api/v1/auth/logout`	Logout (invalidate session)
`GET`	`/api/v1/auth/me`	Current user info
`GET`	`/api/v1/auth/providers`	List active external auth providers
`GET`	`/api/v1/auth/oauth/:id/authorize`	Initiate OAuth flow
`GET`	`/api/v1/auth/oauth/:id/callback`	OAuth callback
`POST`	`/api/v1/auth/ldap/:id/login`	LDAP login

Users

Method	Endpoint	Description
`GET`	`/api/v1/users`	List users
`POST`	`/api/v1/users`	Create user
`GET`	`/api/v1/users/:id`	Get user

Method	Endpoint	Description
`PUT`	`/api/v1/users/:id`	Update user
`DELETE`	`/api/v1/users/:id`	Delete user

Organizations

Method	Endpoint	Description
`GET`	`/api/v1/organizations`	List organizations
`POST`	`/api/v1/organizations`	Create organization
`GET`	`/api/v1/organizations/:id`	Get organization
`PUT`	`/api/v1/organizations/:id`	Update organization
`DELETE`	`/api/v1/organizations/:id`	Delete organization
`GET`	`/api/v1/organizations/:id/members`	List members
`POST`	`/api/v1/organizations/:id/members`	Add member
`PUT`	`/api/v1/organizations/:id/members/:userId`	Update member role
`DELETE`	`/api/v1/organizations/:id/members/:userId`	Remove member

Repositories

Method	Endpoint	Description
`GET`	`/api/v1/organizations/:orgId/repositories`	List org repos
`POST`	`/api/v1/organizations/:orgId/repositories`	Create repo
`GET`	`/api/v1/repositories/:id`	Get repo
`PUT`	`/api/v1/repositories/:id`	Update repo
`DELETE`	`/api/v1/repositories/:id`	Delete repo

Packages

Method	Endpoint	Description
`GET`	`/api/v1/packages`	Search packages
`GET`	`/api/v1/packages/:id`	Get package details

Method	Endpoint	Description
`GET`	`/api/v1/packages/:id/versions`	List versions
`DELETE`	`/api/v1/packages/:id`	Delete package
`DELETE`	`/api/v1/packages/:id/versions/:version`	Delete version

Tokens

Method	Endpoint	Description
`GET`	`/api/v1/tokens`	List your tokens
`POST`	`/api/v1/tokens`	Create token
`DELETE`	`/api/v1/tokens/:id`	Revoke token

Audit

Method	Endpoint	Description
`GET`	`/api/v1/audit`	Query audit logs

Admin (Platform Admins Only)

Method	Endpoint	Description
`GET`	`/api/v1/admin/auth/providers`	List all auth providers
`POST`	`/api/v1/admin/auth/providers`	Create auth provider
`GET`	`/api/v1/admin/auth/providers/:id`	Get provider details
`PUT`	`/api/v1/admin/auth/providers/:id`	Update provider
`DELETE`	`/api/v1/admin/auth/providers/:id`	Disable provider
`POST`	`/api/v1/admin/auth/providers/:id/test`	Test provider connection
`GET`	`/api/v1/admin/auth/settings`	Get platform settings
`PUT`	`/api/v1/admin/auth/settings`	Update platform settings

Health Check

Method	Endpoint	Description

`GET`	`/health` or `/healthz`	Returns JSON status of MongoDB, S3, and registry

Architecture

```
registry/
├── mod.ts                    # Deno entry point
├── deno.json                 # Deno config, tasks, imports
├── package.json              # Node deps (tsbundle, tsdeno, tswatch)
├── npmextra.json             # tsdeno compile targets & gitzone config
├── install.sh                # Binary installer script
├── .gitea/workflows/         # CI release pipeline
├── ts/
│   ├── registry.ts           # StackGalleryRegistry — main orchestrator
│   ├── cli.ts                # CLI commands (smartcli)
│   ├── plugins.ts            # Centralized dependency imports
│   ├── api/
│   │   ├── router.ts         # REST API router with JWT/token auth
│   │   └── handlers/         # auth, user, org, repo, package, token, audit, oauth, admin
│   ├── opsserver/            # TypedRequest RPC handlers
│   ├── models/               # MongoDB models via @push.rocks/smartdata
│   │   ├── user.ts, organization.ts, team.ts
│   │   ├── repository.ts, package.ts
│   │   ├── apitoken.ts, session.ts, auditlog.ts
│   │   ├── auth.provider.ts, external.identity.ts, platform.settings.ts
│   │   └── *.member.ts, *.permission.ts
│   ├── services/             # Business logic
│   │   ├── auth.service.ts           # JWT login/refresh/logout
│   │   ├── external.auth.service.ts  # OAuth/OIDC & LDAP flows
│   │   ├── crypto.service.ts         # AES-256-GCM encryption
│   │   ├── token.service.ts          # API token CRUD
│   │   ├── permission.service.ts     # RBAC resolution
│   │   └── audit.service.ts          # Audit logging
│   ├── providers/            # smartregistry integration
│   │   ├── auth.provider.ts          # IAuthProvider implementation
│   │   └── storage.provider.ts       # IStorageHooks for quota/audit
│   └── interfaces/           # TypeScript interfaces & types
```

```
└── ts_interfaces/          # Shared API contract (TypedRequest interfaces)
    ├── data/               # Data types (auth, org, repo, package, token, audit, admin)
    └── requests/           # Request/response interfaces for all API endpoints
```

□□ Technology Stack

Component	Technology
Runtime	Deno 2.x
Language	TypeScript (strict mode)
Database	MongoDB via `@push.rocks/smartdata`
Storage	S3 via `@push.rocks/smartbucket`
Registry Core	`@push.rocks/smartregistry`
Frontend	Web Components via `@design.estate/dees-element` + `@design.estate/dees-catalog`
UI Build	`@git.zone/tsbundle`
Auth	JWT (HS256) + OAuth/OIDC + LDAP
Build	`@git.zone/tsdeno` cross-compilation
CI/CD	Gitea Actions → binary releases

□□ Development

Commands

```
# Start dev server with hot reload (reads .nogit/env.json)
deno task dev

# Watch mode: backend + UI + bundler concurrently
pnpm run watch

# Build UI (web components via tsbundle)
deno task build-ui

# Cross-compile binaries for all platforms
```

```
deno task compile

# Type check / format / lint
deno task check
deno task fmt
deno task lint

# Run tests
deno task test              # All tests
deno task test:unit         # Unit tests only
deno task test:integration  # Integration tests (requires running server)
deno task test:e2e          # E2E tests (requires running server + services)
```

Build & Release

Releases are automated via Gitea Actions (`.gitea/workflows/release.yml`):

1. Push a `v*` tag
2. CI builds the Web Components UI via `tsbundle`
3. `tsdeno compile` produces binaries for 4 platforms (linux-x64, linux-arm64, macos-x64, macos-arm64)
4. Binaries + SHA256 checksums are uploaded as Gitea release assets

Compile targets are configured in `npmextra.json` under `@git.zone/tsdeno`.

Storage Layout

Artifacts are stored in S3 at:

```
{storagePath}/{protocol}/packages/{packageName}/{version}/{filename}
```

For example: `packages/npm/packages/@myorg/mypackage/mypackage-1.0.0.tgz`

License and Legal Information

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @stack.gallery/registry

2026-03-22 - 1.8.5 - fix(registry)

restore protocol routing and test coverage for npm, oci, and api flows

- initialize and route REST API requests through ApiRouter alongside smartregistry
- add org-aware npm path handling and OCI bearer token endpoint support in the registry server
- enforce API token scopes in the auth provider instead of allowing all authenticated writes
- start the test server in integration and e2e suites and update assertions to match current API responses
- fix npm unpublish and OCI image test flows, including docker build loading and storage cleanup

2026-03-21 - 1.8.4 - fix(deps)

bump @stack.gallery/catalog to ^1.0.2 and remove committed test fixture auth token

- Updates the @stack.gallery/catalog dependency from ^1.0.1 to ^1.0.2.
- Removes the .npmrc auth token from the npm test fixture package to avoid keeping credentials in the repository.

2026-03-21 - 1.8.3 - fix(test-fixtures)

update npm fixture registry configuration for scoped package installs

- refreshes the test fixture auth token in .npmrc
- adds the @stack-test scoped registry mapping to the npm fixture configuration

2026-03-21 - 1.8.2 - fix(deps)

replace local catalog dependency with published version and simplify npm fixture auth config

- switch @stack.gallery/catalog from a local file reference to the published ^1.0.1 release
- update the npm test fixture .npmrc to use only an auth token entry

2026-03-21 - 1.8.1 - fix(release,test)

streamline release UI bundling and add npm fixture registry configuration

- Update the release workflow to build the UI with tsbundle directly instead of installing UI-specific dependencies and running a separate bundling script
- Add an .npmrc fixture for the demo npm package to configure the scoped registry and authentication token for local registry tests

2026-03-21 - 1.8.0 - feat(web)

add public package browsing and organization redirect management

- introduces a public packages view and root route behavior for unauthenticated users
- updates the app shell to support public browsing mode with an optional sign-in flow
- adds organization redirect state, fetching, and deletion in the organization detail view

2026-03-20 - 1.7.0 - feat(organization)

add organization rename redirects and redirect management endpoints

- add OrgRedirect model and resolve organizations by historical names
- support renaming organizations while preserving the previous handle as a redirect alias
- add typed requests to list and delete organization redirects with admin permission checks
- allow organization update actions to send name changes

2026-03-20 - 1.6.0 - feat(web-organizations)

add organization detail editing and isolate detail view state from global navigation

- adds an update organization action to persist organization detail edits from the detail view
- updates organization and package views to track selected detail entities locally instead of mutating global ui state
- preserves resolved app shell tabs for role-based filtering after async tab loading
- includes type-cast fixes for admin auth provider responses and bundled file Response bodies

2026-03-20 - 1.5.1 - fix(web-app)

update dashboard navigation to use the router directly and refresh admin tabs on login changes

- removes the global router workaround in the dashboard and imports appRouter directly
- re-filters resolved view tabs when login state changes so the Admin tab matches system admin access
- adds dashboard navigation support for the organizations view

2026-03-20 - 1.5.0 - feat(opsserver,web)

replace the Angular UI and REST management layer with a TypedRequest-based ops server and bundled web frontend

- add a new OpsServer with TypedRequest handlers for auth, organizations, repositories, packages, tokens, audit, admin, OAuth, and user settings flows
- introduce shared TypedRequest contracts under ts_interfaces and wire the registry to serve POST /typedrequest requests
- replace the embedded Angular build pipeline with tsbundle/tswatch-based web bundling, static html entrypoint, and new ts_web app state and shell views
- remove the legacy Angular frontend, custom UI bundler script, reload websocket hot-reload path, and related build configuration

2026-03-20 - 1.4.2 - fix(registry)

align registry integrations with updated auth, storage, repository, and audit models

- update smartregistry auth and storage provider implementations to match the current request, token, and storage hook APIs
- fix audit events for auth provider, platform settings, and external authentication flows to use dedicated event types
- adapt repository, organization, user, and package handlers to renamed model fields and revised repository visibility/protocol data
- add missing repository and team model fields plus helper methods needed by the updated API and permission flows
- correct AES-GCM crypto buffer handling and package version checksum mapping

2026-03-20 - 1.4.1 - fix(repo)

no changes to commit

2026-03-20 - 1.4.0 - feat(release,build,tests)

add automated multi-platform release pipeline and align runtime, model, and test updates

- add a Gitea release workflow that builds the UI, bundles embedded assets, cross-compiles binaries for Linux and macOS, generates checksums, and publishes release assets from version tags
- switch compilation to tsdeno with compile targets defined in npmextra.json and simplify project scripts for check, lint, format, and compile tasks
- improve CLI startup error handling in mod.ts and guard execution with import.meta.main
- update test configuration to load MongoDB and S3 settings from qenv-based environment files and adjust tests for renamed model and token APIs
- rename package search usage to searchPackages, update audit event names, and align package version fields and model name overrides with newer dependency behavior

2025-12-03 - 1.3.0 - feat(auth)

Add external authentication (OAuth/OIDC & LDAP) with admin management, UI, and encryption support

- Introduce external authentication models: AuthProvider, ExternalIdentity, PlatformSettings to store provider configs, links, and platform auth settings
- Add AuthProvider admin API (AdminAuthApi) to create/update/delete/test providers and manage platform auth settings
- Add public OAuth endpoints (OAuthApi) for listing providers, initiating OAuth flows, handling callbacks, and LDAP login
- Implement ExternalAuthService to orchestrate OAuth and LDAP flows, user provisioning, linking, session/token generation, and provider testing
- Add pluggable auth strategy pattern with OAuthStrategy and LdapStrategy plus AuthStrategyFactory to select appropriate strategy
- Add CryptoService for AES-256-GCM encryption/decryption of provider secrets and helper for key generation
- Extend AuthService and session/user handling to support tokens/sessions created by external auth flows and user provisioning flags
- Add UI: admin pages for managing auth providers (list, provider form, connection test) and login enhancements (SSO buttons, LDAP form, oauth-callback handler)
- Add client-side AdminAuthService for communicating with new admin auth endpoints and an adminGuard for route protection
- Register new API routes in ApiRouter and wire server-side handlers into the router
- Implement safeguards: mask secrets in admin responses, validate provider configs, and track connection test results and audit logs

2025-11-28 - 1.2.0 - feat(tokens)

Add support for organization-owned API tokens and org-level token management

- ApiToken model: added optional organizationId and createdById fields (persisted and indexed) and new static getOrgTokens method
- auth.interfaces: IApiToken and ICreateTokenDto updated to include organizationId and createdById where appropriate
- TokenService: create token options now accept organizationId and createdById; tokens store org and creator info; added getOrgTokens and revokeAllOrgTokens (with audit logging)
- API: TokenApi now integrates PermissionService to allow organization managers to list/revoke org-owned tokens; GET /api/v1/tokens accepts organizationId query param and token lookup checks org management permissions
- Router: PermissionService instantiated and passed to TokenApi
- UI: api.service types and methods updated — IToken and ITokenScope include organizationId/createdById; getTokens and createToken now support an organizationId parameter and scoped scopes

- .gitignore: added stories/ to ignore

2025-11-28 - 1.1.0 - feat(registry)

Add hot-reload websocket, embedded UI bundling, and multi-platform Deno build tasks

Introduce a ReloadSocketManager and client ReloadService for automatic page reloads when the server restarts. Serve UI assets from an embedded generated file and add Deno tasks to bundle the UI and compile native binaries for multiple platforms. Also update dev watch workflow and ignore generated embedded UI file.

- Add ReloadSocketManager (ts/reload-socket.ts) to broadcast a server instance ID to connected clients for hot-reload.
- Integrate reload socket into StackGalleryRegistry and expose WebSocket upgrade endpoint at /ws/reload.
- Add Angular ReloadService (ui/src/app/core/services/reload.service.ts) to connect to the reload WS and trigger page reloads with exponential reconnect.
- Serve static UI files from an embedded generated module (getEmbeddedFile) and add SPA fallback to index.html.
- Ignore generated embedded UI file (ts/embedded-ui.generated.ts) in .gitignore.
- Add Deno tasks in deno.json: bundle-ui, bundle-ui:watch, compile targets (linux/mac x64/arm64) and a release task to bundle + compile.
- Update package.json watch script to run BACKEND, UI and BUNDLER concurrently (deno task bundle-ui:watch).

2025-11-28 - 1.0.1 - fix(smartdata)

Bump @push.rocks/smartdata to ^7.0.13 in deno.json

- Updated deno.json imports mapping for @push.rocks/smartdata from ^7.0.9 to ^7.0.13

2025-11-28 - 1.0.0 - Initial release

Release with core features, UI, and project scaffolding.

- Implemented account settings and API tokens management.
  - SettingsComponent: user profile management (display name updates, password change).
  - TokensComponent: create and revoke API tokens.

- Application layout and navigation:
  - LayoutComponent for consistent layout, navigation, and user session display.
  - Main entry points added (index.html, main.ts).
- UI and styling:
  - Integrated Tailwind CSS for responsive design and styling.
  - UI updates and refinements.
  - Integrated toast notifications for user feedback.
- Code quality and tooling:
  - Refactored code structure for improved readability and maintainability.
  - Configured TypeScript with strict type checking and module resolution.
  - Dependency updates and fixes.

2025-11-27 - 2025-11-28 - unknown -> 1.0.0 - housekeeping / duplicate commits

Minor housekeeping and duplicate commits consolidated into the 1.0.0 release.

- Added initial README with project overview, features, and setup instructions.
- Consolidated a duplicate "feat: add account settings and API tokens management" commit (unknown version) into the 1.0.0 release.
- Miscellaneous UI tweaks and dependency updates.